ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

Desarrollo de videojuego para introducir a la resolución de problemas de programación, P-Learning

Elihú Salcedo Ruiz

22 de octubre de 2015

ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA EN INFORMÁTICA

Desarrollo de videojuego para introducir a la resolución de problemas de programación, P-Learning

- Departamento de Ingeniería Informática
- Director del proyecto: Juan Manuel Dodero Beardo
- Autor del proyecto: Elihú Salcedo Ruiz

Cádiz, 22 de octubre de 2015

Fdo: Elihú Salcedo Ruiz

Agradecimientos

Agradecimientos para Javier Osuna Herrera por haberme ayudado con sus videos y tutoriales de LibGdx, además de las tardes de café y portátil. También quería agradecer a Adrián Rodriguez Rivera por su gran aportación creando la música del juego y por su talento. Agradezco a Kenney de www.kenney.nl por su gran aportación ya que la mayoría del apartado gráfico del juego usa o deriva de su trabajo, compartido con licencia de dominio público. Por último a mi familia y amigos por aguantar todo el estrés que he desprendido en estos meses. ¡Muchas gracias!

Resumen

Este Trabajo de Fin de Grado, en adelante TFG, se centra en el desarrollo de un videojuego multiplataforma. Siendo las plataformas compatibles: Android, distribuciones de Linux y web principalmente. Ésto es así debido a que se desea desarrollar el juego con herramientas y recursos libres, por tanto las plataformas compatibles son principalmente plataformas de código abierto o con licencia libre.

Un videojuego es un software creado para el entretenimiento en general y basado en la interacción entre una o varias personas y un aparato electrónico que ejecuta dicho videojuego. Así pues un videojuego es considerado un software cuyo fin directo es el entretenimiento.

Con respecto a los aparatos que ejecutan este software, en estos últimos años los smartphones y tabletas se han convertido en las nuevas plataformas móviles para juegos que compiten con los sistemas portátiles clásicos que generalmente tienden a ser más cerrados. Aún así las plataformas más generalistas son tanto los ordenadores comunes como dispositivos especialmente diseñados para ejecutar videojuego.

Una vez introducidos estos conceptos generales, P-Learning Game es un videojuego educativo multiplataforma pensado para desarrollar las destrezas en lógica de niños y niñas de entre 8 y 12 años para así hacer que paulatinamente pueda resultarles más fácil aprender a programar en un lenguaje de programación secuencial y estructurado.

Este proyecto nace de la idea de que se pueden demostrar mejoras en resultados de aprendizaje mediante el juego y la gamificación.

Para entender este concepto, la gamificación se puede definir como el empleo de mecánicas de juego en entornos y aplicaciones no lúdicas con el fin de potenciar la motivación, la concentración, el esfuerzo, la fidelización y otros valores positivos comunes a todos los juegos.

Siendo este útlimo uno de los objetivos principales también es obvio que, para aprender a programar hay que resolver muchos problemas ya que la repetición es también un gran método de aprendizaje.

Uniendo ambos conceptos se concibe P-Learning Game, un juego de puzzles y lógica por niveles en el que se parte de una entrada y se pide una salida determinada, además tenemos diversas representaciones de estructuras de control que interactúan con las entradas y las manipulan para llegar a obtener esta salida.

Aquí es donde está más clara la correlación con la programación secuencial que, aún siendo sutil, podría tomarse como un lenguaje visual a muy alto nivel y de dominio muy específico.

Para cumplir con estos objetivos se ha desarrollado este videojuego que consiste en el uso de lenguaje visual en un dominio muy específico para que pueda ser fácilmente entendido por niños, por tanto se ha optado por formas geométricas sencillas y colores para representar las entradas y salidas, que en este caso son pelotas. Además se han introducido unos objetos fácilmente distinguibles y característicos para los controles y se juega con las relaciones de colores entre

entradas, controles y salidas para conseguir los objetivos del juego en sí. La secuencialidad, el flujo de control y opción a estructuras de selección se da con la representación de un laberinto que recorren las entradas y donde se colocan los controles.

Tras probar estos métodos de representación se han podido balancear para que sea a la vez lo suficientemente descriptivo y fácil de usar para niños llegando a cumplir muchos de los prerequisitos y objetivos iniciales planteados.

Palabras clave: videojuego educativo, juego de puzzles, lógica, puzzles, programación, libgdx, software libre, multiplataforma.

Índice general

Índice de figuras

Parte I

Prolegómeno

La primera parte de la memoria del TFG contiene una introducción y una planificación del proyecto.

Capítulo 1

Introducción

A continuación, se describe la motivación del presente proyecto y su alcance. También se incluye un glosario de términos y la organización del resto de la presente documentación.

1.1. Motivación

En los últimos años los diferentes países de la Unión Europea y Estados Unidos han visto la importancia de incluir la Programación como asignatura en las primeras etapas de los niveles educativos [eldiario.es, 2014]. Ven como los niños y niñas tienen capacidad de desarrollar suficientes destrezas para la implementación de programas sencillos y básicos además de aprender rápidamente.

Por ello han surgido lenguajes visuales como Scratch, desarrollado por el "Massachusetts Institute of Technology"[MIT, 2015] y otras iniciativas para hacer más fácil y natural la introducción a la programación mediante el juego [Periódico El Mundo, 2014].

En este caso se llegó a la conclusión de que una de las condiciones necesarias para aprender un lenguaje de programación es desarrollar la lógica matemática y la lógica proposicional, además de poder desarrollar mentalmente un sentido de secuencialidad.

Por todo ello comenzó el desarrollo de P-Learning Game como instrumento para desarrollar esas habilidades en lógica antes de comenzar con un lenguaje o pseudolenguaje más específico.

1.2. Alcance

El TFG es un videojuego orientado a un público con edades comprendidas entre 8 y 12 años. Sin embargo fue interesante incluir en el público objetivo a personas más mayores también.

Para conseguir ésto la solución que se ha adoptado es enfocar el juego visualmente para niños mientras que la dificultad de los ejercicios a resolver puede ser atractiva para un público más adulto. Sin embargo éste aspecto necesita ser balanceado frecuentemente.

El principal objetivo que se desea cumplir con este videojuego es que los usuarios vean verdaderamente incrementadas sus capacidades y habilidades en lógica matemática y proposicional con

el fin de que les sea más fácil aprender un lenguaje o pseudolenguaje de programación secuencial y estructurada.

El siguiente objetivo más prioritario es que, dado que es un desarrollo con licencia libre GPLv3, haya gente interesada en compartir y mejorar el videojuego en sí y que además sirva como herramienta de aprendizaje el mismo código fuente de éste, ya que se han seguido patrones de diseño y técnicas de ingeniería del software para realizarlo.

1.3. Glosario de Términos

Esta sección contiene una lista ordenada alfabéticamente de los principales términos, acrónimos y abreviaturas específicos del dominio del problema.

- **Controles:** En este dominio los controles van a ser los elementos con los que el usuario interactúa directamente en el juego para poder transformar las entradas en la salida deseada.
- **Entradas:** En este dominio las entradas son lo elementos que en cada nivel se van a aportar y los que recorrerán el laberinto para interactuar con los controles
- **LibGdx:** Es un framework para el desarrollo de videojuegos multiplataforma, soportando actualmente Windows, Linux, Mac OS X, Android, iOS y HTML5. Permite codificar en un único proyecto y exportarlo a las tecnologías mencionadas anteriormente sin modificar casi nada
- **PLearning:** Es el nombre que se la ha dado al videojuego y viene dado por la contracción de *Programming Learning*.
- **Salidas:** En este dominio las salidas son el objetivo que debe conseguir el usuario para ganar un nivel de juego
- **SCRUM:** El proceso Scrum se basa en ciclos de desarrollos cortos en los que los requisitos pueden ser variables y el proceso está orientado al desarrollo de prototipos incrementales y funcionales.

1.4. Organización del documento

Esta sección contiene una descripción de los contenidos de la presente memoria.
La primera parte de la memoria del TFG consta de una introducción y una planificación del proyecto.

La introducción es un capítulo que, a modo de resumen, contiene una breve descripción del contexto del proyecto y la motivación para su desarrollo, así como del alcance previsto.

La planificación deberá incluir los plazos, los entregables, los recursos y el método de ingeniería de software a emplear.

La segunda parte de la memoria consta de la documentación del desarrollo siguiendo los métodos de ingeniería asociados al desarrollo de software. Con apartados de requisitos, análisis, diseño,

construcción y pruebas.

En la útlima parte quedarán recogidas las conclusiones y los manuales necesarios para el manejo de la aplicación resultado del desarrollo.

Capítulo 2

Planificación

En esta sección se describen todos los aspectos relativos a la gestión del proyecto: metodología, organización, costes, planificación, riesgos y aseguramiento de la calidad.

2.1. Metodología de desarrollo

Un videojuego no deja de ser un proyecto de software, por tanto para la realización de este proyecto se ha querido poner en práctica una metodología ágil. En este caso SCRUM, modificado ligeramente a las necesidades del proyecto por ser realizado por una persona. Este ciclo de vida se compone de las siguientes etapas:

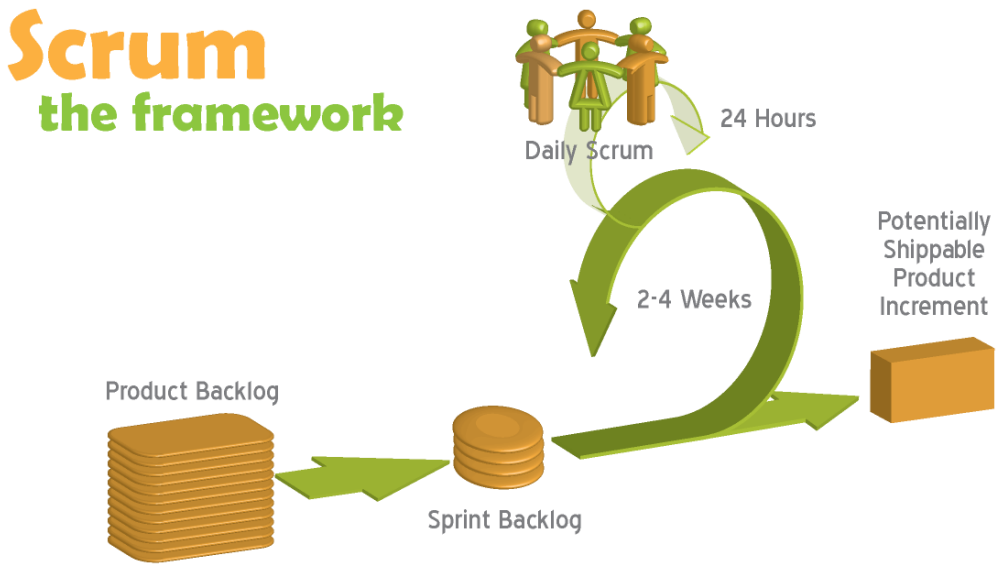

Figura 2.1: Ciclo de Scrum.

El proceso Scrum se basa en ciclos de desarrollos cortos en los que los requisitos pueden ser variables y el proceso está orientado al desarrollo de prototipos incrementales y funcionales.

Se parte de una lista priorizada de los requisitos. Cada requisito tendrá un nombre, una prioridad, una estimación inicial, un solicitante, notas, test.

Ésto se denomina *Product Backlog* y los requisitos están descritos en un lenguaje no técnico y priorizados por valor de negocio.

Antes de empezar el periodo de *Sprint*, El *Product Owner*, en este caso al ser un proyecto propio este rol lo toma el desarrollador principal, presenta el *Product Backlog* al equipo de desarrollo y este último determina cuales serán las funcionalidades desarrolladas durante el *Sprint* y eligiendo una fecha que no supere 4 semanas. Esta reunión se titula *Sprint Planning* y lo que resulta de ella es el *Sprint Backlog* que es la lista de las tareas necesarias para llevar a cabo las funcionalidades de la entrega parcial del *Sprint*.

Un *Sprint* queda definido por el equipo de trabajo y se compone de:

- Una descripción.
- Una fecha de finalización.
- Las funcionalidades que se desarrollarán durante el Sprint.
- La asignación de las funcionalidades a desarrollar.

Durante un *Sprint*, cada día se suele hacer un *Daily Scrum* (reuniones diarias) donde cada miembro del equipo expone lo que hizo el día anterior, lo que va a realizar hoy y si ha tenido algún problema durante su desarrollo. Esto permite que la cohesión del grupo sea mejor y más ágil.

Al final del Sprint, el equipo de desarrollo hace una presentación de las funcionalidades conseguidas y posteriormente se analiza lo que se hizo bien, cuales son los procesos que pueden ser mejorados y como perfeccionarlos.

Este TFG ha sido desarrollado sólo por un componente, por tanto para emular el proceso de desarrollo SCRUM se han adoptado diferentes roles y se han adaptado los ciclos de desarrollo a la metodología definida.

2.2. Planificación del proyecto

En los siguientes apartados se desarrolla la estimación temporal y definición del calendario básico. Además del desarrollo de la planificación detallada, utilizando un diagrama de Gantt.

2.2.1. Planificación de actividades

Análisis previo y planificación

En este punto se ha abordado la visión global del proyecto, comprendida en establecer los puntos iniciales necesarios para crear una planificación de tareas en función de las actividades a realizar.

Actividad	Descripción
Diseño conceptual del videojuego	Mecánicas del videojuego
Definición de las tecnologías	Tecnologías que se utilizaran en el desarrollo del videojuego
Análisis de riesgos	Posibles riesgos del proyecto
Planificación del proyecto	Crear una planificación adaptada a los plazos de entrega
Redacción del documento	Generación del plan de trabajo

Análisis de Requisitos

En esta fase se ha detallado que necesidades debe cubrir el videojuego, tanto desde un punto de vista funcional, como desde un punto de vista arquitectónico de la aplicación a diseñar, se ha diseñado un modelo troncal de clases para posteriormente desarrollar en base a estas, todo el videojuego. No obstante, en ningún caso se ha entrado en detalle en las tecnologías a utilizar que serán definidas más adelante.

Actividad	Descripción
Especificación de requisitos funcionales	Definición de casos de usos.
Diseño básico estructura aplicación	Diagrama básico de clases.

Diseño

En este apartado, teniendo en cuenta que en este proyecto se va a desarrollar un videojuego, se ha realizado la recopilación conceptual de arte, conjuntamente con el diseño de todas las pantallas que intervienen en el. Conjuntamente con la selección de los tipos de fuente que se utilizaran para el videojuego, selección de mundos y niveles y lógica de juego.

Actividad	Descripción
Definición de usuarios y contexto de uso del videojuego	Conocer las características de los usuarios, sus necesidades y objetivos, así como los contextos de uso.
Diseño conceptual	Un escenario de uso describe desde el punto de vista del usuario como será utilizado un producto determinado (es este caso un videojuego) en un escenario determinado.
Selección de arte	Selección de arte ya hecho con licencias libres ya que no se disponen de conocimientos de arte para abordar el diseño.
Diseño de pantallas que intervienen (prototipo)	Con todo el arte, fuentes y personajes, se realizara el diseño de todas las pantallas que intervienen en el juego. Sketches y prototipo de alta fidelidad.
Selección de fuentes	Selección de los tipos de fuente adecuados para el diseño del videojuego.
Redacción del documento	Generación del documento de diseño

Implementación y pruebas

En este punto se ha desarrollado la implementación del videojuego partiendo de todas las definiciones de los apartados anteriores y las verificaciones oportunas para asegurar el correcto

funcionamiento del videojuego.

Actividad	Descripción
Implementación de navegables y contenidos	Implementación de la estructura de todas las pantallas del videojuego y la navegabilidad entre ellas además de las funcionalidades principales de cada una de ellas.
Implementación del gameplay	Implementación de la logica de juego.
Implementación de sonido y efectos	Implementación de los efectos de sonido y la música.
Version incremental del videojuego	En este punto se dispone de la primera versión iterativa del videojuego.
Pruebas e incrementos en el Sprint	Pruebas y correciónes de errores. Incrementos en funcionalidades.
Preparación de los paquetes entregables	Pruebas de integración y despliegue.

Finalización

En un proyecto de software esta parte quedaría focalizada en la fase de implantación y despliegue. Sin embargo, al tratarse de un videojuego, esta fase quedaría focalizada en la fase de lanzamiento del producto. No obstante en este caso se ha utilizado para documentar todo el trabajo realizado a lo largo de este TFG.

Actividad	Descripción
Redacción memoria final proyecto	Redacción memoria final
Preparación presentación	Montaje presentación diapositivas
Demostración de funcionamiento	Montaje del video final y la demo para la presentación del proyecto
Entrega final	Entrega final

2.2.2. Estimación temporal y calendario detallado

El proyecto tiene una estimación temporal de 5 meses para una persona ya que se trata de un TFG con plazos marcados por el currículum académico del grado. Comienza en Mayo de 2015 y termina en Octubre de 2015.

En el siguiente apartado desglosaremos el tiempo que se planea para cada actividad midiendo el tiempo en persona/mes (p/m). Con una dedicación por persona de 8 horas al día en días laborales se puede inferir a un total de 160 horas mensuales:

- Análisis previo y planificación - 0.3 p/m
- Análisis de Requisitos - 0.3 p/m
- Diseño - 1 p/m
- Implementación y pruebas - 1.4 p/m
- Finalización - 0.4 p/m

El cálculo de fechas se tiene en cuenta excluyendo fines de semana.

En el diagrama 2.2 puede verse con mayor detalle las fechas planificadas para cada objetivo y subobjetivo.

2.2.3. Diagrama de Gantt

A continuación, en la figura 2.2, se adjunta el diagrama de Gantt correspondiente a la planificación del proyecto.

2.3. Organización

En los siguientes apartados se define la relación de roles involucradas en el proyecto y de cómo se estructuran las relaciones entre las mismas para ejecutar el proyecto. Además se define la relación de los recursos inventariables utilizados en el proyecto: equipamiento informático (hardware y software), herramientas empleadas, etc.

2.3.1. Roles y responsabilidades

Debido a la naturaleza de un TFG, se ha tenido que representar todos estos roles en diferentes momentos del proceso de desarrollo para así lograr un producto de calidad y siguiendo procesos de ingeniería, lo cual ofrece un valor añadido al proyecto.

Director de proyecto

La responsabilidad fundamental asignada a este rol será la de coordinar y resolver todos los conflictos que pudieran aparecer entre los demás miembros de un equipo de trabajo dado, además será responsable de todos los planes del proyecto de un proyecto dado, será el encargado de gestionar y hacer cumplir el plan de proyecto. También monitorizará todos los procesos que se estén llevando a cabo teniendo en cuenta todos los procesos de gestión de proyectos.

El director será supervisor de los miembros de su equipo, éstos se comunicarán con él, y él será el encargado de comunicarse con los jefes de departamento y directivos, en el caso de que fuese necesario. En el sentido opuesto, funcionará exactamente igual, es decir, los directivos se comunicarán con los directores de departamento y éstos, a su vez, se comunicarán con el jefe de proyecto. Éste informará a los demás miembros, si así fuese oportuno.

Analista

La responsabilidad de este rol será la de realizar la definición y análisis de requisitos y el diseño del sistema a desarrollar según el proyecto dado.

Además, será el encargado de transformar estos requisitos a requisitos técnicos (funcionales y no funcionales), y finalmente realizará el diseño del sistema partiendo de los requisitos de información que habrá capturado y demás información relevante. Será el encargado de liberar los

Figura 2.2: Diagrama de Gantt

documentos: *Documento de Requisitos del Sistema* y *Documento de Diseño del Sistema*.

Además diseñará el plan de pruebas del sistema.

Desarrollador

La responsabilidad de este rol será, realizar la implementación del sistema. Además, realizará la integración del sistema y su liberación. Una vez el sistema esté liberado, será el encargado de ofrecer al usuario final una serie de pautas para que pueda utilizar el sistema a satisfacción. Será el encargado de liberar los correspondientes manuales de uso (usuario y administrador) del producto software. Además, en nuestro caso particular, serán definidos como encargados de llevar a cabo la formación al cliente.

Tester

La responsabilidad de este rol será la de implementar el plan de pruebas asignado al proyecto y realizar todas las pruebas según la especificación para satisfacer las necesidades y los requisitos de calidad propuestos.

Diseñador Gráfico

La responsabilidad de este rol será la de acompañar al desarrollador encargado de la implementación del sistema realizando el diseño gráfico del sistema (logos, botones, esquemas de colores, estilos, etc.), es decir, todos los componentes gráficos necesarios para el desarrollo de los proyectos asignados. Ofrecen un punto de diseño artístico y de gran calidad visual para los sistemas a desarrollar.

2.3.2. Recursos y herramientas

Se hará uso de dos terminales con distribuciones Linux instaladas (en este caso Ubuntu 14.04 LTS.) como sistema operativo principal. Una de estas estaciones de trabajo será un portátil Toshiba Satellite Pro, mientras que la otra será un PC sobremesa. Además para las pruebas se usará un SmartPhone BQ Aquarius E5.
Se dispondrá de un servidor que aloje el software desarrollado con repositorio GIT, en este caso se hará uso de la plataforma GitHub como servidor principal de control de versiones.
Para la realización de este proyecto se han utilizado las siguientes herramientas con la restricción de que todas ellas debían ser libres:

- Diseño
  - Gimp 2.8
  - Inkscape 0.48
  - TexturePacker 3.2.0
- Desarrollo
  - Entorno de Desarrollo Eclipse 4.5 Mars
  - Android develompment tools ADT

    - Android SDK
    - LibGdx 1.6.5
    - Control de versiones Git, utilizando como plataforma GitHub
- Edición de audio
    - Audacity 2.1
- Redacción de documentos
    - Latex
    - LibreOffice 5

2.4. Riesgos

En esta sección se enumeraran los diferentes riesgos por orden de prioridad:

1. Se parte de una experiencia previa nula en desarrollo de videojuegos, no se dispone de experiencia con la librería escogida ni tampoco con programación sobre Android. Sin embargo se desea desarrollar este proyecto para lograr aprender y dominar en la medida de lo posible este campo. No obstante, existe la posibilidad de dedicar más tiempo del previsto consultando documentación para lograr los objetivos establecidos.

