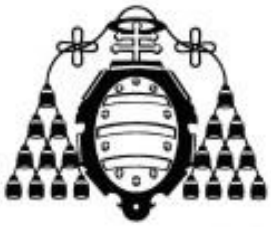
UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

"PLATAFORMA PARA LA GENERACIÓN DE VIDEOJUEGOS BASADOS EN EL CONTROL DE VEHÍCULOS ORIENTADO A DISPOSITIVOS MÓVILES"

DIRECTOR: B. Cristina Pelayo García-Bustelo

CODIRECTOR: Jordán Pascual Espada

AUTORA: Bárbara Posada Menéndez

Vº Bº del Director del Proyecto

Agradecimientos

En primer lugar, me gustaría dar las gracias a mis padres y amigos por haberme apoyado y ayudado en la medida de lo posible en esos momentos en los que el proyecto no marchaba lo bien que uno quisiera.

En segundo lugar, agradecer a Cristina Pelayo García haberme ofrecido un proyecto tan interesante como el presente y sobretodo agradecer a Jordán Pascual la ayuda prestada con la resolución de todas y cada una de las dudas que fueron surgiendo, que no fueron pocas. También quiero agradecer a Yuri Álvarez y a Verónica Viñuelas su aportación diseñando los gráficos de los dos videojuegos que surgieron después de tantas horas de trabajo.

Resumen

Gade4All es un proyecto actualmente en desarrollo cofinanciado por el Ministerio de Industria, Energía y Turismo dentro del Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica

Gade4All busca diseñar y desarrollar una plataforma software-hardware genérica que facilite la generación de videojuegos y software de entretenimiento. La plataforma implementa un conjunto de servicios que permitirán acelerar, abaratar y facilitar los procesos de desarrollo de videojuegos buscando como objetivo primordial convertirse en un lugar de reunión de los agentes de la industria del software de entretenimiento. Este proyecto encaja con la nueva realidad presente en Internet, donde son los propios usuarios los que generan los contenidos.

El presente proyecto se enmarca dentro de Gade4All en tanto que implementará el módulo destinado a videojuegos de plataformas basados en el control de vehículos. Por tanto, se desarrollará una aplicación gráfica usable e intuitiva que permita definir el comportamiento y los elementos específicos de dicho tipo de videojuegos a personas que no tengan conocimientos técnicos específicos sobre programación.

Una vez que el usuario defina los elementos pertinentes del videojuego se extraerá dicha información a un archivo en formato XML (eXtensible Markup Language) siguiendo un lenguaje de dominio específico (DSL) definido en el proyecto Gade4All. Con la información del XML se utilizará un analizador y diversas plantillas que personalizarán el videojuego con los parámetros que introdujo el usuario en primera instancia. La salida final de la aplicación será un videojuego de plataformas en el cual el personaje principal se moverá por un escenario con rampas y obstáculos tratando de alcanzar la meta en el menor tiempo posible. Dicho videojuego será totalmente funcional para dispositivos Android. A pesar de que se haya optado por orientar la aplicación a tecnología Android, el editor de Gade4All está preparado para generar videojuegos en todas las plataformas, por lo que eventualmente será posible exportar también en HTML5, iPhone y Windows Phone.

La tecnología utilizada para desarrollar el videojuego será el IDE Eclipse junto con el SDK de Android. Las eventuales pruebas se llevarán a cabo tanto en dispositivos móviles Android reales como emuladores de dicho SDK. Para el procesamiento de los XML se utilizarán editores XML dentro de Visual Studio 2010 y otras tecnologías ya embebidas dentro del proyecto Gade4All.

Palabras Clave

DSL, Gade4All, Videojuego de plataformas, Plantilla, Android, Java, C#, Eclipse, Visual Studio, Arquitectura dirigida por modelos.

Abstract

Gade4All is a project that is currently being developed and has been co financed by the Ministry of Industry, Energy and Tourism within the National Plan for Scientific Research, Development and Technological Innovation.

Gade4All seeks to design and develop a software-hardware platform that eases the videogames generation and entertainment software as far as possible. The platform implements a set of services that will allow accelerating, facilitating and cheapening videogames development processes. It seeks as one of its main objectives to become a meeting place for entertainment software agents. This project fits with the new reality in the Internet, where the users are the ones who generate the content.

This project is part of Gade4All as it will implement the module for platform games based on vehicle control. Therefore, a usable and intuitive graphical application will be implemented in order to define the behaviour and specific elements of vehicle games to people who have no specific expertise on game programming.