2. Existe un riesgo a nivel artístico, ya que no se dispone de ningún conocimiento de arte de ningún tipo. Por lo tanto, muy probablemente a nivel artístico el juego sea un poco pobre. Es por este motivo que durante la realización de este proyecto no se va a profundizar en temas de diseño artístico, en su lugar se utilizaran recursos de arte existentes en la red con licencias libres tipo creative commons o dominio público. Se hará una selección de distintos tipos de recursos gráficos para posteriormente hacer el montaje de toda la parte gráfica del videojuego.

3. En cuanto a la parte de audio la situación es exactamente la misma, no se dispone de ningún conocimiento para componer música ni efectos. Por lo tanto, se buscarán sonidos y temas musicales con licencias libres. Intentando que estos queden integrados de la mejor manera posible al videojuego.

4. Finalmente, existe el riesgo de que la lógica de juego definida en el diseño conceptual del videojuego, una vez puesta en práctica, no sea del todo funcional o necesite de un balanceo para conseguir un juego equilibrado y jugable. Esto se solucionará gracias a la metodología escogida que al ser incremental permite tomar decisiones funcionales en diferentes etapas del desarrollo.

Parte II

Desarrollo

En esta parte se debe describir el desarrollo del proyecto siguiendo la metodología empleada.

Capítulo 3

Requisitos del Sistema

En esta sección se detalla la situación que origina el desarrollo o mejora de un sistema informático. Luego se presentan los objetivos y el catálogo de requisitos del nuevo sistema. Finalmente se describen las diferentes alternativas tecnológicas y el análisis de la brecha entre los requisitos planteados y la solución base seleccionada.

3.1. Situación actual

Actualmente partimos con un desarrollo nuevo, por tanto más que de la situación actual del sistema podemos hablar de la situación actual del mercado objetivo para los videojuegos en general y para los videojuegos multiplataforma en particular.

Un videojuego es un software creado para el entretenimiento en general y basado en la interacción entre una o varias personas y un aparato electrónico que ejecuta dicho videojuego. Así pues un videojuego es considerado un software cuyo fin directo es el entretenimiento.

Los videojuegos se pueden clasificar en distintos géneros, estos son una forma de clasificación que designan un conjunto de videojuegos que poseen una serie de elementos comunes. A lo largo de la historia de los videojuegos aquellos elementos que han compartido varios de los mismos han servido para clasificar como un género a aquellos que les han seguido en estilo y forma, de la misma manera que ha pasado con la música o el cine. No se ha encontrado estudios certificados sobre estas categorías, pero se enumeran algunos de los géneros más populares: Lucha, Rol, Sigilo, Plataformas, Arcade, Deporte etc. [wikipedia, 2015]

En este caso nos vamos a centrar en un videojuego educativo para niños. No es el género más popular ya que no aparecen videojuegos educativos entre los más descargados [uptodown, 2015], ni entre los más vendidos en las principales plataformas [AEVI, 2015] pero sí puede ser una herramienta eficaz de aprendizaje.

Con la aparición de los teléfonos móviles inteligentes y tabletas se ha visto incrementado el consumo de videojuegos para nuevos perfiles de jugadores y jugadoras que antes no lo eran y además los videojuegos están siendo de las aplicaciones más descargadas en los mercados de aplicaciones de las plataformas móviles.

3.1.1. Entorno Tecnológico

Para este TFG definiremos en este apartado la importancia de incluir como plataforma compatible los dispositivos Android.

Android [Google, 2015] es un sistema operativo con núcleo monolítico basado en el kernel de Linux version 2.6 que fue lanzado al mercado el 23 de septiembre de 2008. Diseñado principalmente para dispositivos móviles con pantalla táctil, como pueden ser Smartphones o tabletas. Esta disponible gratuitamente para un uso comercial o no comercial. Soporta distintas estructuras de procesador, entre ellas ARM, x86, MIPS y IBM POWER, no obstante la arquitectura utilizada mayoritariamente son los procesadores ARM. Esta publicado bajo la licencia Apache 2.0 y GNU GPL2.

En el momento de su aparición tuvo que enfrentarse a las distintas plataformas ya existentes en aquel momento, iPhone OS que actualmente es conocido como iOS, BlackBerry OS y Windows Phone 7. El crecimiento de Android a sido espectacular en estos últimos años, a nivel mundial alcanzó una cuota de mercado del 50,9 % durante el último trimestre de 2011, esto represento más del doble que el sistema operativo iOS de Apple. Actualmente, su cuota de mercado se sitúa en el 74,4 % según un estudio realizado por la empresa Gartner [Gartner, 2013]. Sin embargo en España llega a cuotas de más del 80 %.

Para hacer más sencilla la implementación del videojuego y su exportación a diferentes plataformas se ha decidido usar el framework LibGdx.

LibGDX es un framework multiplataforma de desarrollo de juegos para Windows, Linux y Android. Está escrito en Java con una mezcla de C/C++ para dar soporte y rendimiento a tareas relacionadas con el uso de la física y procesamiento de audio.
LibGDX permite generar una aplicación en el PC y utilizar el mismo código en Android, de esta manera el proceso de pruebas y depuración se realiza de forma más rápida y cómoda ya que el PC es mucho más rápido que el dispositivo Android.

Con LibGDX aseguramos que la misma aplicación puede funcionar correctamente en diferentes dispositivos.

3.1.2. Fortalezas y Debilidades

Esta sección contiene información sobre los aspectos positivos y negativos del negocio actual de la organización para la que se va a desarrollar el sistema software.
Hoy en día el mercado de videojuegos es muy variado y extenso, sin embargo no deja de ser una gran oportunidad de negocio ya que los videojuegos son las aplicaciones más descargas y vendidas tanto en los markets de aplicaciones tanto para móviles como para otras plataformas [AEVI, 2015].
Existen diferentes videojuegos educativos de calidad y lucrativos. Sin embargo es cierto que el público objetivo es muy específico y con menos oportunidades comerciales ya que las temáticas de juego más populares no suelen concordar con un rol educativo.
Además un videojuego educativo o que fomente ciertas habilidades conceptuales prueba ser un gran método de aprendizaje aunque siempre complementario a otras prácticas docentes.

3.2. Objetivos del Sistema

Esta sección contiene la especificación de los objetivos o requisitos generales del sistema. Los objetivos principales son los siguientes y estarán especificados en las siguientes figuras:

- OBJ-01 Menú principal del juego
- OBJ-02 Arte, diseño gráfico y sonido
- OBJ-03 Pantalla de ayuda del juego
- OBJ-04 Pantalla de opciones del juego
- OBJ-05 Pantalla de selección de mundo y nivel del juego
- OBJ-06 Lógica de juego
- OBJ-07 Gestión de los controles del juego
- OBJ-08 Gestión de las entradas y salidas del juego
- OBJ-09 Captura de objetivos del juego
- OBJ-10 Gestión de los laberintos del juego
- OBJ-11 Gestión de niveles del juego
- OBJ-12 Gestión de las puntuaciones y premios del jugador

OBJ-01	Menú principal del juego
Autor	Elihú Salcedo
Descripción	El sistema deberá ofrecer un menú principal de juego navegable, usable y que permita interaccionar con el resto de pantallas de juego
Prioridad	Alta

OBJ-02	Arte, diseño gráfico y sonido
Autor	Elihú Salcedo
Descripción	Se deberá desarrollar o usar arte que siga la línea estética del juego y permita una mayor inmersión
Prioridad	Alta

OBJ-03	Pantalla de ayuda del juego
Autor	Elihú Salcedo
Descripción	El sistema deberá ofrecer una pantalla de ayuda para introducir al usuario al uso de éste además deberá ser navegable y accesible
Prioridad	Media

OBJ-04	Pantalla de opciones del juego
Autor	Elihú Salcedo
Descripción	El sistema deberá ofrecer una pantalla de opciones para que el usuario defina las diferentes configuraciones de sonido que desea usar
Prioridad	Media

OBJ-05	Pantalla de selección de mundo y nivel del juego
Autor	Elihú Salcedo
Descripción	El sistema deberá ofrecer la posibilidad de seleccionar entre diferentes mundos y niveles predefinidos para comenzar el juego
Prioridad	Alta

OBJ-06	Lógica de juego
Autor	Elihú Salcedo
Descripción	El sistema tendrá una logica de juego consistente en dos partes. Una para colocar controles en el escenario y otra para que automáticamente caigan las entradas para interactuar con los controles
Prioridad	Alta

OBJ-07	Gestión de los controles del juego
Autor	Elihú Salcedo
Descripción	El sistema deberá tener definidos unos controles que proporcionen diferentes comportamientos a las entradas en los diferentes escenarios de juego
Prioridad	Alta

OBJ-08	Gestión de las entradas y salidas del juego
Autor	Elihú Salcedo
Descripción	El sistema deberá tener definidas las diferentes entradas y salidas para cada nivel y que éstas puedan interactuar con los controles para que exista solución al problema
Prioridad	Alta

OBJ-09	Captura de objetivos del juego
Autor	Elihú Salcedo
Descripción	El sistema deberá contemplar para cada nivel una o varias soluciones posibles que puedan ser capturadas a la finalización del nivel y así dar respuesta al usuario
Prioridad	Alta

OBJ-10	Gestión de los laberintos del juego
Autor	Elihú Salcedo
Descripción	El sistema deberá tener definidos los diferentes laberintos que pueden ser accesibles en cada mundo además se gestionará que estos se formen de una forma específica y sin posibilidad de crear puntos muertos
Prioridad	Media

OBJ-11	Gestión de niveles del juego
Autor	Elihú Salcedo
Descripción	El sistema deberá tener definidos los diferentes niveles que pueden ser accesibles en cada mundo además se gestionará que estos contengan y controlen las diferentes entras y salidas posibles, los controles y los laberintos asociados
Prioridad	Alta

OBJ-12	Gestión de las puntuaciones y premios del jugador
Autor	Elihú Salcedo
Descripción	El sistema deberá tener definidas diferentes puntuaciones para cada nivel acabado con éxito y un sistema de recompensa para más adelante servir como moneda de pago en la accesibilidad de los mundos posteriores
Prioridad	Baja

3.3. Catálogo de Requisitos

Esta sección contiene la descripción del conjunto de requisitos específicos del sistema a desarrollar.

3.3.1. Requisitos funcionales

Descripción completa de la funcionalidad que ofrece el sistema. Los requisitos irán catalogados por cada objetivo al que hacen referencia.

- **OBJ-01** Menú principal del juego
  - **RQ-01** Mostrar selección de mundo.
  - **RQ-02** Mostrar opciones de sonido.
  - **RQ-03** Mostrar ayuda.
  - **RQ-04** Salir de la aplicación.
- **OBJ-04** Pantalla de opciones del juego
  - **RQ-05** Seleccionar volumen de música y sonido.
  - **RQ-06** Silenciar música y sonido.
- **OBJ-05** Pantalla de selección de mundo y nivel del juego

    - **RQ-07** Seleccionar mundo
- **OBJ-06** Lógica de juego
    - **RQ-08** Colocación de controles en laberinto.
    - **RQ-09** Gestión de tiempo hasta empezar.
    - **RQ-10** Gestión de movimiento de entradas en laberinto.
    - **RQ-11** Gestión de colisiones de controles y entradas.
    - **RQ-12** Gestión de activación de comportamiento de controles.
- **OBJ-07** Gestión de los controles del juego
    - **RQ-13** Gestión de colocación de controles en posiciones iniciales.
    - **RQ-14** Gestión de comportamientos de controles.
- **OBJ-08** Gestión de las entradas y salidas del juego
    - **RQ-15** Gestión de colocación de entradas y salidas en posiciones iniciales.
    - **RQ-16** Gestión de astributos de entradas y salidas.
- **OBJ-09** Captura de objetivos del juego
    - **RQ-17** Gestión de inicio del recorrido de las entradas en laberinto.
    - **RQ-18** Gestión de captura de salidas del laberinto y comparación.
- **OBJ-10** Gestión de los laberintos del juego
    - **RQ-19** Generador de laberintos.
    - **RQ-20** Verificador de laberintos.

3.3.2. Requisitos no funcionales

Descripción de otros requisitos (relacionados con la calidad del software) que el sistema deberá satisfacer.

- **NFR-01 Portabilidad.** El sistema deberá poder ejecutarse en diferentes sistemas operativos. En las diferentes distribuciones de linux y Android principalmente ya que son software libre o abierto.
- **NFR-02 Multiplataformas.** El sistema deberá poder ejecutarse en diferentes plataformas ya sea web, escritorio o en dispositivos móviles y tablets.
- **NFR-03 Usabilidad.** El sistema deberá ser usable, por ello deberá tener una navegación entre pantallas fluida e intuitiva, además de unos tamaños y resolución adecuados.
- **NFR-04 Dependencias libres.** El sistema deberá desarrollarse únicamente con herramientas y recursos de software libre y sus dependencias serán también de software libre.
- **NFR-01 Escalabilidad.** El sistema deberá ser escalable, es decir. Su arquitectura permitirá hacer mejoras y actualizaciones sin que se sufra un gran impacto o haciendo cambios mínimos.

3.3.3. Reglas de negocio

La restricción o regla más importante que viene impuesta al desarrollo del sistema ya se ha mencionado en el requisito no funciona *NFR-04 Dependencias libres*. Ésto es importante ya que el alumno ha pertenecido a la Oficina de Software Libre de la Universidad de Cádiz como becario durante un año y ha aprendido las virtudes y beneficios del uso del software libre.
Por tanto se considera que, dado que es un videojuego educativo, y los recursos educativos deberían de ser de libre acceso, la mejor forma de potenciar el uso y la mejora del mismo sistema es mediante el uso de software y tecnologías libres.

3.3.4. Requisitos de información

En esta sección se describen los requisitos de gestión de información (datos) que el sistema debe gestionar.
El sistema debe gestionar los datos referentes a la creación de niveles y todos sus atributos de forma que por cada instancia de la aplicación ejecutada existan los mismos niveles con los mismos atributos.

Para ello debemos gestionar los siguientes requisitos de información para cada nivel:

- **IRQ-01.** Configuración del laberinto. Se debe almacenar los parámetros de construcción de laberintos para cada nivel de juego.
- **IRQ-02.** Entradas en forma de secuencia de pelotas de diferentes colores. La información sobre las entradas de cada nivel deberá ser almacenada con el fin de recuperarlas en la carga.
- **IRQ-03.** Salidas en forma de secuencia de pelotas de diferentes colores. La información sobre las salidas de cada nivel deberá ser almacenada con el fin de recuperarlas en la carga.
- **IRQ-04.** Controles, siendo un subconjunto del total disponible en el sistema. La información sobre los controles de cada nivel deberá ser almacenada con el fin de recuperarlas en la carga.

Las restricciones que se aplican a estos requisitos son:

- Existe un máximo de entradas y salidas, siendo éste seis elementos
- Existe un máximo de controles disponibles para cada nivel, siendo éste siete
- La configuración del laberinto debe dar la posibilidad de alcanzar la meta a las entradas siempre

3.4. Alternativas de Solución

En esta sección, se ofrece un estudio del arte de las diferentes alternativas tecnológicas que permitan satisfacer los requerimientos del sistema, para luego seleccionar la herramienta o conjunto de herramientas que utilizaremos como base para el software a desarrollar.
Existen diferentes tecnologías disponibles para desarrollar videojuegos multiplataforma hoy en día.

3.4.1. Libgdx

libgdx
Libgdx es un framework java de desarrollo de juegos libre que provee una API unificada que trabaja sobre una serie de plataformas soportadas.
El framework provee de un entorno desarrollo para un prototipado rápido e iteraciones rápidas. En lugar de desplegar en Android/iOS/Javascript después de cada cambio en el código, se pueden ejecutar y depurar en tu propio escritorio de forma nativa.
Libgdx intenta no ser una única solución. No fuerza un diseño específico.

3.4.2. HTML5, CSS3, JavaScript

Con estos estándares libres se pueden realizar aplicaciones web fácilmente exportables a diferentes plataformas.
Interesa esta opción para realizar un desarrollo totalmente personalizado, ampliable mediante diferentes frameworks.
Framework JavaScript para juegos sencillos Completo framework javascript para juegos Framework Javascript
Además interesa obtener una envoltura para la exportación a otras plataformas.
CocconJS PhoneGap
También existe la opción de usar un engine completo. Game Closure

3.4.3. Unity3D

Unity3D
Unity es un ecosistema de desarrollo de juegos: un motor de renderizado totalmente integrado con un conjunto completo de herramientas y flujos de trabajo rápido para crear contenido 3D interactivo; publicación multiplataforma; miles de activos, listos para usar en la Tienda de Activos y una Comunidad donde se intercambian conocimientos.

3.5. Solución Propuesta

Se ha optado por utilizar LibGdx, ya que es una solución de software libre y ésta ha sido una de las restricciones principales para el desarrollo del sistema. Además está ampliamente desarrollado y probado y existe mucha documentación y bibliografía que ayuda a bajar la curva de aprendizaje.

Esta opción permite un desarrollo en Java que es uno de los lenguajes enseñados en la carrera.

Por todo ésto un desarrollo con LibGdx ha sido seleccionado como la mejor solución dadas las restricciones del proyecto.

Capítulo 4

Análisis del Sistema

Esta sección cubre el análisis del sistema de información a desarrollar, haciendo uso del lenguaje de modelado UML.

4.1. Modelo Conceptual

A partir de los requisitos de información, se desarrollará un diagrama conceptual de clases UML.

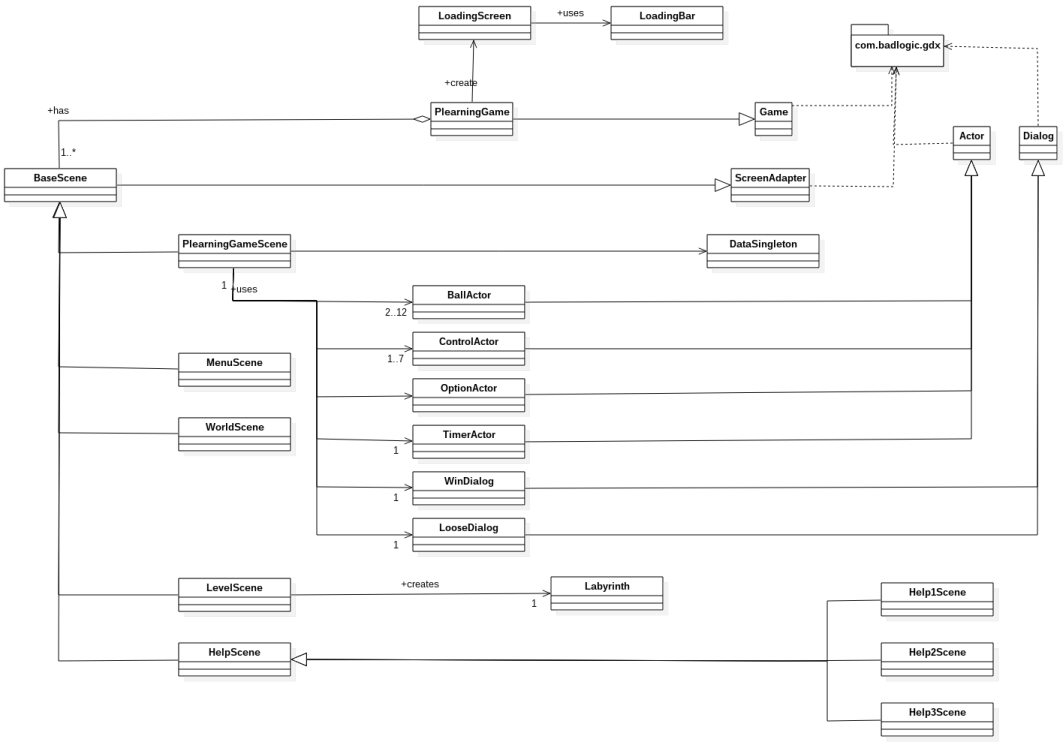

Figura 4.1: Diagrama conceptual de clases.

A partir del diagrama de clases que se muestra en la figura 4.2, se comenzará el diseño de la aplicación para su posterior implementación.

Paso a explicar, a modo de resumen, las clases que aquí se muestran:

- PlearningGame. Es la clase principal del juego. Desde aquí se inicializará la aplicación
- Las clases que contienen la palabra *Scene* hacen referencia a un tipo de diseño para videojuegos para el componente Scene2D de LibGdx en el que se contempla la clase como un contenedor de *actores* que pueden en un momento dado realizar *acciones*. Ésto se explica en profundidad en el libro [Cejas Sánchez and Saltares Márquez, 2014].
- Las clases que contienen la palabra *Actor* también son requisitos de diseño para Scene2D
- DataSingleton. Será la clase que maneje todos los datos físicos necesarios para el funcionamiento de la aplicación y para las configuraciones iniciales
- Labyrinth. Será la clase encargada de implementar los laberintos de los niveles

4.2. Modelo de Casos de Uso

A partir de los requisitos funcionales descritos anteriormente, se emplearan los casos de uso como mecanismo para representar las interacciones entre los actores y el sistema bajo estudio. Para cada caso de uso deberá indicarse los actores implicados, las precondiciones y postcondiciones, los pasos que conforman el escenario principal y el conjunto de posibles escenarios alternativos.

Comenzaremos entonces por el diagrama de casos de uso que describe las relaciones entre los actores, el sistema y los casos de uso de éste.

4.2.1. Actores

Como podemos observar en la figura 4.2, en el sistema existen tres actores principales: el jugador, el tiempo y la IA del juego.

ACT-01	**Jugador**
Descripción	Este rol lo llevará a cabo la persona que juegue al juego

ACT-02	**Tiempo**
Descripción	Este rol lo llevará a cabo el propio sistema. Consiste en el tiempo desde que se ha comenzado el nivel y está representado por un contador en la pantalla.

ACT-03	**IA**
Descripción	Este rol lo llevará a cabo el propio sistema. efectúa las operaciones características que se llevan a cabo en cada nivel según la configuración de entradas, salidas y controles que ha desplegado el jugador.

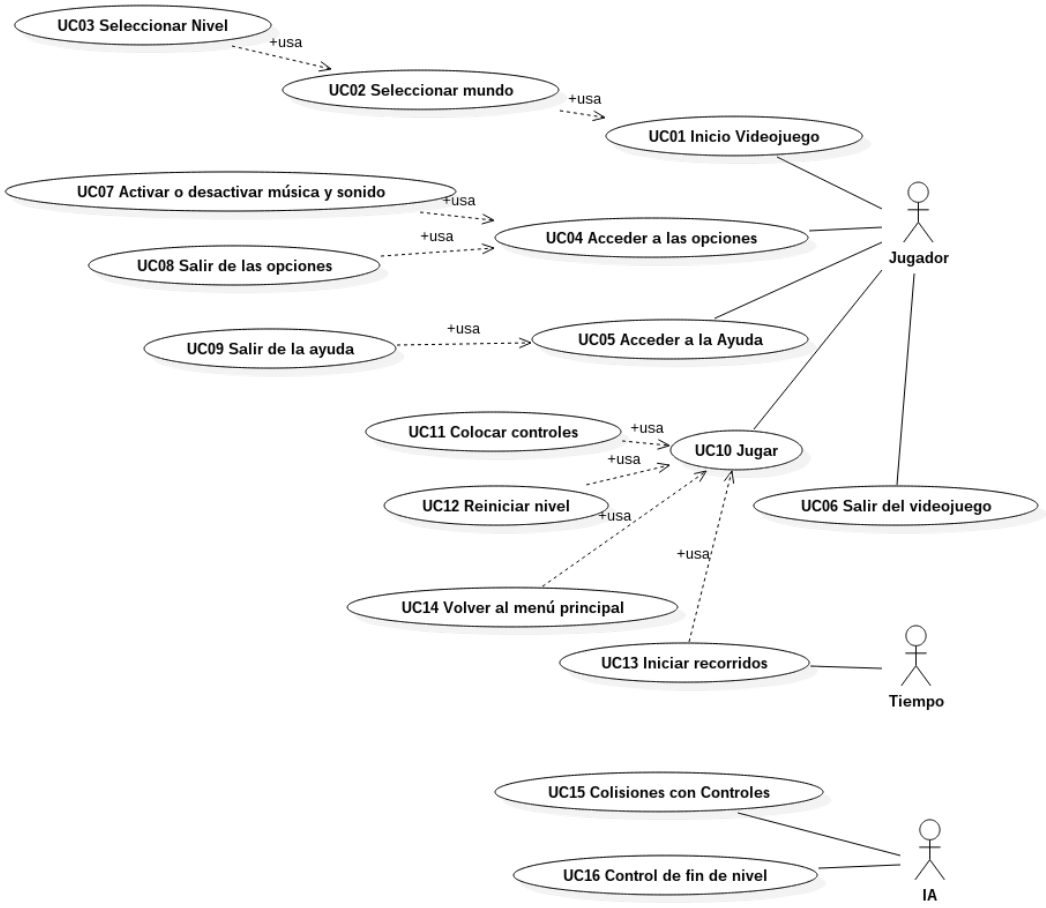

Figura 4.2: Diagrama de casos de uso.

4.2.2. Resumen de casos de uso

En la siguiente tabla se expone un resumen de los casos de uso del sistema.

Código	Nombre	Actor
UC-01	Inicio Videojuego	AC-01 Jugador
UC-02	Seleccionar Mundo	AC-01 Jugador
UC-03	Seleccionar Nivel	AC-01 Jugador
UC-04	Acceder a las Opciones	AC-01 Jugador
UC-05	Acceder a la Ayuda	AC-01 Jugador
UC-06	Salir del Videojuego	AC-01 Jugador
UC-07	Activar o desactivar Música y Sonido	AC-01 Jugador
UC-08	Salir de las Opciones	AC-01 Jugador
UC-09	Salir de la Ayuda	AC-01 Jugador
UC-10	Jugar al Videojuego	AC-01 Jugador
UC-11	Colocar Controles	AC-01 Jugador
UC-12	Reiniciar Nivel	AC-01 Jugador
UC-13	Iniciar Recorridos de Etradas	AC-01 Jugador o AC-02 Tiempo
UC-14	Volver al Menú Principal	AC-01 Jugador
UC-15	Colisiones de Controles	AC-03 IA
UC-16	Control de Fin de Nivel	

4.2.3. Descripción de casos de uso

A continuación se hará una descricpción más detallada de cada caso de uso.

Identificador	**UC-01**
Nombre	Inicio Videojuego
Código Requisito	RQ-01, RQ-02, RQ-03, RQ-04
Autor	Elihú Salcedo
Descripción	Inicia el videojuego
Actor	Jugador
Precondiciones	-
Postcondiciones	El usuario ha iniciado el juego y se encuentra en la pantalla principal
Flujo Principal	1. El usuario accede a la aplicación 2. El sistema se carga 3. El sistema muestra la pantalla principal
Flujos Alternativos	-
Inclusiones	-
Extensiones	Jugar al Videojuego

Identificador	**UC-02**
Nombre	Seleccionar Mundo
Código Requisito	RQ-07
Autor	Elihú Salcedo
Descripción	Permite seleccionar un mundo de juego
Actor	Jugador
Precondiciones	Se ha iniciado el videojuego y se ha pulsado el botón jugar
Postcondiciones	El usuario ha seleccionado un mundo y se permite escoger uno de sus niveles
Flujo Principal	1. El jugador presiona el botón de selección de mundos 2. El sistema carga las configuraciones de ese mundo 3. El sistema muestra la pantalla de selección de nivel de ese mundo
Flujos Alternativos	-
Inclusiones	-
Extensiones	-

Identificador	**UC-03**
Nombre	Seleccionar Nivel
Código Requisito	RQ-07
Autor	Elihú Salcedo
Descripción	Permite seleccionar un nivel de juego
Actor	Jugador
Precondiciones	Se ha seleccionado un mundo
Postcondiciones	El jugador ha seleccionado un nivel y se pasa a la pantalla de juego de éste
Flujo Principal	1. El jugador presiona el botón de selección de nivel 2. El sistema carga las configuraciones de ese nivel 3. El sistema muestra la pantalla de juego del nivel seleccionado
Flujos Alternativos	-
Inclusiones	-
Extensiones	-

Identificador	**UC-04**
Nombre	Acceder a las Opciones
Código Requisito	RQ-02
Autor	Elihú Salcedo
Descripción	Permite acceder a la pantalla de configuración de opciones del videojuego
Actor	Jugador
Precondiciones	Se está en la pantalla de menú principal
Postcondiciones	El jugador ha accedido a la pantalla de configuración de opciones
Flujo Principal	1. El jugador presiona el botón de opciones 2. El sistema carga las configuraciones actuales 3. El sistema muestra la pantalla de configuración de opciones
Flujos Alternativos	-
Inclusiones	-
Extensiones	-

Identificador	**UC-05**
Nombre	Acceder a la Ayuda
Código Requisito	RQ-03
Autor	Elihú Salcedo
Descripción	Permite acceder a la pantalla de ayuda del videojuego
Actor	Jugador
Precondiciones	Se está en la pantalla de menú principal
Postcondiciones	El jugador ha accedido a la pantalla de ayuda
Flujo Principal	1. El jugador presiona el botón de ayuda 2. El sistema carga las pantallas de ayuda 3. El sistema muestra las pantallas de ayuda
Flujos Alternativos	-
Inclusiones	-
Extensiones	-

Identificador	**UC-06**
Nombre	Salir del videojuego
Código Requisito	RQ-04
Autor	Elihú Salcedo
Descripción	Permite terminar de ejecutar el videojuego
Actor	Jugador
Precondiciones	Se está en la pantalla de menú principal
Postcondiciones	El sistema termina su ejecución sin errores
Flujo Principal	1. El jugador presiona el botón de salir 2. El sistema requiere confirmación 3. El jugador acepta
Flujos Alternativos	-
Inclusiones	-
Extensiones	-

Identificador	**UC-07**
Nombre	Activar o desactivar música o sonido
Código Requisito	RQ-05, RQ-06
Autor	Elihú Salcedo
Descripción	Permite silenciar el sonido en el videojuego
Actor	Jugador
Precondiciones	Se está en la pantalla de opciones
Postcondiciones	El sistema guarda y ejecuta las configuraciones de audio
Flujo Principal	1a. El jugador presiona el botón de silenciar 2. El sistema guarda la configuración y la ejecuta 3. El jugador termina la configuración
Flujos Alternativos	1b. El jugador ajusta el volumen de la aplicación 1c. El jugador no hace cambios en la configuración
Inclusiones	-
Extensiones	-

Identificador	**UC-08**
Nombre	Salir de las Opciones
Código Requisito	RQ-05, RQ-06
Autor	Elihú Salcedo
Descripción	Permite volver al menú principal
Actor	Jugador
Precondiciones	Se está en la pantalla de opciones
Postcondiciones	Se vuelve a la pantalla de menú principal
Flujo Principal	1. El jugador presiona el botón de volver 2. El sistema guarda la configuración y la ejecuta 3. El sistema muestra la pantalla de menú principal
Flujos Alternativos	-
Inclusiones	-
Extensiones	-

Identificador	**UC-09**
Nombre	Salir de la Ayuda
Código Requisito	RQ-03
Autor	Elihú Salcedo
Descripción	Permite volver al menú principal
Actor	Jugador
Precondiciones	Se está en la pantalla de ayuda
Postcondiciones	Se vuelve a la pantalla de menú principal
Flujo Principal	1. El jugador presiona el botón de volver 2. El sistema muestra la pantalla de menú principal
Flujos Alternativos	-
Inclusiones	-
Extensiones	-

Identificador	**UC-10**
Nombre	Jugar
Código Requisito	OBJ-06
Autor	Elihú Salcedo
Descripción	Se desarrolla la lógica de juego
Actor	Jugador
Precondiciones	Se ha seleccionado un nivel
Postcondiciones	El nivel termina como ganado o perdido
Flujo Principal	1. El jugador observa las entradas dadas 2. El jugador observa las salidas deseadas 3. El jugador observa el laberinto 4. El jugador observa los controles 5a. El jugador despliega en el laberinto un subconjunto de controles para dar con la solución correcta de que las entradas dadas se conviertan en las salidas pedidas 6. El sistema realiza el recorrido de las entradas en el laberinto 7a. El sistema envía un mensaje de que se ha ganado el nivel 8a. El sistema muestra la pantalla de selección de nivel
Flujos Alternativos	5b. El jugador despliega en el laberinto un subconjunto de controles que no dan la solución correcta de que las entradas dadas se conviertan en las salidas pedidas 7b. El sistema envía un mensaje de que se ha perdido el nivel 7b. El sistema permite reiniciar el nivel
Inclusiones	-
Extensiones	-

Identificador	**UC-11**
Nombre	Colocar Controles
Código Requisito	RQ-08
Autor	Elihú Salcedo
Descripción	Se permite colocar los controles en el laberinto
Actor	Jugador
Precondiciones	Se ha seleccionado un nivel, aún no ha acabado el tiempo o empezado los recorridos de las entradas
Postcondiciones	Se da paso a que las entradas hagan el recorrido del laberinto
Flujo Principal	1.El jugador despliega en el laberinto un control 2. El sistema verifica si las posiciones son válidas 3. El sistema coloca el control en la posición si es válida 4. Se vuelve a 1 hasta que el jugador no desee colocar más controles 5. Termina el caso de uso
Flujos Alternativos	-
Inclusiones	-
Extensiones	-

Identificador	**UC-12**
Nombre	Colocar Controles
Código Requisito	RQ-08
Autor	Elihú Salcedo
Descripción	Se permite reiniciar el nivel
Actor	Jugador
Precondiciones	Se ha seleccionado un nivel, aún no ha acabado el tiempo o empezado los recorridos de las entradas
Postcondiciones	Se vuelve a comenzar el nivel
Flujo Principal	1.El jugador presiona el botón de reinicio 2. El sistema carga la configuración inicial 3. El sistema reinicia el nivel
Flujos Alternativos	-
Inclusiones	-
Extensiones	-

4.3. Modelo de Interfaz de Usuario

En esta sección se incluye un prototipo de baja fidelidad o mockup de la interfaz de usuario del sistema.

En las siguientes figuras se puede observar tanto los prototipos de las diferentes pantallas del juego como la navegación entre éstas.

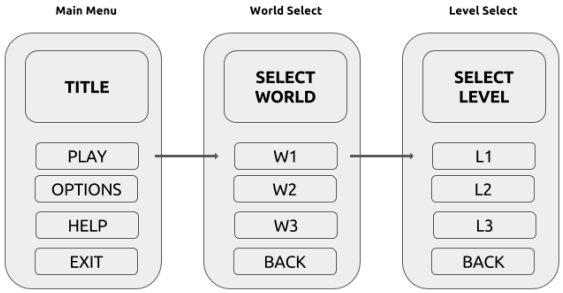

Figura 4.3: Menú principal.

En la figura 4.3 se muestra el menú principal de usuario y su navegabilidad al pulsar en el botón de jugar.

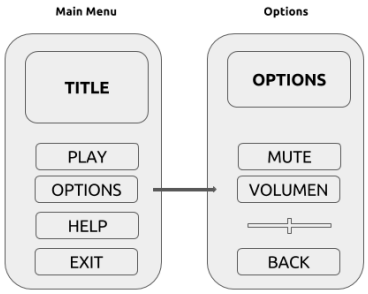

Figura 4.4: Menú de opciones.

En la figura 4.4 se muestra el menú de opciones y su navegabilidad desde el menú principal.

En la figura 4.5 se muestra el menú de ayuda y su navegabilidad desde el menú principal.

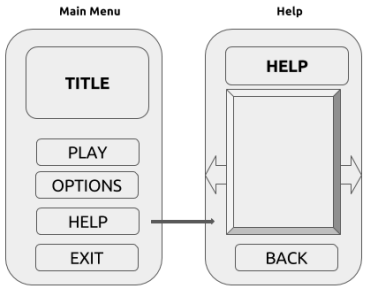

Figura 4.5: Menú de ayuda.

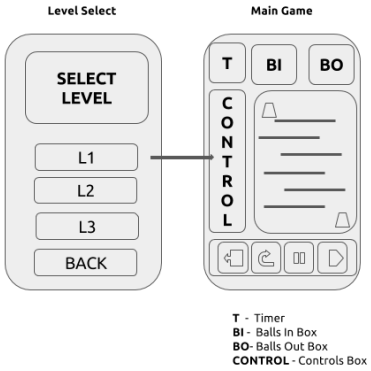

Figura 4.6: Menú de juego.

En la figura 4.6 se muestra la pantalla de juego junto con una pequeña leyenda de sus componentes principales.

En la figura 4.7 se muestra la navegación de volver al menú principal desde el juego.

Con ésto se tiene una idea aproximada de la interfaz que va a llevar el videojuego.

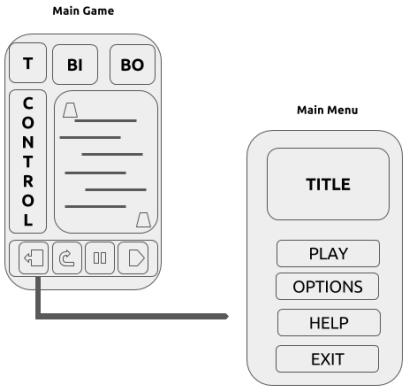

Figura 4.7: Menú de salir.

Capítulo 5

Diseño del Sistema

En esta sección se recoge la arquitectura general del sistema de información, la parametrización del software base (opcional), el diseño físico de datos, el diseño detallado de componentes software y el diseño detallado de la interfaz de usuario.

5.1. Arquitectura del Sistema

En esta sección se define la arquitectura general del sistema de información, especificando la infraestructura tecnológica necesaria para dar soporte al software y la estructura de los componentes que lo forman.

Una aplicación convencional sigue una secuencia lógica lineal, reacciona cuando sucede algún evento, es decir responde a eventos como por ejemplo cuando se pulsa una tecla. Por el contrario un videojuego es más complejo, hace cálculos y dibuja en pantalla constantemente, aunque no se produzca ningún evento.

5.1.1. Arquitectura Física

En este apartado, describimos los principales elementos hardware que forman la arquitectura física de nuestro sistema, recogiendo por un lado los componentes del entorno de producción y los componentes de cliente.

El sistema se despliega en cada cliente, por lo que en esta versión no es necesario un servidor ni un hardware específico más que el entorno de cliente.
Para este caso, el videjouego se ha desarrollado de forma que pueda ejecutarse en diferentes plataformas:

- PC, con sistema operativo Ubuntu, Mac o Windows
- Smartphone, con sistema operativo Android
- Web

Sí es cierto que para un correcto funcionamiento la máquina cliente debe cumplir al menos los siguientes requisitos mínimos de hardware:

- Procesador: mononúcleo a 1GHz de velocidad
- Memoria RAM: 1GB
- Memoria Secundaria: al menos 30MB

Para el despliegue en un entorno de escritorio partimos de la ventaja de que el ejecutable es un .jar, por tanto será necesario tener instalado Java en su versión 7 o una alternativa libre como OpenJDK. Para ejecutarlo simplemente hacer doble click o en una terminal escribir:

```
$ java -jar plearning
```

5.1.2. Arquitectura Lógica

Desarrollar videojuegos es una forma completamente diferente de desarrollo de software. La diferencia principal entre una aplicación convencional y un videojuego radica en la gestión de eventos. El las aplicaciones convencionales se ofrece respuestas a eventos y, una vez son servidos, permanece a la espera de una nueva orden, en cambio, un videojuego es un programa que debe actuar en tiempo real, tiene que estar haciendo cálculos y dibujando en pantalla constantemente. No se puede esperar a que suceda un evento para poder actuar. Aunque el jugador no haga nada, no presione una sola tecla, no mueva el ratón, el juego tiene que calcular el tiempo que lleva jugando en ese nivel, si le va a atacar algún enemigo, si está cambiando de estado y, por supuesto, dibujar en pantalla todo lo que va sucediendo.
La mayoría de videojuegos están construidos bajo la misma estructura básica sobre la cual corre el programa. Esta estructura puede tener variaciones, pero de forma general se presenta de la siguiente manera, figura 5.1:

Inicialización: En esta primera etapa el juego se sitúa en su estado inicial. Se realizan las inicializaciones por lo que respecta a las librerías o motores que vayan a utilizarse, de variables y estructuras de datos referentes a los atributos de las entidades o personajes, escenarios, configuraciones, etc., y de los diferentes recursos físicos que vayan a emplearse, tales como gráficos o sonidos.

Ciclo de juego: En esta parte se desarrolla la lógica de juego, este ciclo se repite de manera indefinida hasta que el jugador pierda, gane o salga del juego, además de otros posibles eventos que dependen de la complejidad del juego en sí. Sin embargo, de forma general este apartado se puede dividir en tres partes: entrada, proceso y salida.

- La entrada o input, captura todo lo que hace el jugador, como presionar los botones de control, mover el ratón, presionar las flechas del teclado, tocar la pantalla táctil y el resto de información que recibe el juego.
- El proceso, es donde se identifica toda la información que se recibió en la entrada y tiene lugar la lógica del videojuego. Por tanto, es donde se realizan los cálculos de físicas si el juego los requiere, así como también los cálculos de la inteligencia artificial de éste, conjuntamente con el tratamiento gráfico.

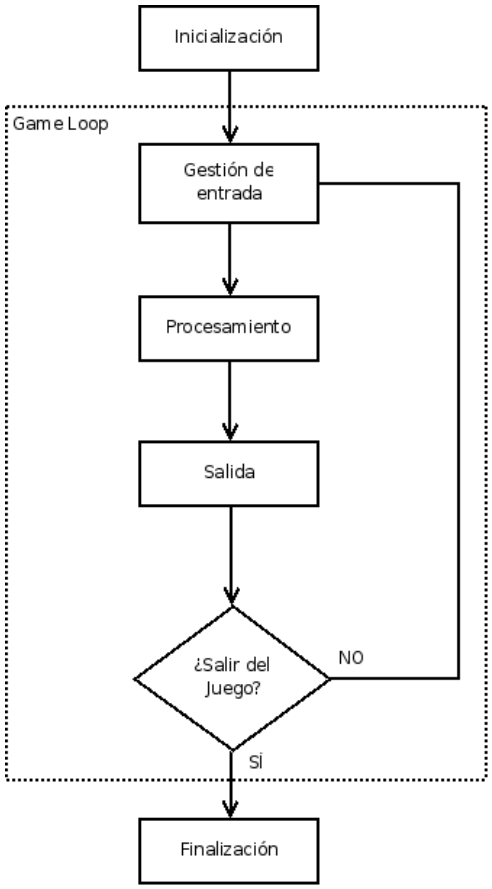

Figura 5.1: Esquema general de videojuego.

- La salida, o output, es la encargada de enviar al jugador toda la información que se ha procesado en la etapa anterior "proceso". Se muestra al jugador el resultado de lo que hizo con la visualización en pantalla del estado actual del videojuego.

Finalización: Por último y fuera del ciclo de juego esta la parte de finalización, esta parte es la encargada de liberar todos los recursos y memoria utilizada durante el transcurso del videojuego. Es importante diferenciar estas tres etapas del ciclo del videojuego y ver qué tarea se delega a cada una de ellas.