Once the user defines the elements of the game, that information will be extracted to a file in XML (eXtensible Markup Language) according to a domain specific language (DSL) that has already been defined in Gade4All project. Analysing the XML document and using various templates will customize the game with the parameters the user entered in the first instance. The final output will be a vehicle videogame where the main character will move along scenery with slopes and obstacles trying to reach the goal in the shortest time possible. That videogame will be totally functional for Android devices. Despite it has been chosen to orient the application to Android technology, Gade4All editor is prepared to generate videogames in diferent platforms so it will eventually be possible to export HTML5, iPhone and Windows Phone format.

The technology that will be used to develop the videogame will be IDE Eclipse with Android SDK. Tests will be carried out both in real Android devices and SDK emulators. For XML processing, XML editor will be used with Visual Studio 2010 and other technologies embedded within the Gade4All project.

Keywords

DSL, Gade4All, Platform videogame, Template, Android, Java, C#, Eclipse, Visual Studio, Model driven engineering.

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

1.1.1 Motivación del proyecto

Un videojuego es todo aquel software creado para entretener a la persona o personas que interactúan con él a través de un controlador o aparato electrónico que ejecuta dicho videojuego.

Los primeros videojuegos modernos surgieron en la década de los 60, a medida que la informática avanzaba y el hardware se hacía más complejo y más pequeño. Desde el primer videojuego llamado *Spacewar!* desarrollado en las inmediaciones del *Massachussets Institute of Technology* (M.I.T.) que sólo podía jugarse en terminales especializados hasta las primeras máquinas recreativas donde despuntaron títulos como *Pacman* o *Pong* conocidos por los usuarios incluso hoy en día.

Las primeras videoconsolas llegaron a los hogares a finales de la década de los 70 de la mano de la empresa japonesa Atari. La irrupción del color y los gráficos vectoriales aumentó la variedad de videojuegos disponibles hasta límites incalculables durante lo que se conoce como la edad de oro de los videojuegos que llegó hasta principios de los 80.

En la década de los 80 llegó el boom de los ordenadores personales como Commodore o ZX Spectrum, por lo que el emergente negocio de los videojuegos no se limitaba a terminales específicos o máquinas recreativas.

A finales de los 80 y principios de los 90 el mercado ya no era un monopolio de Atari. Había aparecido Sega con personajes tan conocidos como *Sonic* o Nintento con *Mario Bross*. La Game Boy de Nintento supuso una nueva revolución aportando una consola portátil con un precio asequible y amplia variedad de títulos disponibles.

Las Nintendo64 y la Playstation fueron las primeras videoconsolas de éxito que contaban con videojuegos con gráficos en 3-D. Este hecho acabó por catapultar a la industria de los videojuegos como una industria multimillonaria de dimensiones inimaginables años atrás.

A diferencia de décadas atrás y debido a la complejidad que han adquirido los videojuegos actuales, los programadores deben trabajar en una parte muy concreta del producto dentro de un equipo de desarrollo con diseñadores, músicos, *testers* y otros programadores a menudo bajo el control de grandes multinacionales.

Sin embargo, la irrupción de los *smarthphones* en el mercado actual ha potenciado la industria de videojuegos más sencillos y adaptados al hardware de los dispositivos móviles destronando a los videojuegos de consolas. A través de la App Store o de Google Play (cuyos enlaces pueden encontrarse en la sección 12.2 Referencias de Internet) los usuarios pueden descargarse juegos para sus iPhone o dispositivos Android respectivamente de manera gratuita o por precios más que asequibles. Esta facilidad para la adquisición de productos de entretenimiento ha hecho que el número de aplicaciones descargadas ya alcance la apabullante cifra de 25 billones en Google Play.

El proyecto Gade4All nace ante el auge masivo de los juegos para dispositivos móviles, especializándose en los principales sistemas operativos que éstos usan (Android, iPhone, Windows Phone) además de la tecnología HTML5 para ordenadores personales.

De esta manera, el usuario que desee generar un videojuego no precisará de los conocimientos específicos que se necesitan para cada sistema operativo y el proceso de desarrollo tedioso y largo se reducirá significativamente. Además, los elementos de cualquier videojuego se estandarizarán gracias a un lenguaje de dominio específico (DSL) que fomentarán el uso y la extensión de la aplicación por parte de otros usuarios del mundo de los videojuegos.

En concreto, este proyecto se centrará sobre los videojuegos de vehículos debido a la cantidad de descargas de juegos de ese tipo en Google Play, por ejemplo *Hill Climb Racing, Skater Boy, BMX Boy, Stickman Ski Racer, Moto X Mayhem,* etc.