5.2. Esquema de la estructura básica del videojuego

Si usamos la estructura básica de un videjuego, para el sistema a desarrollar en la inicialización se cargarán todos los assets y se crearán todos los objetos necesarios para el correcto funcionamiento de un nivel.

En el Game Loop o procesamiento del juego por una parte se colocarían los controles en el mapa y tras el inicio de la fase de recorrido en la salida se podría observar las colisiones de las entradas con los controles y el comportamiento de éstos.

En la finalización se comprobaría si se ha resuelto el nivel de forma correcta o no.

5.3. Patrones de diseño

En esta sección se describirán los principales patrones de diseño de videojuegos utilizados para el desarrollo del sistema.

5.3.1. Introducción

El diseño de aplicaciones es complejo y la etapa más importante del desarrollo y que más impacto tiene, no sólo sobre el producto final, sino también sobre su vida futura.

En el diseño en donde se definen las estructuras y entidades que se van a encargar de resolver el problema planteado. Como de bien se definan estas estructuras y entidades influirá en gran medida , en el éxito o fracaso del proyecto y en la viabilidad de su mantenimiento.

Los patrones de diseño son formas reconocidas y probadas de resolver problemas de diseño que son recurrentes en el tiempo. No es más que basarse en experiencias anteriores en problemas similares aplicado soluciones probadas anteriormente en determinados contextos para alcanzar un mejor diseño más rápidamente.

Existen distintos patrones para distintos problemas de diseño. No obstante, en este documento sólo se hará hincapié en algunos de los patrones utilizados para este proyecto. Quedando fuera del alcance de este documento los demás patrones existentes.

5.3.2. Patrón creador

Estos patrones nos proporcionan una solución relacionada con la construcción de clases, objetos y otras estructuras de datos. Algunos ejemplos de patrones pueden ser Abstract Factory, Builder, Singleton, estos ofrecen mecanismos de creación de instancias de objetos y estructuras escalables dependiendo de las necesidades.

5.3.3. Patrón Singleton

Para este proyecto se ha utilizado este patrón ya que era necesario tener una única instancia de algunas determinadas clases. En java cuando se utiliza el operador new es posible crear una instancia de un objeto. No obstante a veces es necesario tener una única instancia de un objeto, por ejemplo para poder acceder a ella desde cualquier punto de la aplicación.
La solución para garantizar que solo existe una instancia de un objeto se consigue impidiendo que ninguna otra clase pueda acceder al constructor, por este motivo de declara el constructor como mínimo de forma protegida o privada y se proporciona un único punto (controlado) en donde se proporciona la única instancia. Además se define como clase estática "static"para que sea accesible al resto de clases.
En el videjuego, por ejemplo, la clase que controla toda la inicialización de datos sigue el patrón singleton. Una sola instancia que inicializa todos los valores generales del videjuego y que es

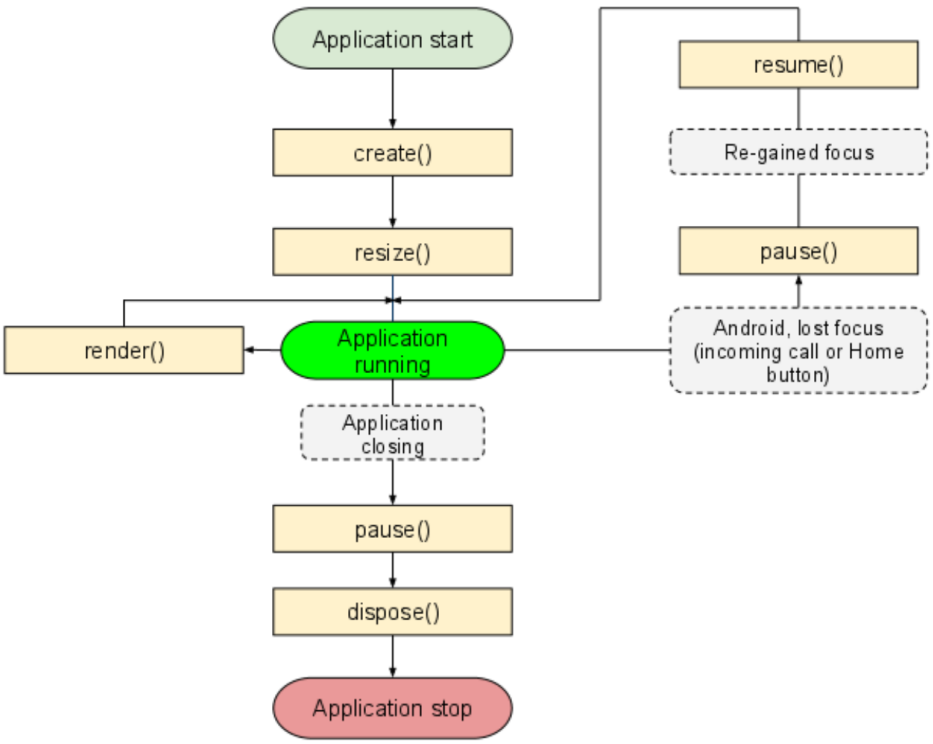

Figura 5.2: Ciclo de vida LibGdx

accesible desde los inicializadores de otras clases para así ofrecer distintas configuraciones de niveles.

5.4. Esquema de diseño LibGdx

En este apartado se explica el ciclo de vida básico que ya ofrece LibGdx por defecto. Durante el desarrollo se ha intentado llevar a cabo este método de diseño para todas las clases creadas

Como podemos ver en la figura 5.2, el ciclo de vida de una aplicación en LibGdx viene de un diseño específico que hay que cumplir para poder desarrollar usando este framework. El ciclo consta de manera general de:

- Create: Se llama una vez cuando se crea la aplicación
- Resize: Se llama cada vez que la pantalla cambia su tamaño
- Render: El bucle del juego y su actualización se realizan en este estado varias veces por segundo
- Pause: Es un estado que se utiliza para parar el bucle de juego ya sea porque termine o porque se salga del flujo principal

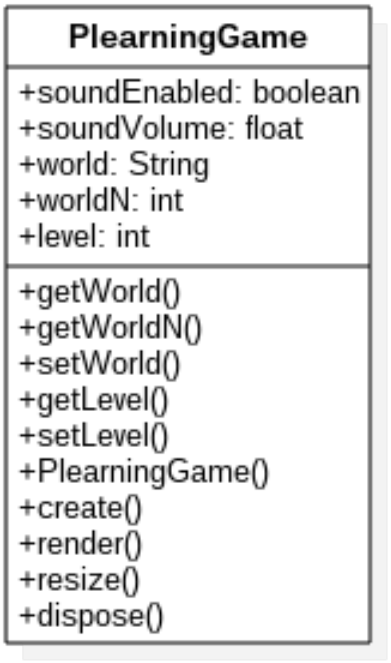

Figura 5.3: clase principal del juego

- Resume: Es un estado para volver de la pausa
- Dispose: Es el estado en el que se destruyen los recursos y se libera memoria.

5.5. Diseño detallado de Componentes

En esta sección se enumenrarán las clases de diseño más importantes y cómo interactúan entre ellas.

Comenzamos por la clase principal, en la figura 5.3 es la clase principal del juego. La primera que es llamada desde el main. Además es usada por la clase de la figura 5.4 en la relación que puede observarse en la figura.
En la figura 5.5, podemos ver la relación de las clases mencionadas.
Usamos el concepto de escenas de Scene2D como ya se mencionó en capítulos anteriores. Por tanto nos hemos creado una escena base y a partir de ésta heredan las escenas del juego como puede verse en la figura 5.6.

En la figura 5.7 tenemos la escena principal del juego la que maneja la lógica de juego. La escena de la figura 5.8, es la que se encarga del menú principal mientras que 5.9 y 5.10, se encargan de la selección de mundo y nivel respectivamente.

La escena de la figura 5.11 se encarga de la pantalla de ayuda.
Las escenas en Scene2D contienen diferentes actores que son los que llevan a cabo las acciones. Por

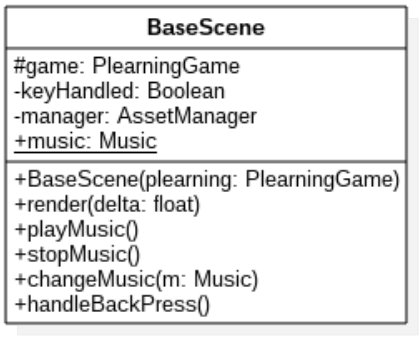

Figura 5.4: Clase de escena base

tanto para la escena principal de juego se han definido diferentes actores como puede observarse en la figura 5.12.

En la figura 5.13 tenemos el actor que se encarga de los botones de opciones. El actor 5.14, es el que define el comportamiento de los controles del juego mientras que 5.15 se encarga de las entradas y salidas representadas en forma de pelotas de colores.

5.6. Diseño detallado de la Interfaz de Usuario

En esta sección se detallarán las interfaces entre el sistema y el usuario, incluyendo un prototipo de alta fidelidad con el diseño de la IU. Se definirá el comportamiento de las diferentes pantallas, indicando qué ocurre en los distintos componentes visuales de la interfaz cuando aparecen y qué acciones se disparan cuando el usuario trabaja con ellas.
Desde la interfaz de menú principal, figura 5.16, será de la que se parta al iniciar el sistema. Dispondrá de cuatro botones que llevarán a otras pantallas.

Botón jugar Lleva a la pantalla de selección de mundo que además permitirá seleccionar un nivel en el que comenzar el juego.

Opciones de sonido Lleva a la pantalla de opciones de sonido que describiremos más adelante.

Ayuda del juego Esta pantalla contendrá un mínimo de información que permita al jugador aprender a jugar desde el primer momento, será como un tutorial del funcionamiento del juego y sus diferentes controles y recursos.

Salir del juego Permite terminar o salir del juego.
En el menú de opciones del juego, figura 5.17, se permitirá manejar algunas de las opciones de sonido para el juego. Así el jugador podrá adaptarlo a su gusto.

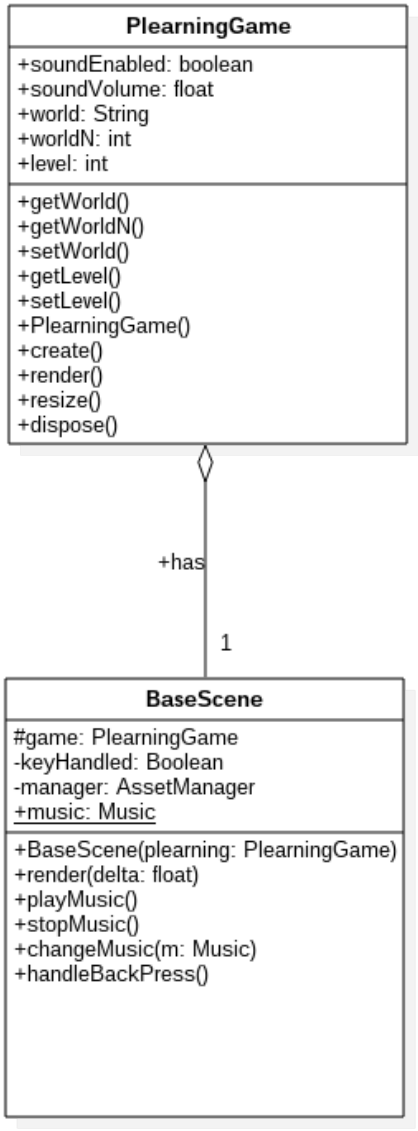

Figura 5.5: Clase de agregación

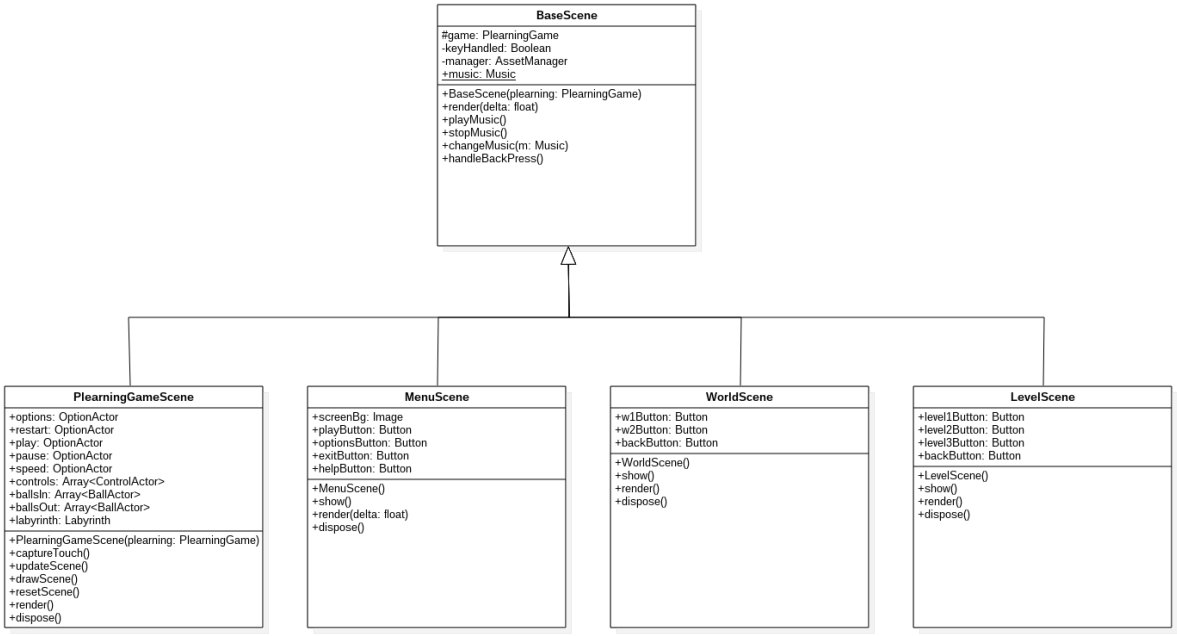

Figura 5.6: Relación de escenas

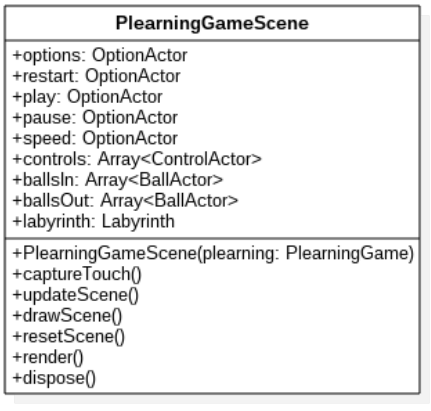

Figura 5.7: Clase de escena de juego

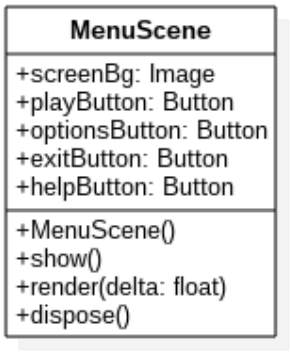

Figura 5.8: Clase de menú principal

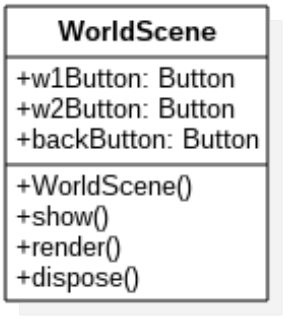

Figura 5.9: Clase de selección de mundos

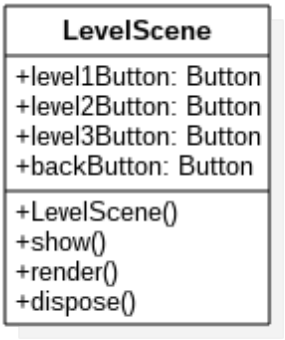

Figura 5.10: Clase de selección de nivel

Figura 5.11: Clase de ayuda

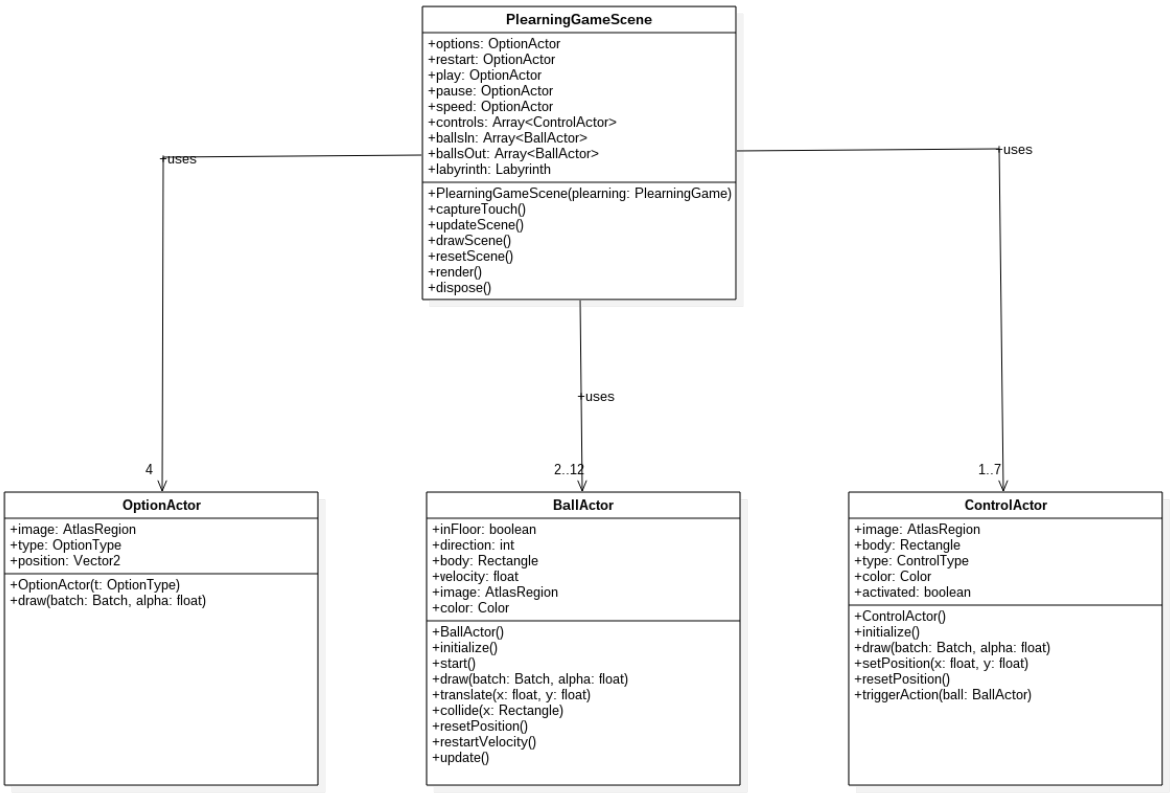

Figura 5.12: Relación de actores

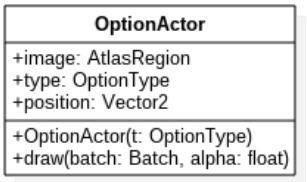

Figura 5.13: Clase de las opciones del juego

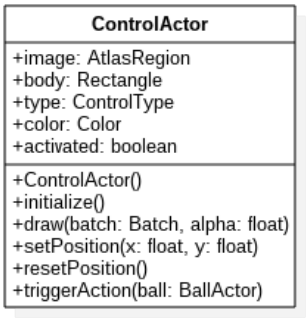

Figura 5.14: Clase de los controles del juego

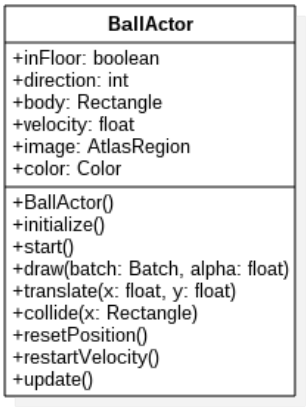

Figura 5.15: Clase de entradas y salidas

Figura 5.16: Concepto menú de juego

Figura 5.17: Concepto opciones de juego

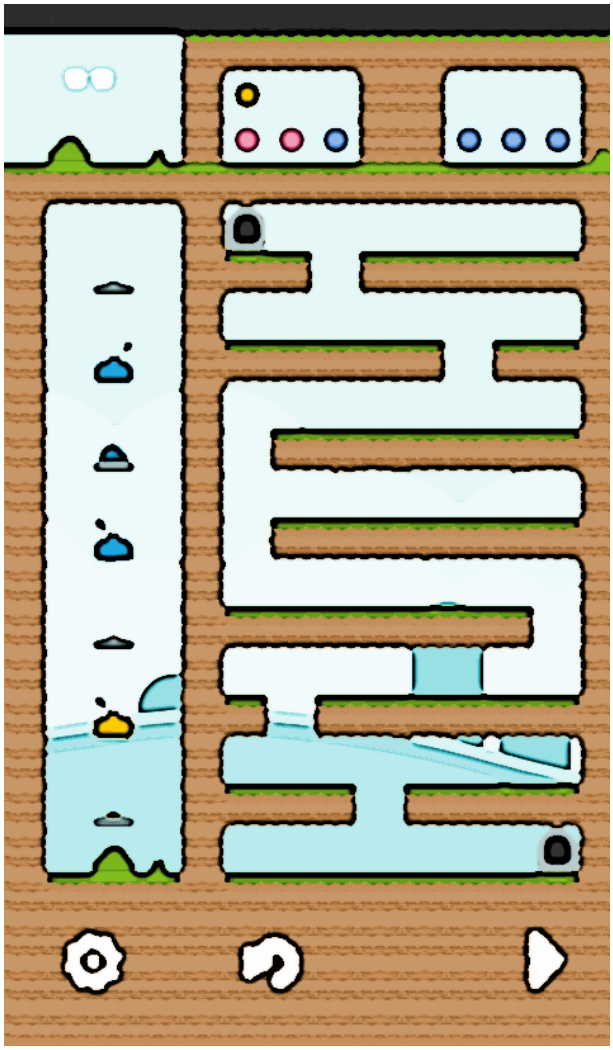

Figura 5.18: Concepto juego

Silencio Silencia todo el sonido y música del juego.