1.1.2 Objetivos genéricos

- Crear una aplicación intuitiva y sencilla para el usuario no especializado que le permita definir los elementos de un videojuego de plataformas 2-D donde el personaje principal se mueva de izquierda a derecha de la pantalla subiendo y bajando inclinaciones de diferentes pendientes y sortee obstáculos hasta llegar a la meta dentro de un tiempo determinado.
- Extender la funcionalidad de un editor de videojuegos (Gade4All) ya implementado para agilizar los procesos de creación de videojuegos a usuarios no expertos.
- Generar videojuegos como proyectos Eclipse que puedan ser ejecutados en dispositivos Android con versión superior a la 2.0.

1.1.3 Alcance

El presente proyecto se especializará únicamente en videojuegos en los cuales un vehículo se mueve de izquierda a derecha de la pantalla subiendo y bajando inclinaciones y sorteando obstáculos hasta que consigue alcanzar su meta.

Como es lógico, el usuario no podrá personalizar todos los elementos del videojuego pues esta opción requeriría una parte de programación y uno de los objetivos principales es que no se necesitan conocimientos técnicos concretos para generar un juego.

Los elementos personalizables serán, por tanto, finitos pero lo suficientemente variados como para generar distintos tipos de videojuegos introduciendo diferentes parámetros. Abarcarán desde sonidos, tipo de personaje principal, velocidad del personaje principal, situación, elevación y longitud de las rampas, situación de los obstáculos, lugar de la meta, gravedad del entorno, etc.

Actualmente Gade4All cuenta con editores para desarrollar videojuegos de diversas tipologías (puzles, trivial, plataformas, estrategia y habilidad). El objetivo del presente proyecto consiste en crear una nueva rama del editor que permita dar soporte a la creación completa de juegos de la nueva tipología orientada al control de vehículos basándose en la arquitectura de los editores ya implementados.

1.2 Resumen de Todos los Aspectos

Este subapartado resume el contenido de cada apartado de la presente documentación para facilitar la búsqueda y comprensión del contenido al lector.

1.2.1 Memoria del proyecto

La memoria del proyecto es un resumen escrito para personas que no posean conocimientos específicos sobre el mismo. La memoria se divide en motivación del proyecto, objetivos genéricos y alcance.

Este apartado ahonda en el contexto histórico de los videojuegos para justificar la necesidad de un generador de software de entretenimiento como el que propone Gade4All. Se citan una serie de objetivos muy genéricos en los cuales se profundizará en apartados posteriores y se expone brevemente en qué consiste el proyecto y qué funcionalidad se pretende alcanzar.

1.2.2 Introducción

El apartado de introducción supone el primer acercamiento a todos los ámbitos del proyecto centrándose en su justificación, sus objetivos de manera más concreta que el apartado de memoria y un estudio de la situación actual en el que se identifiquen y describan sistemas similares al que se va a desarrollar.

En el primer apartado de la introducción se ofrece una justificación del proyecto Gade4All en general y del módulo que implementará el presente trabajo además de sus características principales y los objetivos a alcanzar de manera más específica. En el estudio de la situación actual se analizan las similitudes y diferencias de Gade4All frente a tres aplicaciones que permiten la definición de videojuegos a usuarios no expertos ensalzando las ventajas que el enfoque de Gade4All aporta.

1.2.3 Aspectos teóricos

Esta sección está destinada a describir brevemente aquellos conceptos, herramientas y tecnologías existentes que se vayan a usar en el proyecto.

En el caso concreto de este trabajo se explica de manera detallada pero accesible los conceptos de ingeniería dirigida por modelos, lenguaje de dominio específico o DSL, juegos basados en *Tiles,* Android, Métrica 3 y UML.

1.2.4 Planificación del proyecto y resumen de presupuestos

Esta sección de divide en dos subsecciones principales. La primera trata de ofrecer una planificación preliminar en la que se describa el plazo de ejecución del proyecto de forma que puedan fijarse las expectativas de quienes van a recibir el producto resultado del mismo.

Se ha incluido un cronograma realizado con GanttProject tipo diagrama de Gantt que incluye una planificación inicial de los plazos en los que se llevarán a cabo las diferentes tareas del proyecto.

La segunda sección aporta un presupuesto aproximado del precio de los recursos a utilizar para la realización del proyecto.

1.2.5 Análisis del sistema

En la fase de análisis de cualquier proyecto se efectúa una primera descripción precisa y clara del mismo. Debe ser entendible tanto por cualquier miembro del equipo técnico como por cualquier cliente para que éste pueda cambiar o modificar cualquier aspecto antes de comenzar con el desarrollo.

En este proyecto, la sección de análisis incluye una definición más precisa del sistema que las secciones anteriores, los requisitos del sistema definitivos y que deben comprobarse al finalizar el proyecto, identificación y descripción de los casos de uso, diagrama de clases con las clases más importantes que se piensan utilizar y su cometido, análisis y escenarios de los casos de uso identificados, prototipos de las pantallas de la aplicación y especificación del plan de pruebas.

1.2.6 Diseño del sistema

La fase de diseño profundiza y detalla los procesos que fueron presentados en el análisis del sistema. Durante esta fase se desarrollan, revisan y documentan las estructuras de datos a utilizar en el proyecto final, la estructura de la aplicación y los detalles a nivel de código de la misma.

En la sección de diseño del presente proyecto se incluye un diagrama de paquetes de la aplicación a nivel general, concretando cada paquete a través de su diagrama de clases. Asimismo se incluyen los diagramas de interacción y actividades para los principales casos de uso, el diseño de la interfaz a través las pantallas ya elaboradas de la aplicación, un listado detallado de todos los elementos involucrados en el diseño del DSL y la especificación del plan de pruebas que se llevará a cabo en secciones posteriores.

1.2.7 Implementación del sistema

La implementación del sistema abarca toda aquella información relativa a la generación de código desarrollado. Por tanto se incluye un breve resumen de los lenguajes de programación utilizados en el proyecto, las herramientas y programas usados para el desarrollo del mismo, los problemas encontrados durante la implementación y cómo se resolvieron y una guía detallada de las clases más importantes de la aplicación.

1.2.8 Desarrollo de las pruebas

Para asegurarse que el producto software final funciona correctamente y no presenta errores se deben realizar una serie de casos de prueba de diferente ámbito que aseguren este aspecto.

En esta sección se incluyen los resultados de las pruebas unitarias, pruebas de integración y pruebas de usabilidad a usuarios, así como los cambios que se han efectuado para resolver los errores detectados en esta fase.

1.2.9 Manuales del Sistema

Esta sección contiene los manuales que permiten trabajar con la herramienta a un usuario no experto o sirven de ayuda a eventuales desarrolladores que quieran modificar o mejorarla en un futuro. Este proyecto en concreto ofrece los siguientes manuales:

- Manual de instalación.
- Manual de ejecución.
- Manual del usuario.
- Manual de programador.

1.2.10 Conclusiones y ampliaciones

En esta sección se exponen brevemente los pasos que se han seguido durante el desarrollo de proyecto y los objetivos que se cumplen en cada fase y en la parte final del mismo. Se incluyen además una serie de mejoras para la aplicación a nivel generar que no han podido llevarse a cabo por falta de tiempo o falta de recursos.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

Gade4All es un ambicioso proyecto que pretende ofrecer una aplicación que permita crear múltiples variedades de videojuegos multiplataforma a personas sin conocimientos específicos sobre el desarrollo y la programación de los mismos. La versión completa incluirá juegos tipo trivial, de estrategia, de habilidad mediante control táctil, de plataformas, etc. y la salida final de la aplicación será un videojuego completamente funcional para las plataformas de Android, iPhone, Windows Phone y HTML5.

Debido a la amplitud del proyecto, el presente trabajo se centrará en el desarrollo específico de videojuegos de vehículos dentro del género de plataformas para la tecnología Android aunque se procurará que la integración para otras plataformas ser pueda realizar de la forma más directa posible.

Se creará una nueva rama en el interfaz gráfico que soporte la definición de la nueva tipología de plataformas basado en el control de vehículos respetando la arquitectura del editor ya existente.

Los principales elementos configurables del juego serán:

- **Física del juego**: gravedad del entorno, velocidad máxima del coche y coeficientes de rozamiento.
- **Gráficos**: imágenes de fondo, imagen para el personaje principal, imágenes para las trampas, imágenes para los puntos de control y botones.
- **Estructura**: se podrá especificar el recorrido a seguir por el personaje principal incluyendo superficies rectas o cuestas de diferentes inclinaciones, zona de salida, zona de llegada, trampas, ítems a recolectar, etc.
- **Niveles**: se podrá definir el número de niveles del juego y el tiempo requerido para superar el nivel.
- **Sonidos**: se podrá definir el sonido ambiente además de los diferentes sonidos que se ejecutarán ante determinados eventos.
- **Forma de pasar de nivel**: el usuario podrá elegir de qué manera se pasará al siguiente nivel, ya sea llegando a la meta sin agotar las vidas y/o haciéndolo dentro de un límite de tiempo.