Control de volumen Permite al jugador configurar el volumen que desea para que se escuche en el juego.

Volver Permite volver a la pantalla anterior. Habrá un botón de este tipo en muchas de las pantallas para que la navegabilidad entre todas las pantallas sea completa.
En la pantalla de juego, figura 5.18, se ofrece un esquema de lo las funcionalidades básicas que se han de cubrir para desarrollar la lógica de juego.

Tenemos en la parte de arriba dos cajas con pelotas de colores. La de la izquierda es la entrada y la de la derecha la salida deseada.

Tenemos un temporizador (esquina superior izquierda) con un tiempo dado para colocar los controles (columna de la izquierda) en el laberinto.

Cuando seleccionamos un control podemos colocarlo en el laberinto y cada uno tiene un comportamiento específico al interactuar con una pelota:

- Duplicator – Creador de pelotas: Cuando una pelota entra por el control se genera una pelota nueva de ese mismo color (Un sólo uso).
- Remove(color) – Destructor de pelotas: Cuando una pelota del color especificado entra por el control ésta se destruye.
- Convert – Coloreador de pelotas: Cuando una pelota entra por el control se cambia el color de la pelota al indicado.
- Ifleft, Ifright – Condicional: Cuando una pelota de un color indicado entra por el control se irá a la izquierda/derecha, mientras que si no es del color indicado irá a la otra parte (Un sólo uso).
- Dirchange – Cambio de dirección: Cuando una pelota de un color determinado entra por el control cambiará a la dirección opuesta (solo izquierda y derecha) (Un sólo uso).
- Stop(color) – Retenedor: Cuando una pelota entre en el control, ésta quedará inmóvil un tiempo, transcurrido ese tiempo la pelota continuará su camino (Un sólo uso).

Una vez colocados los controles en el laberinto (parte derecha) podremos pulsar “play” o dejar que acabe el tiempo. Las pelotas de la entrada caerán de la puerta superior izquierda del laberinto, rodarán hacia la derecha hasta chocar con una pared, en ese caso cambiarán de dirección. Cuando caigan por un agujero seguirán con la dirección que tenían antes de caer. Mientras tanto interactuarán con los controles colocados y una vez que las pelotas pasen por la salida (puerta inferior derecha del laberinto) se comprobará si se cumple con la salida deseada. En caso de que sea así se habrá superado el nivel.

Capítulo 6

Construcción del Sistema

Este capítulo trata sobre todos los aspectos relacionados con la implementación del sistema en código, haciendo uso de un determinado entorno tecnológico.

6.1. Entorno de Construcción

En esta sección se debe indicar el marco tecnológico utilizado para la construcción del sistema: entorno de desarrollo (IDE), lenguaje de programación, herramientas de ayuda a la construcción y despliegue, control de versiones, repositorio de componentes, integración contínua, etc.

Antes de empezar a desarrollar juegos con libGDX, es necesario instalar y configurar el entorno de desarrollo. Para ello se utilizará Eclipse como entorno de desarrollo integrado IDE. A su vez se creara un proyecto que nos servirá como base para el desarrollo de este. Es posible desarrollar con otros entornos distintos a Eclipse, pero el hecho que Eclipse sea el entrono de desarrollo soportado oficialmente y la experiencia previa con este IDE, han sido motivos suficientes para escogerlo para este proyecto como entorno de desarrollo.

Los proyectos Libgdx utilizan Gradle para gestionar las dependencias, el proceso de construcción, e integración con el IDE.

Configuración de Eclipse

Para desarrollar una aplicación con Eclipse, se necesitan instalar las siguientes herramientas:

- Java Development Kit 7+ (JDK)
- Eclipse
- Android SDK , únicamente el SDK, no el ADT bundle, que incluye Eclipse. Instalar todas las plataformas vía SDK Manager
- Android Development Tools para Eclipse , o ADT Plugin
- Integración de Eclipse con Gradle

Una vez que están instaladas todas estas herramientas, se puede proceder a crear un proyecto con la aplicación gdx-setup.rar que luego podrá ser importado desde eclipse ya con la estructura, componentes y dependencias necesarias para desarrollar con LibGdx.

6.2. Código Fuente

Organización del código fuente, describiendo la utilidad de los diferentes ficheros y su distribución en paquetes o directorios. Asimismo, se incluirá algún extracto significativo de código fuente que sea de interés para ilustrar algún algoritmo o funcionalidad específica del sistema.

En total para el desarrollo del videojuego se han usado diecinueve clases que se enumerarán a continuación por orden de importancia:

- PlearningGame
- BaseScene
- PlearningGameScene
- MenuScene
- WorldScene
- LevelScene
- Help1Scene
- Help2Scene
- Help3Scene
- BallActor
- ControlActor
- OptionActor
- TimerActor
- DataSingleton
- Labyrinth
- LoadingBar
- LoadingScreen
- LooseDialog
- WinDialog

En el siguiente apartado se añadirán varios fragmentos de código de las clases principales.

La clase principal del juego es PlearningGame y hereda de la clase Game de LibGdx que es una de las principales del framework.

```
public class PlearningGame extends Game {
        public static final int windowHeight =800;
        public static final int windowWidth = 480;

        SpriteBatch batch;
        OrthographicCamera camera;
        Viewport viewport;
        TextureAtlas atlas;

        AssetManager manager;
        public boolean soundEnabled;
        public float soundVolume;

        private String world;
        private int worldN;
        private int level;

        public String getWorld() {
                return world;
        }
        public int getWorldN(){
                return worldN;
        }
        public void setWorld(String world, int n) {
                this.world = world;
                this.worldN = n;
        }
        public int getLevel() {
                return level;
        }
        public void setLevel(int level) {
                this.level = level;
        }
        public PlearningGame(){
                //Camera
                camera = new OrthographicCamera();
                camera.position.set((windowWidth*0.5f), (windowHeight
                   *0.5f), 0);
                viewport = new FitViewport(windowWidth, windowHeight,
                   camera);
                manager = new AssetManager();
                soundEnabled = true;
                soundVolume = 1.5f;
        }
        @Override
        public void create() {
                scaleFactorX = Math.round(windowWidth/14);
                scaleFactorY = Math.round(windowHeight/23);
                batch = new SpriteBatch();
                setScreen(new LoadingScreen(this));

        }
```

```
51          public void render(){
52                  super.render();
53          }
54
55          public void resize(int width, int height){
56                  viewport.update(width, height);
57                  scaleFactorX = Math.round(width/14);
58                  scaleFactorY = Math.round(height/23);
59
60          }
61          public void dispose(){
62                  batch.dispose();
63                  atlas.dispose();
64          }
65  }
```

La siguiente clase es de la que heredan todas las escenas, siguiendo el estilo de diseño de Scene2D y usando o heredando de sus componentes.

```
1   public class BaseScene extends ScreenAdapter {
2           protected PlearningGame game;
3           private boolean keyHandled;
4           AssetManager manager;
5           static Music music;
6
7           public BaseScene(PlearningGame plearning){
8                   game = plearning;
9                   keyHandled = false;
10                  //For music
11                  manager = game.manager;
12                  music = manager.get("SOUNDS/themeSong.mp3", Music.class)
                        ;
13                  music.setLooping(true);
14
15                  Gdx.input.setCatchBackKey(true);
16                  Gdx.input.setCatchMenuKey(true);
17          }
18          public void render(float delta){
19                  super.render(delta);
20                  if(Gdx.input.isKeyPressed(Keys.BACK)){
21                          if(keyHandled){
22                                  return;
23                          }
24                          handleBackPress();
25                          keyHandled = true;
26                  }
27                  else{
28                          keyHandled = false;
29                  }
30          }
31
32          public void playMusic(){
```

```
33                  if(game.soundEnabled){
34                          music.play();
35                  }
36          }
37
38          public void stopMusic(){
39                  if(music!=null){
40                          music.stop();
41                  }
42          }
43
44          public void changeMusic(Music m){
45                  if(game.soundEnabled){
46                          stopMusic();
47                          music = m;
48                          music.setLooping(true);
49                  }
50
51          }
52          protected void handleBackPress(){
53
54          }
55
56  }
```

Esta clase es de las más importantes ya que maneja el escenario principal del videojuego, es decir, alberga toda la lógica de juego y sus diferentes estados.

Declaramos la cabecera y algunas enumeraciones interesantes para controlar los estados de juego y los colores de los elementos.

```
1   public class PlearningGameScene extends BaseScene {
2           ...
3
4           static enum Color{
5                   RED, BLUE, GREEN, YELLOW
6           }
7           static enum GameState{
8                   INIT, CONTROLS, PLAYING, PAUSE, FINISH, OPTIONS, START,
                        WIN, LOOSE
9           }
10
11          GameState gameState = GameState.INIT;
12
13          public PlearningGameScene (PlearningGame plearning) {
14                  ...
```

Aquí se aporta comportamiento a los elementos en los que se puede hacer clicken el juego

```
1                   //Touch Listeners
```

```
2                   restart.addListener(new ClickListener(){
3               @Override
4               public void clicked(InputEvent event, float x, float y) {
5                   resetScene();
6               }
7           });
8                   options.addListener(new ClickListener(){
9               @Override
10              public void clicked(InputEvent event, float x, float y) {
11                  if(game.soundEnabled){
12                                  stopMusic();
13                          }
14                  game.setScreen(new MenuScene(game));
15              }
16          });
17
18                  speed.addListener(new ClickListener(){
19              @Override
20              public void clicked(InputEvent event, float x, float y) {
21                                  if(speeded){
22                                          BallActor.restartVelocity();
23                                          speeded = false;
24                  }
25                                  else{
26                                          BallActor.VELOCITY *= 2;
27                                          speeded = true;
28                                  }
29              }
30          });
31                  play.addListener(new ClickListener(){
32              @Override
33              public void clicked(InputEvent event, float x, float y) {
34                  if(!pauseState){
35                          gameState = GameState.START;
36                          //Sound and Music
37                                  if(game.soundEnabled){
38                                          stopMusic();
39                                          changeMusic(manager.get("SOUNDS/
                                               w1-2Song.mp3", Music.class));
40                                  }
41                          }
42                  else{
43                          pauseState = false;
44                          gameState = GameState.PLAYING;
45                  }
46              }
47          });
48                  pause.addListener(new ClickListener(){
49              @Override
50              public void clicked(InputEvent event, float x, float y) {
51                  if(pauseState){
52                          pauseState = false;
53                          gameState = GameState.PLAYING;
```

```
54                  }
55                  else{
56                          pauseState = true;
57                          gameState = GameState.PAUSE;
58                  }
59              }
60          });
61                  for(final ControlActor c : controls){
62                          c.addListener(new ClickListener(){
63                      @Override
64                      public void clicked(InputEvent event, float x, float
                            y) {
65
66                                              c.playTap();
67                                              controlsPushed[c.index] = true;
68
69                      }
70                  });
71                  }
72
73          ...
74
75          }
```

Continuamos con las actualizaciones en el juego que se ejecutarán en cada renderizado y va alternando según el estado de juego en el que nos encontremos:

```
1           ...
2
3           private void updateScene(){
4                   scaleFactorX = game.scaleFactorX;
5                   scaleFactorY = game.scaleFactorY;
6
7                   if(gameState == GameState.INIT){
8                           if(game.soundEnabled){
9                                   playMusic();
10                                  music.setVolume(game.soundVolume);
11                          }
12
13                          pause.setVisible(false);
14                          speed.setVisible(false);
15                          BallActor.restartVelocity();
16                          gameState = GameState.CONTROLS;
17                  }
18                  else if(gameState == GameState.CONTROLS){
19                          captureTouch();
20
21                          if(TimeUtils.millis() - time > 1000){
22                                  //Esta parte se ejecutara cada segundo.
23                                  time = TimeUtils.millis();
24                                  timeOut -= 1000;
25                          }
```

```
26
27
28                      timer.setText(""+(int)(timeOut*0.001f));
29                      if(timeOut <= 0){
30                              if(game.soundEnabled){
31                              stopMusic();
32                              changeMusic(manager.get("SOUNDS/w1-2Song
                                    .mp3", Music.class));
33                      }
34                              gameState = GameState.START;
35                      }
36              }
37              else if(gameState == GameState.PAUSE){
38                      play.setVisible(true);
39                      speed.setVisible(false);
40
41              }
42              else if(gameState == GameState.START){
43                      for(BallActor ball: ballsIn){
44                              ball.start();
45                      }
46
47                      //ballsIn.get(0).start();
48                      gameState = GameState.PLAYING;
49              }
50              else if(gameState == GameState.PLAYING){
51                      pause.setVisible(true);
52                      play.setVisible(false);
53
54                      if(ballsEnd.size==BallActor.nBallsIn){
55                              gameState = GameState.FINISH;
56                      }
57                      if(game.soundEnabled){
58                              playMusic();
59                              music.setVolume(game.soundVolume);
60                      }
61                      if(timeB > 0){
62                              if(TimeUtils.millis() - time > 1000){
63                                      //Esta parte se ejecutara cada
                                            segundo.
64                                      time = TimeUtils.millis();
65                                      timeB -= 1000;
66
67                              }
68                      }
69                      if(timeB<=0){
70                              speed.setVisible(true);
71                      }
72
73              ...
74
75              else if(gameState == GameState.FINISH){
76
```

```
77                          if(game.soundEnabled){
78                                  stopMusic();
79                          }
80
81                          if(ballsEnd.size == ballsOut.size){
82                                  win = (ballsEnd.get(0).type == ballsOut.
                                       get(0).type);
83                                  for(int i=1; i<ballsEnd.size; i++){
84                                          win = win && (ballsEnd.get(i).
                                               type == ballsOut.get(i).type)
                                               ;
85                                  }
86                          }
87
88                          if(win){
89                                  gameState = GameState.WIN;
90
91                          }
92                          else{
93                                  gameState = GameState.LOOSE;
94                          }
95                  }
96                  else if(gameState == GameState.WIN){
97                          winDialog.show(stage);
98                  }
99                  else if(gameState == GameState.LOOSE){
100                         looseDialog.show(stage);
101                         if(looseDialog.retry){
102                                 resetScene();
103                         }
104                 }
105
106         }
```

En este fragmento veremos métodos auxiliares para poder controlar la matriz de posiciones de los elementos además de los métodos que exigen el ciclo de vida en LibGdx.

```
1           private int gridValueX(float value){
2                   return (int) Math.floor(value/35);
3           }
4           private int ungridValueX(float value){
5                   return gridValueX(value)*35;
6           }
7
8           private int gridValueY(float value){
9                   return (int) Math.floor(value/35);
10          }
11          private int ungridValueY(float value){
12                  return gridValueY(value)*35;
13          }
14
15          private void drawScene(){
```

```
                camera.update();
                batch.setProjectionMatrix(camera.combined);

                batch.begin();

                batch.disableBlending();
                batch.draw(background, 0, 0);
                batch.enableBlending();

                batch.end();


                labyrinth.draw();
        }
        private void resetScene(){
                if(game.soundEnabled){
                        stopMusic();
                }
                BallActor.restartVelocity();
                game.setScreen(new PlearningGameScene(game));
        }

        @Override
        public void render (float delta) {
                Gdx.gl.glClearColor(0, 0, 0, 1);
                Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

                updateScene();
                drawScene();
                stage.draw();
        }
        protected void handleBackPress() {
                //-- depurar
                System.out.println("back");
                if(game.soundEnabled){
                        stopMusic();
                }
                game.setScreen(new MenuScene(game));
        }

        public void dispose(){
                tapSound.dispose();
                music.dispose();
                stage.dispose();
                skin.dispose();
        }

}
```

Para más información se puede consultar el código en los anexos digitales en el directorio:

```
p-learning/codigoFuente/PlearningGame/core/src/com/plearning/game
```

6.3. Scripts de Base de datos

No se ha desarrollado una base de datos para el sistema. Sin embargo en el siguiente fragmento se puede observar el manejo de datos persistentes e iniciales de los que se nutrirá la aplicación.

Para más información se puede consultar el código en los anexos digitales en el directorio:

```
p-learning/codigoFuente/PlearningGame/core/src/com/plearning/game
```

```
1
2  public class DataSingleton {
3          PlearningGame game;
4          //Managing controls
5          int[] nControls;
6          //BooleanArray controlsPushed;
7          Array<Array<ControlActor>> controls;
8          //Vector<ControlActor> controls;
9
10         //Managin balls
11                 //Timer for all balls
12         long timeB;
13         long timeBMax;
14
15         Array<Array<BallActor>> ballsIn;
16         Array<Array<BallActor>> ballsOut;
17         int level;
18
19         public DataSingleton(PlearningGame plearning){
20                 game = plearning;
21
22                 //static inicializations
23                 ControlActor.initialize();
24                 BallActor.initialize();
25                 //Level 0 ----------------------
26                 level = 0;
27                 //Here we choose the controls
28                 /* watch out here --> int nControls = 7; int level = 1*/
29                 nControls = new int[6];
30
31                 controls = new Array<Array<ControlActor>>();
32                 controls.size = 6;
33                 for(int i=0; i<6;i++){
34                         controls.set(i, new Array<ControlActor>());
35                 }
36                 nControls[level] = 7;
37
38                 controls.get(level).add(new ControlActor(game,
                       ControlActor.controlType.DESTRUCTOR, Color.YELLOW, 0)
                       );
39                 controls.get(level).add(new ControlActor(game,
                       ControlActor.controlType.IFLEFT, Color.YELLOW, 1));
40                 controls.get(level).add(new ControlActor(game,
```

```
                    ControlActor.controlType.DESTRUCTOR, Color.RED, 2));
41                  controls.get(level).add(new ControlActor(game,
                    ControlActor.controlType.IFLEFT, Color.BLUE, 3));
42                  controls.get(level).add(new ControlActor(game,
                    ControlActor.controlType.CONVERTER, Color.BLUE, 4));
43                  controls.get(level).add(new ControlActor(game,
                    ControlActor.controlType.IFRIGHT, Color.BLUE, 5));
44                  controls.get(level).add(new ControlActor(game,
                    ControlActor.controlType.DESTRUCTOR, Color.BLUE, 6));
45
46                  //Here we choose the balls in and out
47                  ballsIn = new Array<Array<BallActor>>();
48                  ballsOut = new Array<Array<BallActor>>();
49
50
51                          /*      Care with this
52                           *              int nBallsIn = 4;
53                           *              int nBallsOut = 3;
54                           *
55                          */
56                  timeBMax = 4 * 2 * 1000;
57                  timeB = timeBMax;
58                  ballsIn.size = 4;
59                  for(int i=0; i<4;i++){
60                          ballsIn.set(i, new Array<BallActor>());
61                  }
62
63                  ballsIn.get(level).add(new BallActor(game, Color.BLUE,
                    BallActor.BallInOut.IN, 0, timeBMax));
64                  ballsIn.get(level).add(new BallActor(game, Color.RED,
                    BallActor.BallInOut.IN, 1, timeBMax-2000));
65                  ballsIn.get(level).add(new BallActor(game, Color.RED,
                    BallActor.BallInOut.IN,2, timeBMax-4000));
66                  ballsIn.get(level).add(new BallActor(game, Color.YELLOW,
                     BallActor.BallInOut.IN, 3, timeBMax -6000));
67
68                  ballsOut.add(new Array<BallActor>());
69                  ballsOut.get(level).add(new BallActor(game, Color.BLUE,
                    BallActor.BallInOut.OUT, 0, 0));
70                  ballsOut.get(level).add(new BallActor(game, Color.BLUE,
                    BallActor.BallInOut.OUT, 1, 0));
71                  ballsOut.get(level).add(new BallActor(game, Color.BLUE,
                    BallActor.BallInOut.OUT, 2, 0));
72
73                  //Level 1 ----------------------
74                  level = 1;
75                  //Here we choose the controls
76                  /* watch out here --> int nControls = 7; int level = 0*/
77                  nControls[level] = 5;
78
79                  controls.get(level).add(new ControlActor(game,
                    ControlActor.controlType.DESTRUCTOR, Color.YELLOW, 0)
                    );
```

```
80                  controls.get(level).add(new ControlActor(game,
                       ControlActor.controlType.IFLEFT, Color.YELLOW, 1));
81                  controls.get(level).add(new ControlActor(game,
                       ControlActor.controlType.DESTRUCTOR, Color.RED, 2));
82                  controls.get(level).add(new ControlActor(game,
                       ControlActor.controlType.IFLEFT, Color.BLUE, 3));
83                  controls.get(level).add(new ControlActor(game,
                       ControlActor.controlType.CONVERTER, Color.BLUE, 4));
84
85
86                  //Here we choose the balls in and out
87
88
89                          /*      Care with this
90                           *              int nBallsIn = 5;
91                           *              int nBallsOut = 3;
92                           *
93                          */
94                  timeBMax = 5 * 2 * 1000;
95                  timeB = timeBMax;
96                  ballsIn.add(new Array<BallActor>());
97                  ballsIn.get(level).add(new BallActor(game, Color.GREEN,
                       BallActor.BallInOut.IN, 0, timeBMax));
98                  ballsIn.get(level).add(new BallActor(game, Color.RED,
                       BallActor.BallInOut.IN, 1, timeBMax-2000));
99                  ballsIn.get(level).add(new BallActor(game, Color.BLUE,
                       BallActor.BallInOut.IN,2, timeBMax-4000));
100                 ballsIn.get(level).add(new BallActor(game, Color.YELLOW,
                        BallActor.BallInOut.IN, 3, timeBMax -6000));
101                 ballsIn.get(level).add(new BallActor(game, Color.GREEN,
                       BallActor.BallInOut.IN, 4, timeBMax -8000));
102
103                 ballsOut.add(new Array<BallActor>());
104                 ballsOut.get(level).add(new BallActor(game, Color.BLUE,
                       BallActor.BallInOut.OUT, 0, 0));
105                 ballsOut.get(level).add(new BallActor(game, Color.GREEN,
                        BallActor.BallInOut.OUT, 1, 0));
106                 ballsOut.get(level).add(new BallActor(game, Color.RED,
                       BallActor.BallInOut.OUT, 2, 0));
107
108                 . . .
109
110                 //Level 5 ----------------------
111                 level = 5;
112                 //Here we choose the controls
113                 /* watch out here --> int nControls = 6; int level = 5*/
114                 nControls[level] = 6;
115
116                 controls.get(level).add(new ControlActor(game,
                       ControlActor.controlType.DESTRUCTOR, Color.YELLOW, 0)
                       );
117                 controls.get(level).add(new ControlActor(game,
                       ControlActor.controlType.IFLEFT, Color.RED, 1));
```

```
118                     controls.get(level).add(new ControlActor(game,
                           ControlActor.controlType.DESTRUCTOR, Color.RED, 2));
119                     controls.get(level).add(new ControlActor(game,
                           ControlActor.controlType.IFLEFT, Color.BLUE, 3));
120                     controls.get(level).add(new ControlActor(game,
                           ControlActor.controlType.CONVERTER, Color.GREEN, 4));
121                     controls.get(level).add(new ControlActor(game,
                           ControlActor.controlType.DIRCHANGER, Color.BLUE, 5));
122
123                     //Here we choose the balls in and out
124
125
126                             /*      Care with this
127                              *              int nBallsIn = 4;
128                              *              int nBallsOut = 3;
129                              *
130                             */
131                     timeBMax = 4 * 2 * 1000;
132                     timeB = timeBMax;
133                     ballsIn.add(new Array<BallActor>());
134                     ballsIn.get(level).add(new BallActor(game, Color.GREEN,
                           BallActor.BallInOut.IN, 0, timeBMax));
135                     ballsIn.get(level).add(new BallActor(game, Color.RED,
                           BallActor.BallInOut.IN, 1, timeBMax-2000));
136                     ballsIn.get(level).add(new BallActor(game, Color.BLUE,
                           BallActor.BallInOut.IN,2, timeBMax-4000));
137                     ballsIn.get(level).add(new BallActor(game, Color.YELLOW,
                            BallActor.BallInOut.IN, 3, timeBMax -6000));
138
139                     ballsOut.add(new Array<BallActor>());
140                     ballsOut.get(level).add(new BallActor(game, Color.GREEN,
                            BallActor.BallInOut.OUT, 0, 0));
141                     ballsOut.get(level).add(new BallActor(game, Color.GREEN,
                            BallActor.BallInOut.OUT, 1, 0));
142                     ballsOut.get(level).add(new BallActor(game, Color.RED,
                           BallActor.BallInOut.OUT, 2, 0));
143             }
144             public Array<ControlActor> getControls(int level){
145                     return controls.get(level);
146             }
147             public Array<BallActor> getBallsIn(int level){
148                     return ballsIn.get(level);
149             }
150             public Array<BallActor> getBallsOut(int level){
151                     return ballsOut.get(level);
152             }
153             public int countBallsIn(int level){
154                     return ballsIn.get(level).size;
155             }
156             public int countBallsOut(int level) {
157                     return ballsOut.get(level).size;
158             }
159             public int countControls(int level) {
```

```
160                     return nControls[level];
161             }
162     }
```

Capítulo 7

Pruebas del Sistema

En este capítulo se presenta el plan de pruebas del sistema de información, incluyendo los diferentes tipos de pruebas que se han llevado a cabo, ya sean manuales (mediante listas de comprobación) o automatizadas mediante algún software específico de pruebas.

7.1. Estrategia

En esta sección se incluye el alcance de las pruebas, hasta donde se pretende llegar con ellas, si se registrarán todas o sólo aquellas de un cierto tipo y cómo se interpretarán y evaluarán los resultados.

La mayor parte de las pruebas efectuadas al sistema se han realizado con checklists de objetivos. Aunque también se incluyen pruebas unitarias y de integración a menor escala.

Además de esto se ha realizado pruebas a posibles roles de usuarios, que como ya se dijo en capítulos anteriores, son niños de entre 8 y 12 años.

Se ha querido hacer pruebas también con usuarios de mayor edad para comprobar si el producto resulta atractivo para ese perfil.

En las siguientes secciones se detallarán los procedimientos llevados a cabo.

7.2. Entorno de Pruebas

Para realizar las pruebas se han usado diferentes dispositivos y entornos.

Para las pruebas en el desarrollo del sistema se ha utilizado un equipo con las siguientes especificaciones:

- Procesador: Intel i5 3.2Ghz con cuatro núcleos
- RAM: 12GB de memoria RAM DDR3
- Sistema Operativo: Ubuntu 14.04

- IDE: Eclipse Mars

También se ha usado un equipo portátil con las siguientes especificaciones:

- Procesador: Intel i3 2.6Ghz con tres núcleos
- RAM: 4GB de memoria RAM DDR3
- Sistema Operativo: Ubuntu 14.04
- IDE: Eclipse Mars

Para las pruebas con smartphone se ha usado un teléfono BQ Aquaris E5 HD

7.3. Roles

En esta sección se describen cuáles serán los perfiles y participantes necesarios para la ejecución de cada uno de los niveles de prueba.

Para las pruebas en la programación, pruebas unitarias y de integración, el perfil lo ha cubierto el desarrollador.

Para las pruebas de aceptación se han usado cuatro perfiles:

- Niña de 9 años.
- Niño de 11 años.
- Adolescente de 16 años
- Hombre de 25 años

Estos perfiles los he usado para comprobar que tanto el arte como la lógica del videojuego fuera adecuado y vistoso, mientras que los niveles de dificultad no fuesen ni demasiado altos ni demasiado bajos.

Con ésto se ha conseguido balancear los niveles de dificultad hasta un punto bastante óptimo, aunque insuficiente en estos niveles del desarrollo.

7.4. Niveles de Pruebas

En este sección se documentan los diferentes tipos de pruebas que se han llevado a cabo, ya sean manuales o automatizadas mediante algún software específico de pruebas.

7.4.1. Pruebas Unitarias

Las pruebas unitarias tienen por objetivo localizar errores en cada nuevo artefacto software desarrollado, antes que se produzca la integración con el resto de artefactos del sistema.

Las pruebas unitarias han consistido en una comprobación en cada clase de su consistencia aún con las dependencias oportunas. Se ha creado un código que crea varios objetos de cada clase desarrollada y se le hacen ciertas pruebas de integridad controladas, buscando que no se produzcan incongruencias ni fallos de ejecución.

Sin embargo este código se ha ido eliminando paulatinamente ya que era un mero control. La estrategia adecuada hubiese sido mantener este código mejor organizado ya sea con la herramienta JUnit o con un paquete nuevo.

Se tendrá en cuenta estos criterios para el próximo desarrollo.

7.4.2. Pruebas de Integración

Este tipo de pruebas tienen por objetivo localizar errores en módulos o subsistemas completos, analizando la interacción entre varios artefactos software.

Una vez que tenía varias clases interdependientes correctas analizaba sus interacciones y mediante el Debbuger de Java comprobaba los estados de variables y funciones para que fuesen correctos.

7.4.3. Pruebas de Sistema

En esta actividad se realizan las pruebas de sistema de modo que se asegure que el sistema cumple con todos los requisitos establecidos.

Estas pruebas se realizaron mediante la ejecución manual del videojuego.
Para ésto cree un checklist en el que se comprobaba cuales eran los resultados deseados con los que realmente se podían observar de la ejecución.

El checklist consiste en:

- La navegabilidad entre pantallas es correcta y bidireccional. Es decir, se puede llegar a cualquier pantalla desde cualquier otra.
- La configuración de opciones afecta a todas las pantallas. Sobre todo las de sonido.
- Al navegar entre los mundos y niveles todos ellos tienen la configuración inicial esperada.
- Cada nivel es resoluble al menos con una posible solución.
- Los cambios de estado en el nivel son correctos y responden a su programación
- Los controles se pueden colocar únicamente en las posiciones válidas del laberinto o en su caja inicial.
- Al acabar el tiempo de colocación de controles las entradas entran en el laberinto.

- Al pulsar el PLAY las entradas entran en el laberinto.
- Los controles sólo interactúan con las entradas con el mismo color en su caso.
- Las opciones del nivel realizan sus funciones correctamente.
- Cuando hay una solución al nivel correcta se muestra el mensaje de que se ha ganado.
- Cuando hay una solución al nivel incorrecta se muestra el mensaje de que se ha perdido.

7.4.4. Pruebas de Aceptación

El objetivo de estas pruebas es demostrar que el producto está listo para el paso a producción. Suelen ser las mismas pruebas que se realizaron anteriormente pero en el entorno de producción. En estas pruebas, es importante la participación del cliente final.

En este proceso de pruebas se ha dejado a los participantes directamente con la aplicación para ver si realmente es lo suficiente intuitiva o sencilla de usar, si la ayuda es funcional, si el juego es divertido y si realmente se usan habilidades lógicas para la resolución de problemas.

El método consiste en dejar a los usuarios con la aplicación, calcular los tiempos que tardan con cada nivel y al final realizar un pequeño cuestionario para saber las dificultades con las que se econtraban y lo que más les había gustado.

El cuestionario tiene el siguiente formato:

Este apartado lo rellena el entrevistador:

- ¿La navegabilidad del menú principal es ágil?
- ¿Ha sido necesario explicar el funcionamiento de los controles?
- ¿Ha sido necesario explicar el comportamiento de entradas y salidas?
- ¿Ha sido necesario explicar la colocación de controles?
- Tiempo que tarda el usuario en terminar el nivel 1:
- Tiempo que tarda el usuario en terminar el nivel 2:
- Tiempo que tarda el usuario en terminar el nivel 3:
- Tiempo que tarda el usuario en terminar el nivel 4:
- Tiempo que tarda el usuario en terminar el nivel 5:
- Tiempo que tarda el usuario en terminar el nivel 6:

Este apartado lo rellena el usuario:

- ¿La navegabilidad del menú principal es ágil?

- ¿Con la ayuda has comprendido el funcionamiento de los controles?
- ¿Con la ayuda has comprendido el comportamiento de entradas y salidas?
- ¿Te parece sencilla la colocación de controles?
- Evalúa de Fácil/Medio/Difícil la dificultad del nivel 1:
- Evalúa de Fácil/Medio/Difícil la dificultad del nivel 2:
- Evalúa de Fácil/Medio/Difícil la dificultad del nivel 3:
- Evalúa de Fácil/Medio/Difícil la dificultad del nivel 4:
- Evalúa de Fácil/Medio/Difícil la dificultad del nivel 5:
- Evalúa de Fácil/Medio/Difícil la dificultad del nivel 6:

7.4.5. Conclusiones

Las pruebas de aceptación han sido definitivas a la hora de realizar ciertos cambios a la lógica de juego y la dificultad de niveles del videojuego. Fue necesario que en la ayuda se explicase mejor la forma de jugar y además se retocó las opciones en la pantalla de juego para que se pudiese aceleran los recorridos de las entradas en el laberinto.

Gracias a los cuestionarios he podido balancear mejor la dificultad de los niveles llegando a la conclusión de que era mejor aumentar la dificultad de forma progresiva y con la introducción de nuevos controles en cada mundo diferente.
Tendré en cuenta estos cuestionarios para las mejoras futuras del videojuego y sería interesante realizar las pruebas con una muestra mayor de usuarios y más heterogénea.

Parte III

Epílogo

En esta última parte quedarán recogidas las conclusiones y los manuales necesarios para el manejo de la aplicación resultado del desarrollo.

Capítulo 8

Manual de implantación y explotación

Las instrucciones de instalación y explotación del sistema se detallan a continuación.

8.1. Introducción

Resumen de los principales objetivos, ámbito y alcance del software desarrollado.

P-Learning Game es un videojuego educativo multiplataforma pensado para desarrollar las destrezas en lógica de niños y niñas de entre 8 y 12 años para así hacer que paulatinamente pueda resultarles más fácil aprender a programar en un lenguaje de programación secuencial y estructurado.

Nace de la idea de que se pueden demostrar mejoras en resultados de aprendizaje mediante el juego y la gamificación.

La gamificación se puede definir como el empleo de mecánicas de juego en entornos y aplicaciones no lúdicas con el fin de potenciar la motivación, la concentración, el esfuerzo, la fidelización y otros valores positivos comunes a todos los juegos.

Pero además de ésto, es obvio que, para aprender a programar hay que resolver muchos problemas ya que la repetición es también un gran método de aprendizaje.

Uniendo ambos conceptos se concibe P-Learning Game, un juego de puzzles y lógica por niveles en el que se parte de una entrada y se pide una salida determinada, además tenemos diversas representaciones de estructuras de control que interactúan con las entradas y las manipulan para llegar a obtener esta salida.

8.2. Requisitos previos

Requisitos hardware y software para la correcta instalación del sistema.

El sistema se despliega en cada cliente, por lo que en esta versión no es necesario un servidor ni un hardware específico más que el entorno de cliente.

Para este caso, el videjouego se ha desarrollado de forma que pueda ejecutarse en diferentes plataformas:

- PC, con sistema operativo Ubuntu, Mac o Windows
- Smartphone, con sistema operativo Android
- Web

Sí es cierto que para un correcto funcionamiento la máquina cliente debe cumplir al menos los siguientes requisitos mínimos de hardware:

- Procesador: mononúcleo a 1GHz de velocidad
- Memoria RAM: 1GB
- Memoria Secundaria: al menos 30MB

8.3. Inventario de componentes

Se incluyen los siguientes componentes software siendo éstos versiones del mismo para poder ejecutarse en diferentes plataformas: sistemas Android, Sistemas de escritorio con Java (Windows, Linux o Mac) o web.

- plearning.jar - Es una aplicación java que puede ser ejecutado siempre y cuando el cliente tenga al menos las versión 1.7 del JDK de Oracle o bien OpenJDK.
- plearning.apk - Es una aplicación android instalable en sistemas operativos Android desde la versión 4 en adelante.

8.4. Procedimientos de instalación

Para el despliegue en un entorno de escritorio partimos de la ventaja de que el ejecutable es un .jar, por tanto será necesario tener instalado Java en su versión 7 o una alternativa libre como OpenJDK. Para ejecutarlo simplemente hacer doble click o en una terminal escribir:

```
$ java -jar plearning
```

Para el despliegue en un smartphone Android debemos instalar directamente la apk. Sin embargo al no estar firmada aún ya que es necesaria una cuenta de desarrollador para google play, se deben activar en las opciones del teléfono el modo desarrollador y seleccionar las opciones para aplicaciones no firmadas y modo debug.

Una vez instalada en el smartphone o tablet tan sólo es necesario iniciar la aplicación desde el menú principal.

Se puede encontrar más información sobre estos procedimientos en los libros:
[Cejas Sánchez and Saltares Márquez, 2014] o en [Bose, 2014].

8.5. Pruebas de implantación

Una vez que la aplicación está instalada en el sistema existe un procedimiento de pruebas para comprobar que todo está correctamente configurado:

- Iniciar la aplicación. La aplicación debe iniciar correctamente mostrando la pantalla de menú principal.
- Configuración de sonido. Se debe comprobar que si se realizan cambios en la configuración de sonido éstos funcionan correctamente.
- Probar la ayuda. Se debe probar que la ayuda se visualice correctamente y que además sea navegable.
- Iniciar un nivel cualquiera. Se debe comprobar que las pantallas de selección de mundo y nivel son navegables y que se iniciar el nivel sin problemas.
- Probar que se pueden colocar los controles en el laberinto. Los controles sólo deberán poder colocarse o en el laberinto o en la cajetilla de inicio.
- Pulsar Play. Se debe comprobar que los recorridos de las entradas se realizan sin problemas, que estas interactúan con los controles y que además se puede pausar y reiniciar el nivel sin problemas.

Si todo este procedimientos se ejecuta sin problemas la aplicación estará correctamente instalada. Si no es así se debería reinstalar.

Capítulo 9

Manual de usuario

Las instrucciones de uso del sistema se detallan a continuación.

9.1. Introducción

Resumen de los principales objetivos, ámbito y alcance del software desarrollado.

P-Learning Game es un videojuego educativo multiplataforma pensado para desarrollar las destrezas en lógica de niños y niñas de entre 8 y 12 años para así hacer que paulatinamente pueda resultarles más fácil aprender a programar en un lenguaje de programación secuencial y estructurado.

Nace de la idea de que se pueden demostrar mejoras en resultados de aprendizaje mediante el juego y la gamificación.

La gamificación se puede definir como el empleo de mecánicas de juego en entornos y aplicaciones no lúdicas con el fin de potenciar la motivación, la concentración, el esfuerzo, la fidelización y otros valores positivos comunes a todos los juegos.

Pero además de ésto, es obvio que, para aprender a programar hay que resolver muchos problemas ya que la repetición es también un gran método de aprendizaje.

Uniendo ambos conceptos se concibe P-Learning Game, un juego de puzzles y lógica por niveles en el que se parte de una entrada y se pide una salida determinada, además tenemos diversas representaciones de estructuras de control que interactúan con las entradas y las manipulan para llegar a obtener esta salida.

Este videojuego es del género puzzle y educatvo. Por tanto su propósito principal, a parte de entretener como todo juego, es la mejora de habilidades del jugador. En este caso la mejora en las destrezas en lógica matemática que conllevan a una mejora en el aprendizaje de la programación.

El objetivo principal es superar los diferentes niveles del juego para así conseguir estas mejoras.

Los niveles están contenidos en diferentes mundos que cambian la apariencia visual e introducen nuevos elementos al jugador para aumentar el nivel de desafío de éstos.

9.2. Características

Recopilación de las principales funcionalidades del sistema.

Las funcionalidades están distribuidas en las siguientes secciones:

- Menú principal
- Opciones
- Selección de mundo
- Selección de nivel
- Nivel de juego

En el menú principal el jugador podrá navegar entre diferentes submenús. Como se muestra en la figura 9.1. Se puede empezar directamente el juego lo que llevará a la pantalla de selección de mundos.

Se puede navegar a las opciones de sonido.

Se puede acudir a una ayuda que explicará a grandes rasgos la forma de jugar al juego.

También se puede salir directamente del juego.

En las opciones de sonido se puede determinar tanto el volumen del sonido como poner el juego en silencio. Figura 9.2.

En la pantalla de selección de mundos se podrá escoger entre varios mundos con distintas cualidades gráficas y además con la introducción de nuevos controles de juego. Figura 9.4.

En la pantalla de selección de nivel se podrá escoger entre varios niveles pertenecientes a un mundo seleccionado previamente. Tendrán distintos controles y configuraciones de entradas y salidas, por tanto tendrán diferentes soluciones. Figura 9.3.

9.3. Requisitos previos

Requisitos hardware y software para el correcto uso del sistema.

Para este caso, el videjouego se ha desarrollado de forma que pueda ejecutarse en diferentes plataformas:

- PC, con sistema operativo Ubuntu, Mac o Windows ya que se ejecuta una aplicación con Java

Figura 9.1: Menú principal del juego

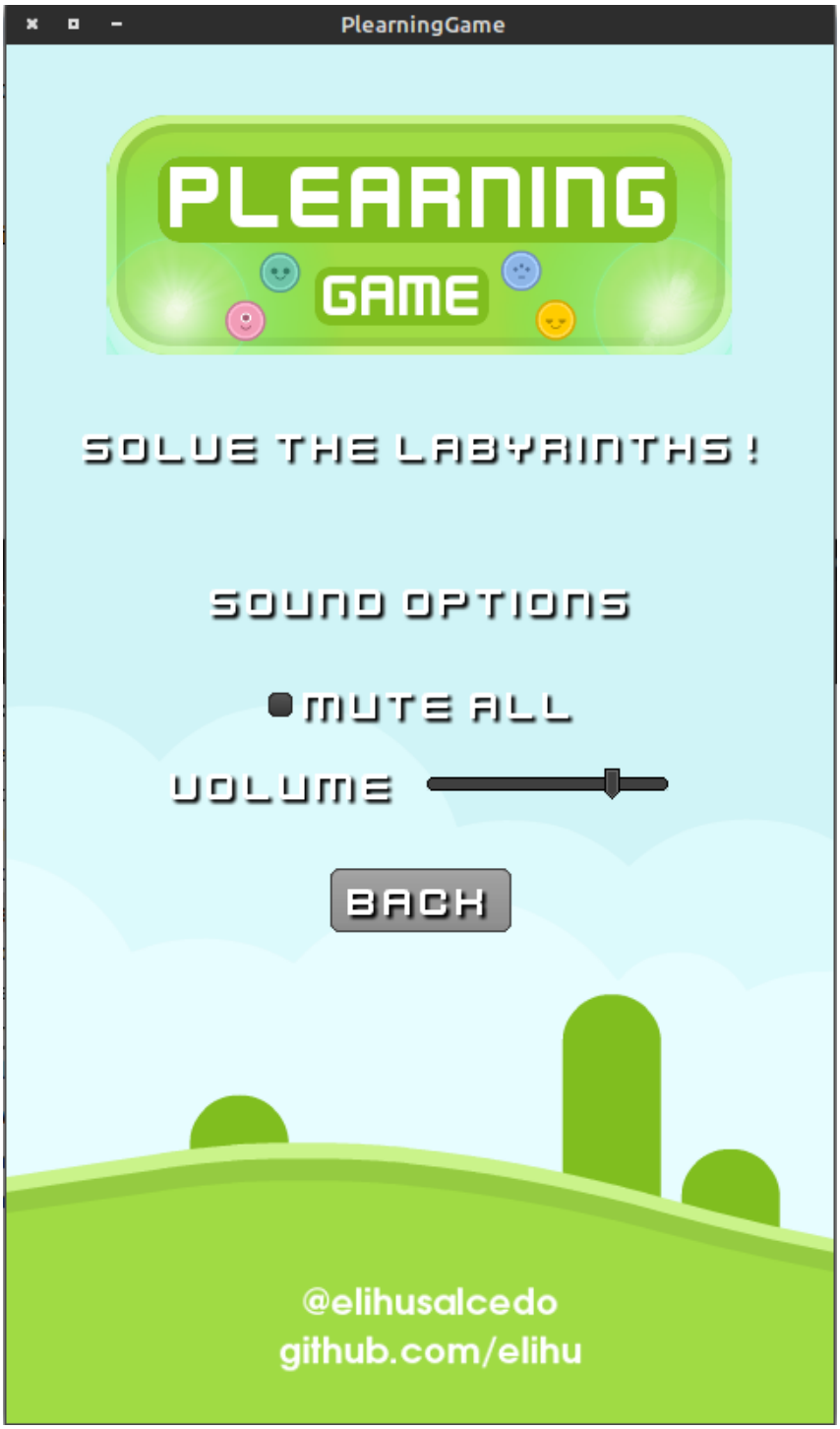

Figura 9.2: Opciones de sonido del juego

Figura 9.3: Pantalla de selección de mundos del juego

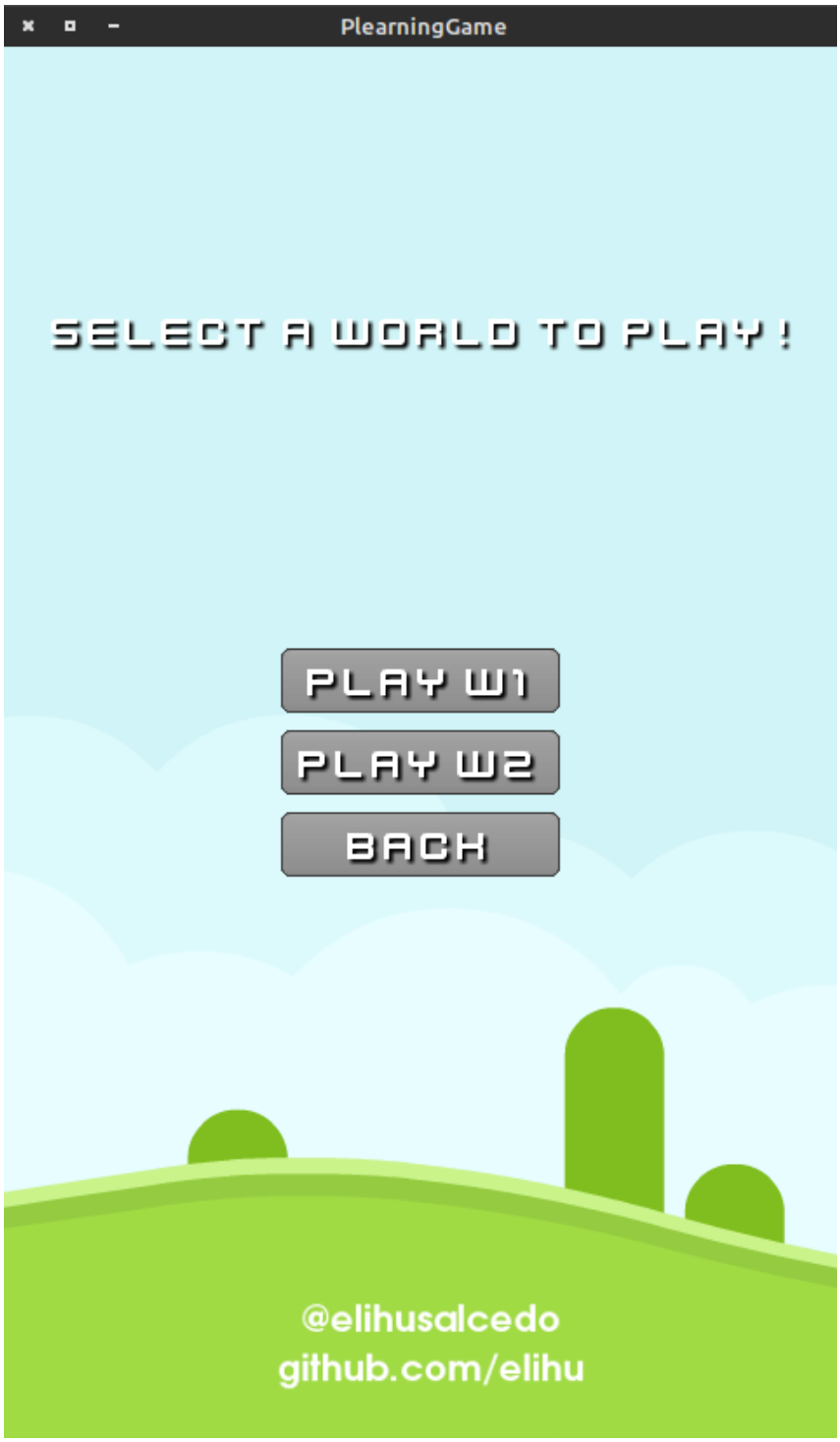

Figura 9.4: Pantalla de selección de mundos del juego

- Smartphone o tablet, con sistema operativo Android desde la versión 4 en adelante.
- Web

Sí es cierto que para un correcto funcionamiento la máquina cliente debe cumplir al menos los siguientes requisitos mínimos de hardware:

- Procesador: mononúcleo a 1GHz de velocidad
- Memoria RAM: 1GB
- Memoria Secundaria: al menos 30MB

9.4. Uso del sistema

Describir todos los aspectos necesarios para una utilización efectiva y eficiente del sistema por parte de los usuarios.

9.4.1. Nivel de juego

Tomaremos como referencia la figura 9.5.
Tenemos en la parte de arriba dos cajas con pelotas de colores. La de la izquierda es la entrada y la de la derecha la salida deseada.

Tenemos un temporizador (esquina superior izquierda) con un tiempo dado para colocar los controles (columna de la izquierda) en el laberinto.

Cuando seleccionamos un control podemos colocarlo en el laberinto y cada uno tiene un comportamiento específico al interactuar con una pelota:

- Duplicator – Creador de pelotas: Cuando una pelota entra por el control se genera una pelota nueva de ese mismo color (Un sólo uso).
- Remove(color) – Destructor de pelotas: Cuando una pelota del color especificado entra por el control ésta se destruye.
- Convert – Coloreador de pelotas: Cuando una pelota entra por el control se cambia el color de la pelota al indicado.
- Ifleft, Ifright – Condicional: Cuando una pelota de un color indicado entra por el control se irá a la izquierda/derecha, mientras que si no es del color indicado irá a la otra parte (Un sólo uso).
- Dirchange – Cambio de dirección: Cuando una pelota de un color determinado entra por el control cambiará a la dirección opuesta (solo izquierda y derecha) (Un sólo uso).
- Stop(color) – Retenedor: Cuando una pelota entre en el control, ésta quedará inmóvil un tiempo, transcurrido ese tiempo la pelota continuará su camino (Un sólo uso).

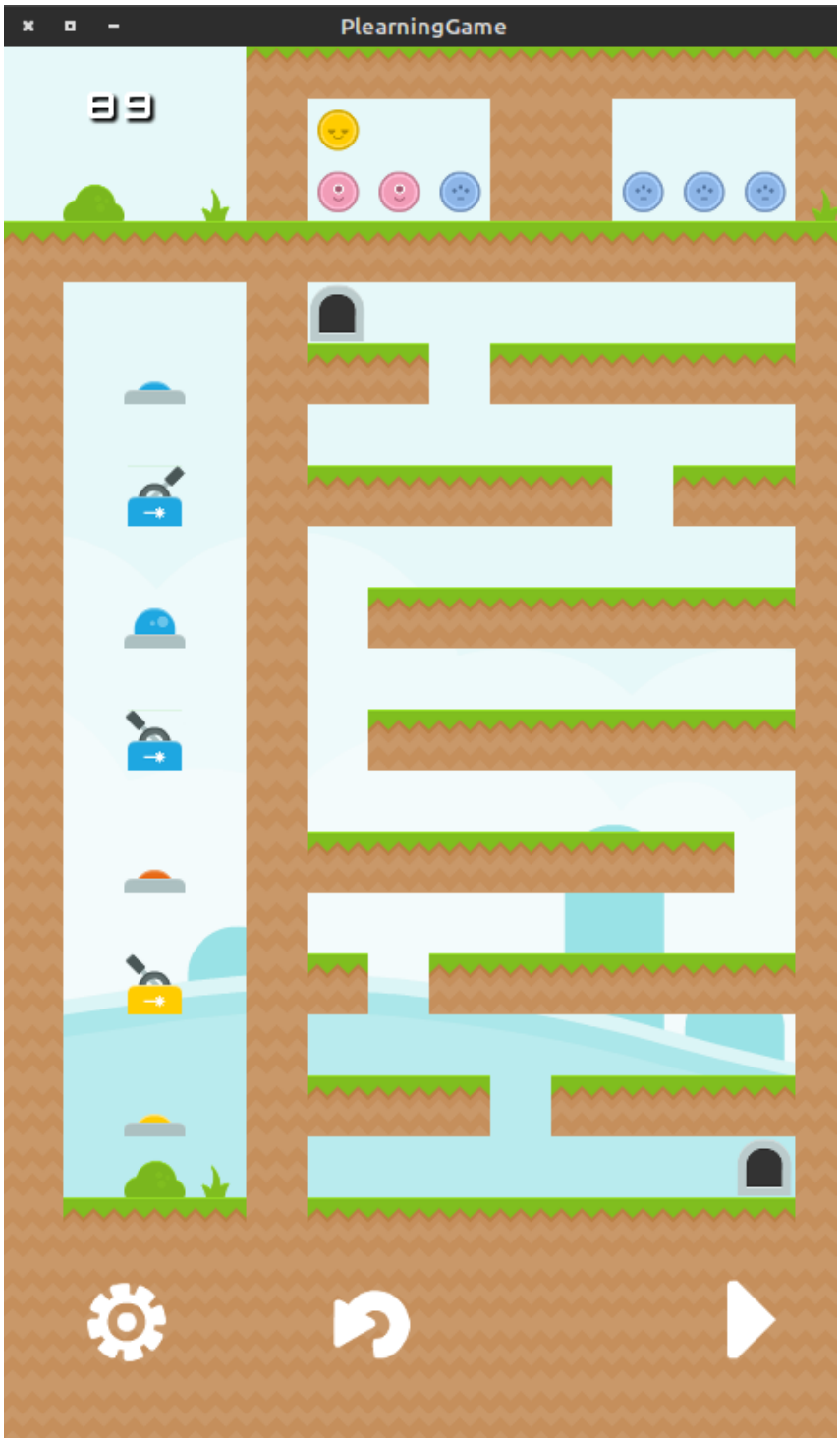

Figura 9.5: Ejemplo de nivel de juego

Una vez colocados los controles en el laberinto (parte derecha) podremos pulsar “play” o dejar que acabe el tiempo. Las pelotas de la entrada caerán de la puerta superior izquierda del laberinto, rodarán hacia la derecha hasta chocar con una pared, en ese caso cambiarán de dirección. Cuando caigan por un agujero seguirán con la dirección que tenían antes de caer. Mientras tanto interactuarán con los controles colocados y una vez que las pelotas pasen por la salida (puerta inferior derecha del laberinto) se comprobará si se cumple con la salida deseada. En caso de que sea así se habrá superado el nivel.

En las figuras 9.6, 9.7, 9.8, 9.9, 9.10 y 9.11; se puede observar una secuencia de juego válida para un nivel de ejemplo en el que se pide como salida tres pelotas azules y se ofrece como entrada una pelota azul, dos rojas y una amarilla, en ese orden. Los controles disponibles son:

- Destructor de pelotas azules
- Condicional a la derecha azul
- Coloreador de pelotas azul
- Condicional a la izquierda azul
- Destructor de pelotas rojo
- Condicional a la izquierda amarillo
- Destructor de pelotas amarillo

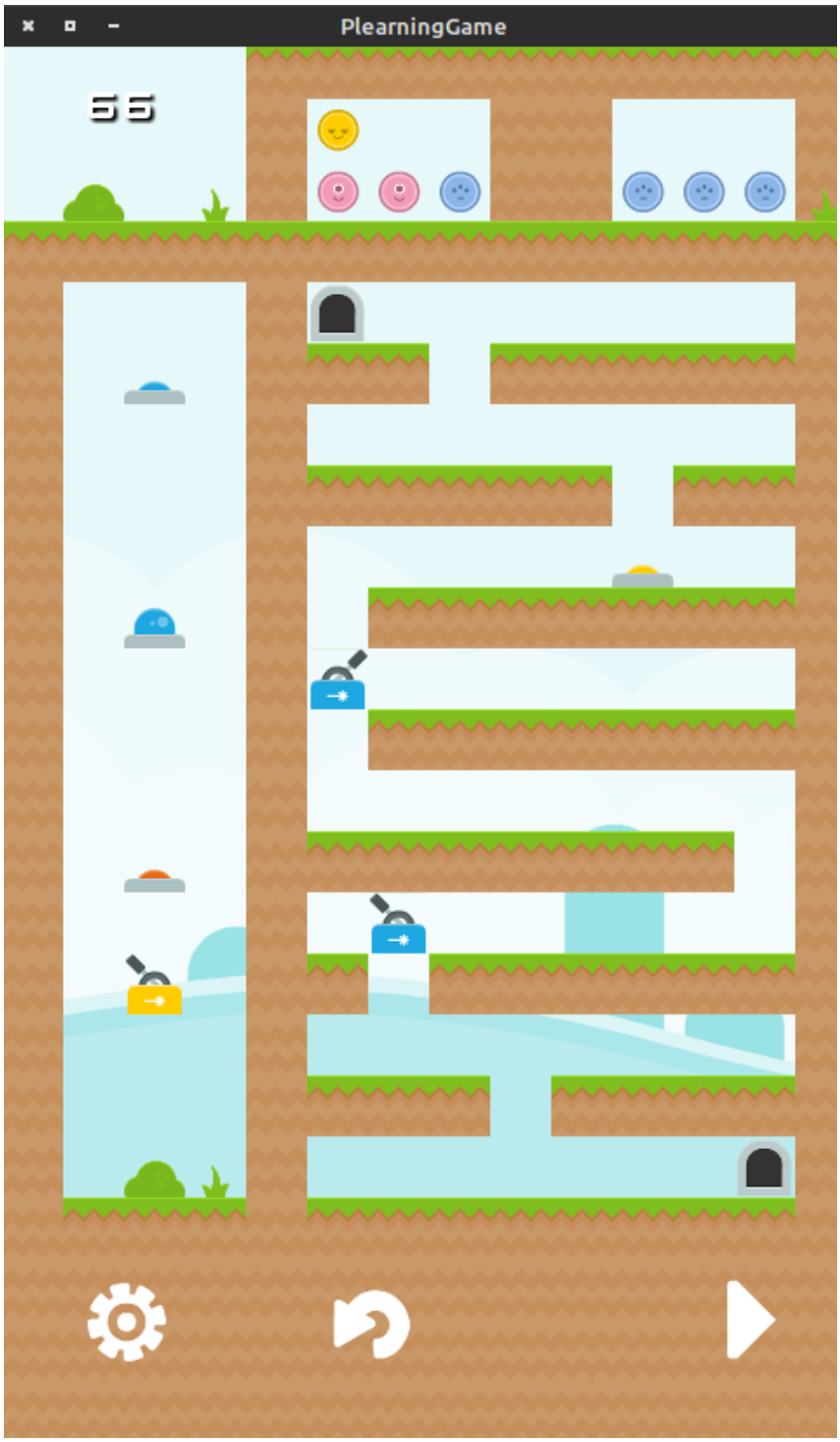

Figura 9.6: Secuencia de ejemplo de nivel de juego

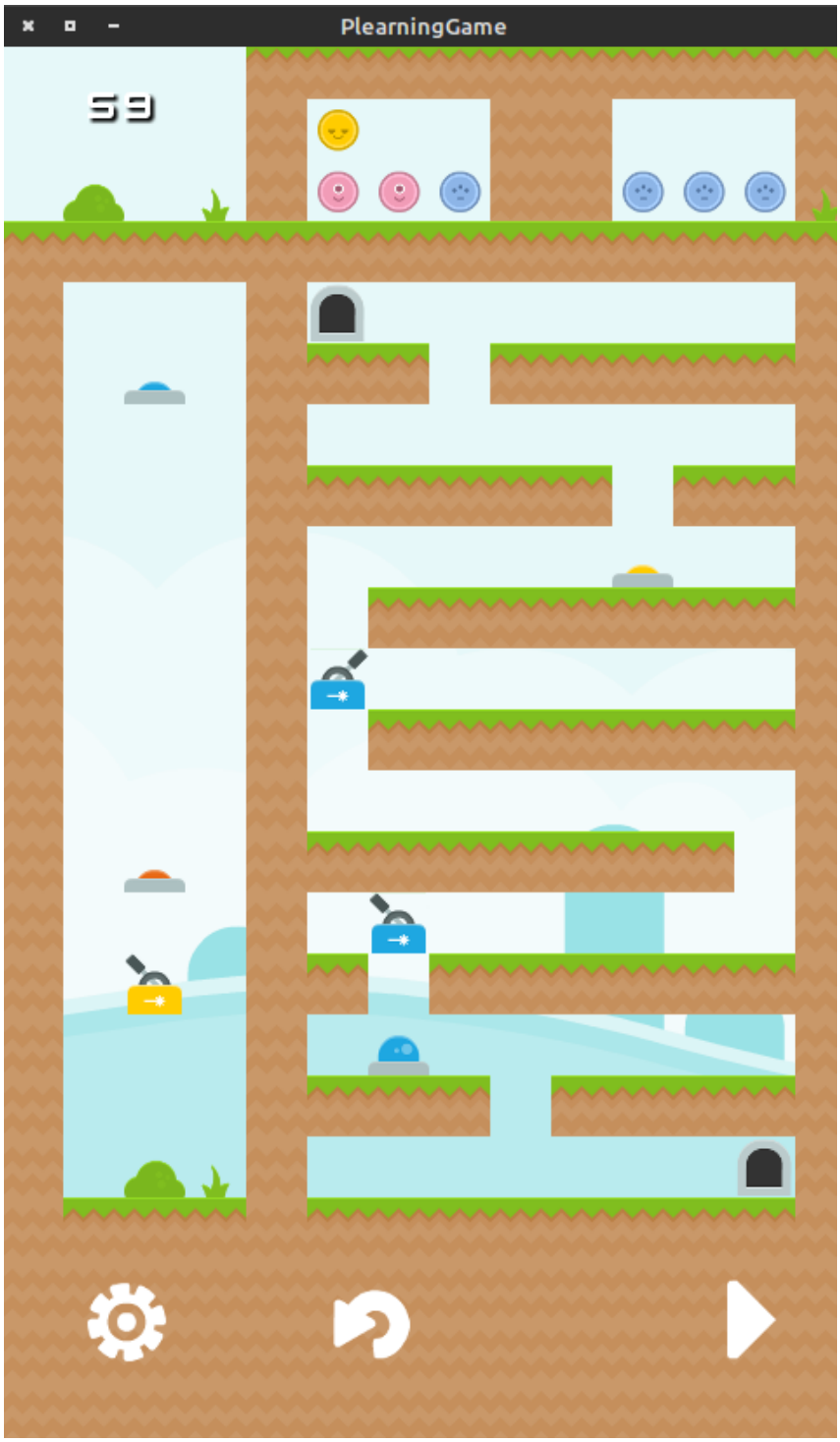

Figura 9.7: Secuencia de ejemplo de nivel de juego

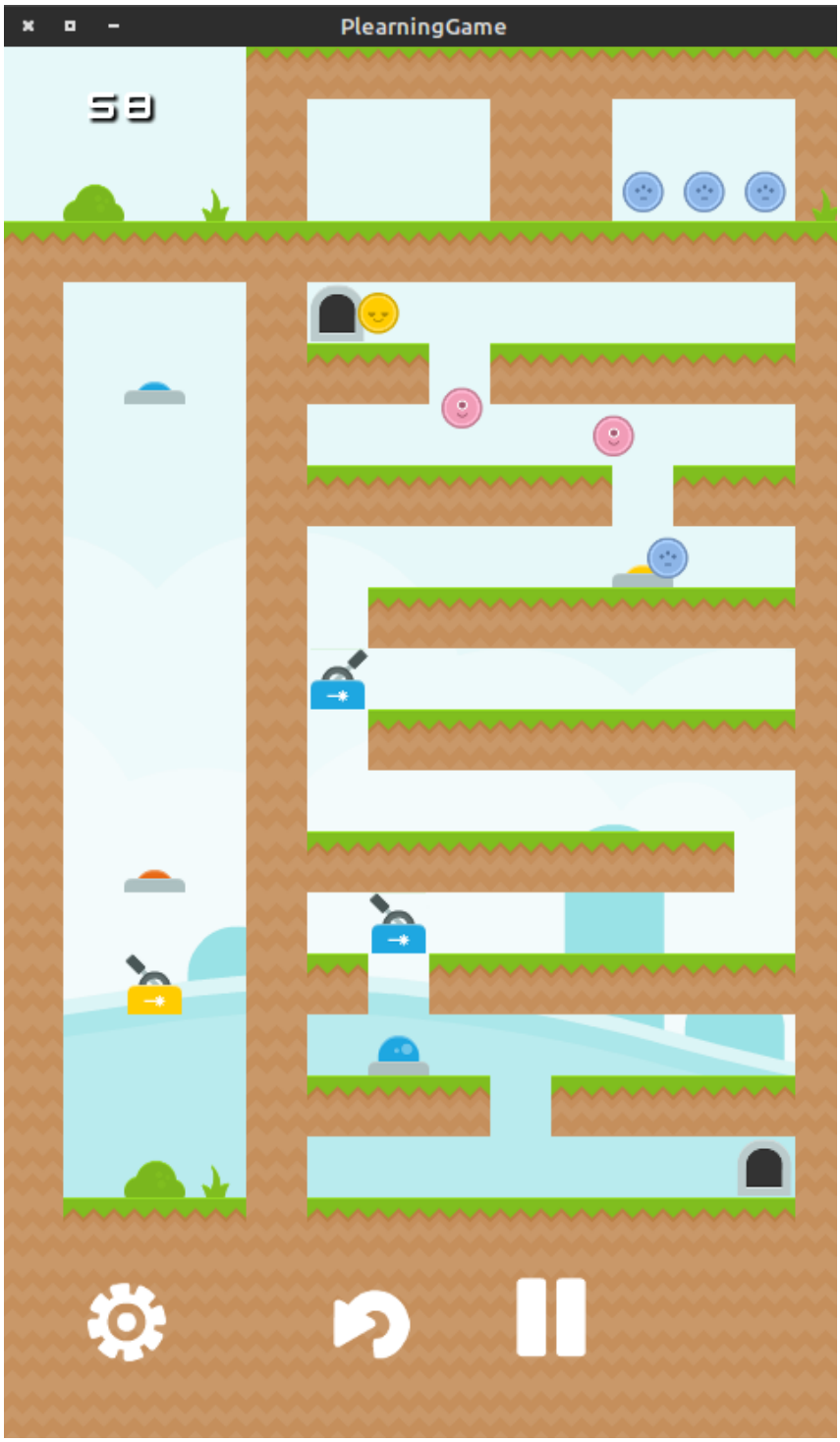

Figura 9.8: Secuencia de ejemplo de nivel de juego

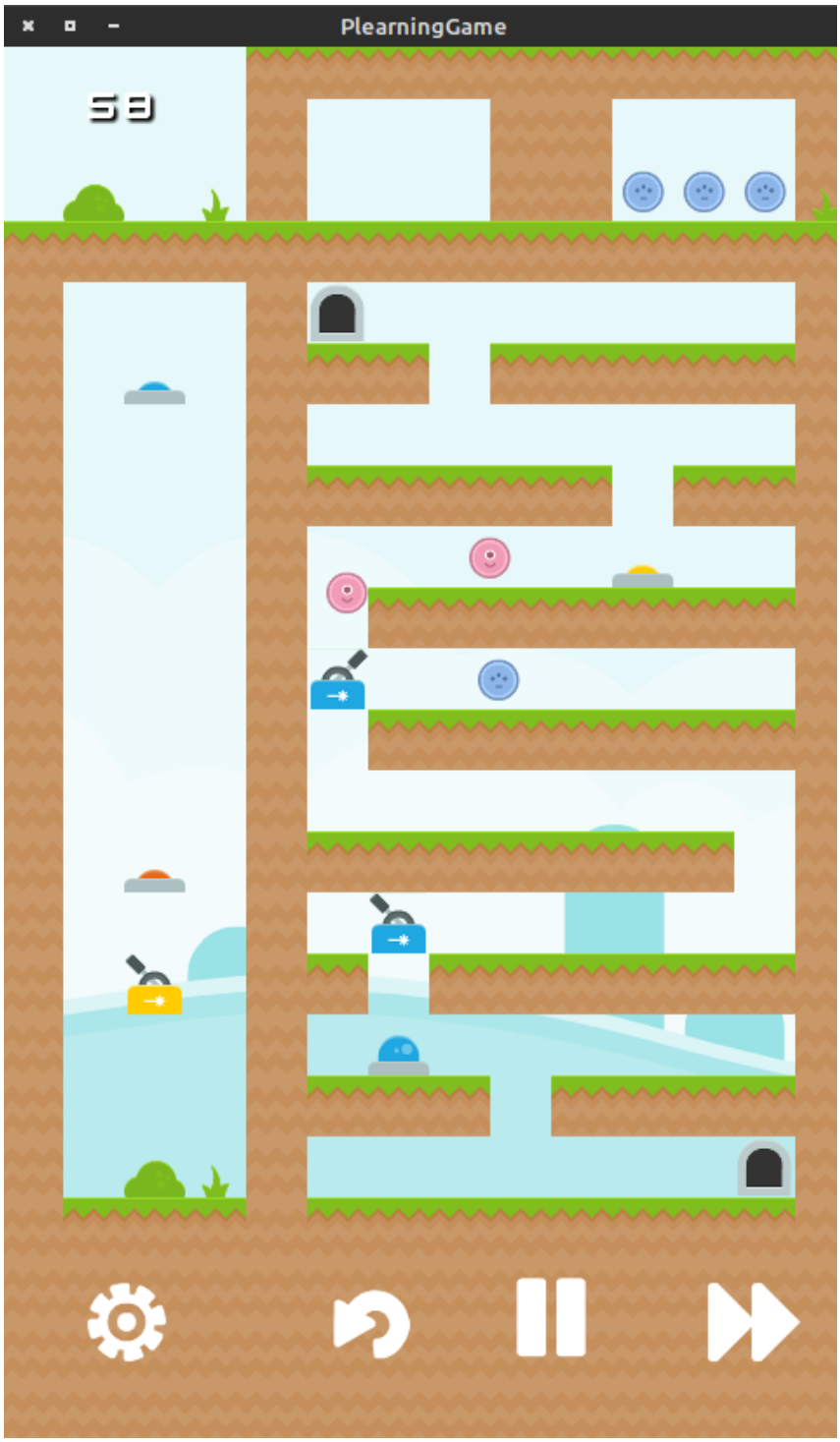

Figura 9.9: Secuencia de ejemplo de nivel de juego

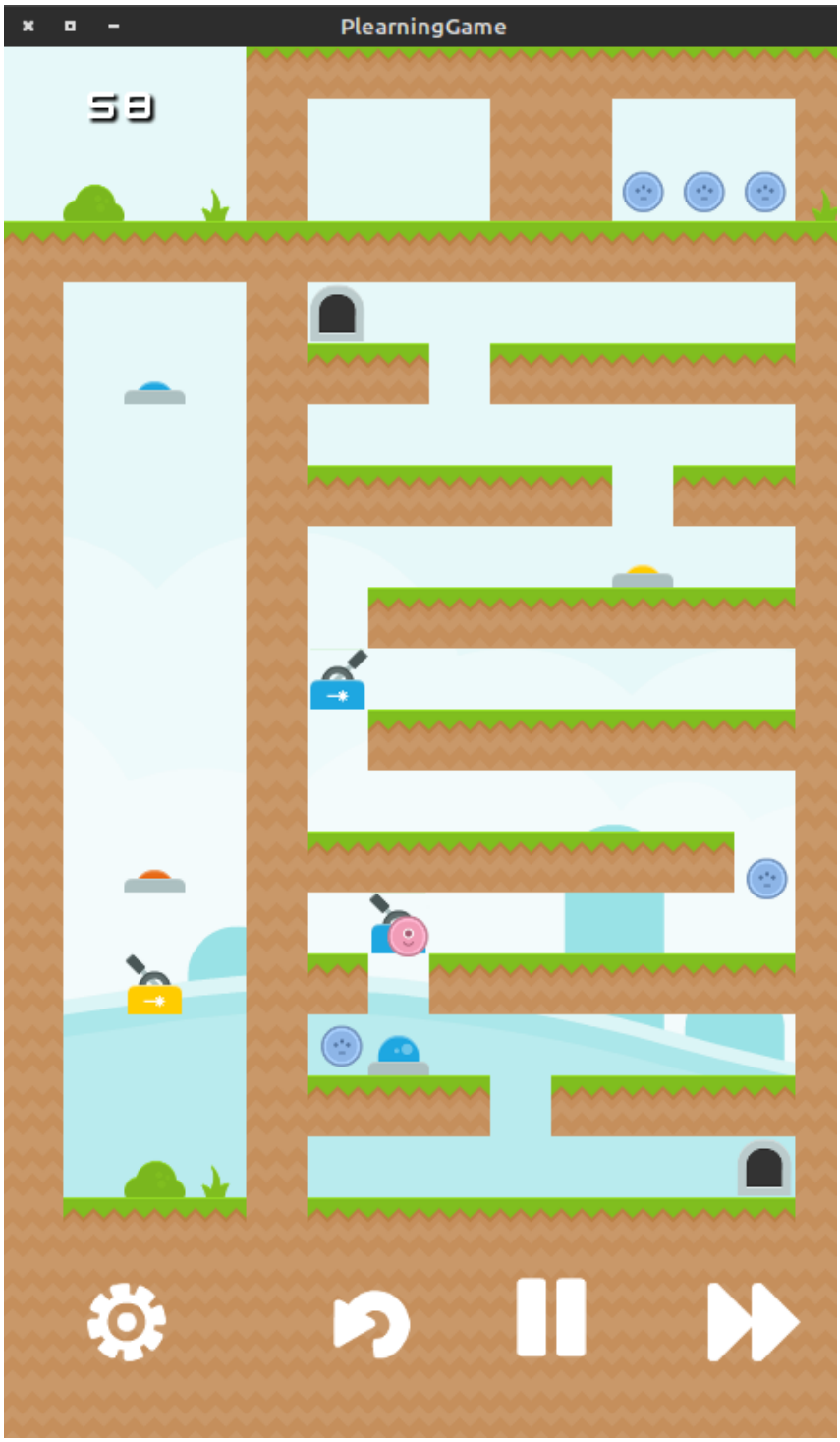

Figura 9.10: Secuencia de ejemplo de nivel de juego

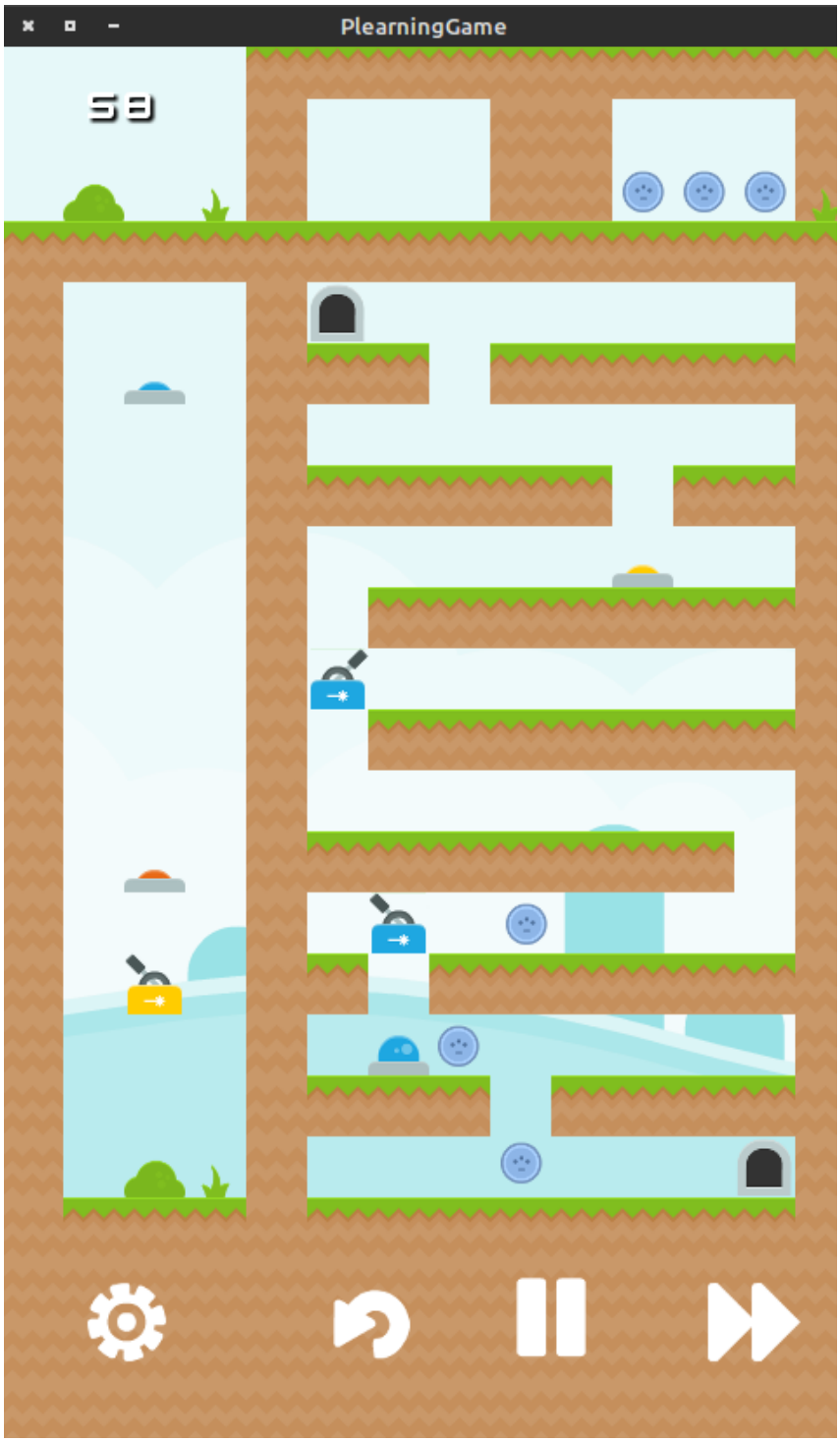

Figura 9.11: Secuencia de ejemplo de nivel de juego

Capítulo 10

Conclusiones

En este último capítulo se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el software desarrollado.

10.1. Objetivos alcanzados

Este apartado resume los objetivos generales y específicos alcanzados, relacionándolos con todo lo descrito en el capítulo de introducción.

El desarrollo del proyecto ha sido tal y como se esperaba en lineas generales, aunque han habido algunas desviaciones de tiempo durante la implementación de éste. Cabe destacar que el hecho de poder probar el proyecto ejecutándolo en la propia máquina donde se ha desarrollado, sin tener que utilizar un dispositivo real o virtual cada vez que se ha querido verificar, ha hecho que el desarrollo sea mucho más fluido. Esto no significa que en ciertos momentos del desarrollo no se haya probado con un dispositivo real Android.

Por tanto, el hecho de haber escogido libGDX para el desarrollo ha sido una buena elección. Es también una buena elección escoger un framework o un motor focalizado al desarrollo de videojuegos, si se quiere implementar uno ya sea para Android o para cualquier plataforma.

Además con el tiempo de que se ha dispuesto para este proyecto, queda aún más justificado la elección de este framework para realizarlo. Ya que probablemente sin éste, debido al esfuerzo mayor en desarrollar apartados que no son exclusivos del videojuego, podrían haber existido desviaciones de tiempo importantes en relación al calendario propuesto.

También se han desarrollado unos niveles de prueba pero sí que es importante desarrollar un mejor sistema para la creación de niveles, de ahí el diseño realizado ya que proporcionan la infraestructura necesaria para crear más mundos y niveles de forma sencilla.

Ha quedado fuera del proyecto por temas de tiempo realizar el sistema de puntuaciones y premios para así conseguir una mejor experiencia de usuario.

Ha quedado pendiente realizar una mejora gráfica a la UI de selección de nivel y mundo. Además de desarrollar la animación de las pelotas y controles. Al ser más un apartado gráfico que de desarrollo me he permitido bajar la prioridad de estos requisitos, quedándose fuera del desarrollo final.

Para resumir enumero a continuación los objetivos alcanzados favorablemente:

- OBJ-01 Menú principal del juego
- OBJ-02 Arte, diseño gráfico y sonido
- OBJ-03 Pantalla de ayuda del juego
- OBJ-04 Pantalla de opciones del juego
- OBJ-05 Pantalla de selección de mundo y nivel del juego
- OBJ-06 Lógica de juego
- OBJ-07 Gestión de los controles del juego
- OBJ-08 Gestión de las entradas y salidas del juego
- OBJ-09 Captura de objetivos del juego
- OBJ-10 Gestión de los laberintos del juego
- OBJ-11 Gestión de niveles del juego

10.2. Lecciones aprendidas

A continuación, se detallan las buenas prácticas adquiridas, tanto tecnológicas como procedimentales, así como cualquier otro aspecto de interés.

He aprendido sobre todo a gestionar los tiempos y resolver incidencias. Es importante destacar que de los riesgos enunciados en la planificación del proyecto se han cumplido la mayoría. Por tanto estuvo bien tener un plan de acción ante estos posibles riesgos para no paralizar el desarrollo del proyecto.

Sobre todo he aprendido lo necesario que es un equipo de trabajo, ya que es muy difícil que una sola persona maneje y domine tantos campos tecnológicos y herramientas a la vez.

Obviamente aún queda mucho trabajo ya que el tiempo disponible y el tiempo necesario para el aprendizaje de uso de la tecnología y herramientas ha hecho que tenga que acelerar el proceso de desarrollo y seleccionar entre requisitos prioritarios para un buen funcionamiento del sistema cumpliendo unos mínimos objetivos de calidad.

Además es importante remarcar que normalmente para el desarrollo de un videojuego completo es necesario un equipo de trabajo bastante heterogéneo y al tener que desarrollar labores para las que no me capacita mis estudios ha conllevado en importantes retrasos.

También he aprendido los principios básicos de diseño y desarrollo de videojuegos ya que es muy distinto a los sistemas y aplicaciones que acostumbraba a desarrollar durante el transcurso del grado.

He visto de forma clara lo importante que es llevar una documentación actualizada y paralela al desarrollo del proyecto ya que es muy fácil perder la visión global del sistema si no se mantiene documentado en ningún tipo de soporte.

10.3. Trabajo futuro

En esta sección, se presentan las diversas áreas u oportunidades de mejora detectadas durante el desarrollo del proyecto y que podrán ser abarcadas en futuras versiones del software.

Para las siguientes versiones del software he preparado una lista de posibles mejoras ordenadas por su prioridad de forma descendente:

- OBF-01 Sistema de puntuaciones y premios, gestionados en un servidor remoto
- OBF-02 Mejora del arte, animaciones y sonidos
- OBF-03 Desarrollo de nuevos controles e introducción de los controles específicos por cada mundo
- OBF-04 Publicación en Google Play Store, firmado de la aplicación y desarrollo de componentes para los juegos de Google
- OBF-05 API para desarrollo de niveles y mundos
- OBF-06 Mejora en el sistema de captura de clicks/touchs en el mapa
- OBF-07 Estudio sobre el impacto del uso de la aplicación en la mejora de las habilidades lógicas del usuario
- OBF-08 Distribución de la aplicación en colegios e institutos
- OBF-09 Migrar los datos persistentes de la memoria secundaria del dispositivo a un servidor remoto

Es necesario ver esta versión del sistema como una versión beta del juego ya que aún quedan apartados por probar y pulir. Además ya que es un proyecto de software libre sería muy interesante contar con la participación de la comunidad de desarrolladores libres para que juntos podamos trabajar en posibles mejoras y así yo pueda continuar aprendiendo sobre el mundo de desarrollo de videojuegos.

El trabajo futuro que debo desarrollar a nivel personal es seguir con el aprendizaje en desarrollo de videojuegos como una oportunidad de expandir mis propios conocimientos y como una oportunidad laboral muy deseable.

Bibliografía

[AEVI, 2015] AEVI, A. E. d. V. (10/10/2015). Videojuegos más vendidos en españa, http://www.aevi.org.es/videojuegos-mas-vendidos.

[Bose, 2014] Bose, J. (2014). *Libgdx Game Development Essentials*. Packt Publishing Ltd., Livery Place 35 Livery Street Birmingham B3 2PB, UK., 1 edition.

[Cejas Sánchez and Saltares Márquez, 2014] Cejas Sánchez, A. and Saltares Márquez, D. (2014). *Libgdx Cross-platform Game Development Cookbook*. Packt Publishing Ltd., Livery Place 35 Livery Street Birmingham B3 2PB, UK., 1 edition.

[eldiario.es, 2014] eldiario.es, P. (19/08/2014). Enseñanza de programación en las escuelas, http://www.eldiario.es/turing/Ninos-programadores-ensenanza-programacion-escuelas_0_293970921.html.

[Gartner, 2013] Gartner (24/06/2013). Crecimiento android, http://www.gartner.com/newsroom/id/2525515.

[Google, 2015] Google, A. (10/10/2015). Historia android, https://www.android.com/intl/es-419_mx/history/.

[MIT, 2015] MIT (10/10/2015). Herramientas scratch, https://scratch.mit.edu/.

[Periódico El Mundo, 2014] Periódico El Mundo (19/09/2014). Juegos para aprender a programar, http://www.elmundo.es/tecnologia/2014/09/19/541ad75c268e3e3f7c8b456d.html.

[uptodown, 2015] uptodown (02/06/2015). Videojuegos más descargados infografía, http://blog.uptodown.com/juegos-android-2015/.

[wikipedia, 2015] wikipedia (10/10/2015). Géneros de videojuegos, https://en.wikipedia.org/wiki/Category:Video_game_genres.

Información sobre Licencia

Incluir aquí la información relativa a la licencia seleccionada para la documentación y software del presente proyecto.

La licencia para esta documentación es la Free Document License de GNU que se detalla a continuación:

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary

Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.
The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.
The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".
Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.
The "**publisher**" means any person or entity that distributes copies of the Document to the public.
A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this

License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.
You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.
If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.
It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of

Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.
You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.
The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the

copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.
If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.
However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.
Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.
Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.
"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.
"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.
An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.
The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.
La licencia de todo el código fuente es la GNU General Public License Versión 3 (GNU GPLv3), detalada a continuación:

The GNU General Public License is a free, copyleft license for software and other kinds of works.
The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.
When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.
To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.
For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.
Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.
For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.
Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand

ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which

is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to

limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified ob-

ject code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work.

These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent

license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.