Se ha elegido esta tipología de juego debido al éxito de juegos similares como *Hill Climb Racing, Skater Boy o Stickman Ski Racer* cuyos enlaces en Google Play pueden encontrarse en la sección 12.2 Referencias en Internet.

Una vez que el usuario defina su videojuego, los parámetros configurables se almacenarán en un XML cuya especificación sigue un lenguaje de dominio específico ya existente llamado GSL (Game Specific Language) que es propio del proyecto Gade4All. La información del XML se procesará mediante un analizador y se generará un proyecto Android en código Java. De esta manera, la aplicación también será útil para aquellos programadores expertos que usarán el proyecto generado como base para continuar personalizado su videojuego. Asimismo, el proyecto final también contendrá el archivo .apk para instalar un juego completamente funcional en dispositivos de tipo Android.

2.2 Objetivos del Proyecto

- Crear una nueva rama del editor gráfico que permita al usuario definir de manera intuitiva, sencilla y rápida los elementos de un videojuego de plataformas orientado al control de vehículos.
- Utilizar un DSL existente y adaptarlo a las necesidades del videojuego y la aplicación en general para fomentar el uso de un lenguaje estándar en la definición de videojuegos.
- La salida del generador será un proyecto de tipo Android por lo que un usuario con conocimientos técnicos sobre desarrollo de videojuegos podrá tomar ese proyecto y realizar los cambios que considere oportunos sobre un juego ya funcional.
- El generador de videojuegos será pensado para integrar fácilmente módulos que generen el mismo juego en diversas tecnologías (HTML5, iPhone y Windows Phone).
- El videojuego final deberá funcionar correctamente en cualquier dispositivo Android con versión superior a la 2.0 e independientemente de la resolución de la pantalla del mismo. Además, la velocidad del juego y la interacción de sus elementos deberá ser coherente y atractiva para el usuario.

2.3 Estudio de la Situación Actual

2.3.1 Gade4All

Gade4All es un proyecto muy grande en el que han participado varios desarrolladores durante más de año y medio. A pesar de que el presente proyecto se encargará de implementar una parte dentro de la totalidad del proyecto Gade4All conviene tener en cuenta la funcionalidad actual del editor para poder ubicar el alcance del presente proyecto.

	Gade4All
¿Exige conocimientos previos de programación?	No
¿Genera un proyecto personalizable, no sólo un ejecutable?	Sí
¿Genera juegos para diversas plataformas?	Sí (Android, HTML5, Windows Phone y iPhone)
¿Ofrece diferentes tipologías de juegos?	Sí (trivial, plataformas, habilidad, control táctil y puzle)
¿Permite salvar y guardar el estado de edición?	Sí
¿Presenta una interfaz fácil de usar sin necesidad de aprendizaje?	Sí
¿Los elementos del juego se definen de manera visual?	Sí
¿Permite definir sonidos para determinados eventos?	Sí
¿Permite cambiar el comportamiento predefinido de los elementos del juego?	No
¿Permite definir pantallas previas al juego?	Sí
¿Permite modificar niveles de manera externa al editor?	Sí, pueden modificarse directamente los ficheros XML
¿Sigue un estándar crear los diferentes tipos de videojuegos?	Sí, todos los parámetros de las diferentes tipologías se definen a través de un GSL, un lenguaje de dominio específico propio del proyecto Gade4All especializado en videojuegos.
¿La herramienta será gratuita?	No

2.3.2 Evaluación de Alternativas

A continuación se detallan una serie de herramientas que también tienen por objetivo facilitar el desarrollo de videojuegos a usuarios no expertos comparando sus características con el proyecto Gade4All.

2.3.2.1 RPG Toolkit

RPG Toolkit [1] es un proyecto de código abierto y gratuito que ofrece una herramienta simple, flexible y potente centrada sobre todo en juegos de rol en 2-D. Fue lanzado por primera vez en 1997 por Christopher B. Matthews y actualmente tiene una amplia comunidad de usuarios y programadores que ayudan a mejorar y a testear la herramienta.

Los videojuegos se programan en un lenguaje de *scripting* propio llamado *RPG Code* que permite programar los elementos básicos del juego en relativamente poco tiempo. Además, incluye la posibilidad de utilizar un editor gráfico que posee todas las herramientas y plugins necesarios para dotar de más funcionalidad al videojuego.

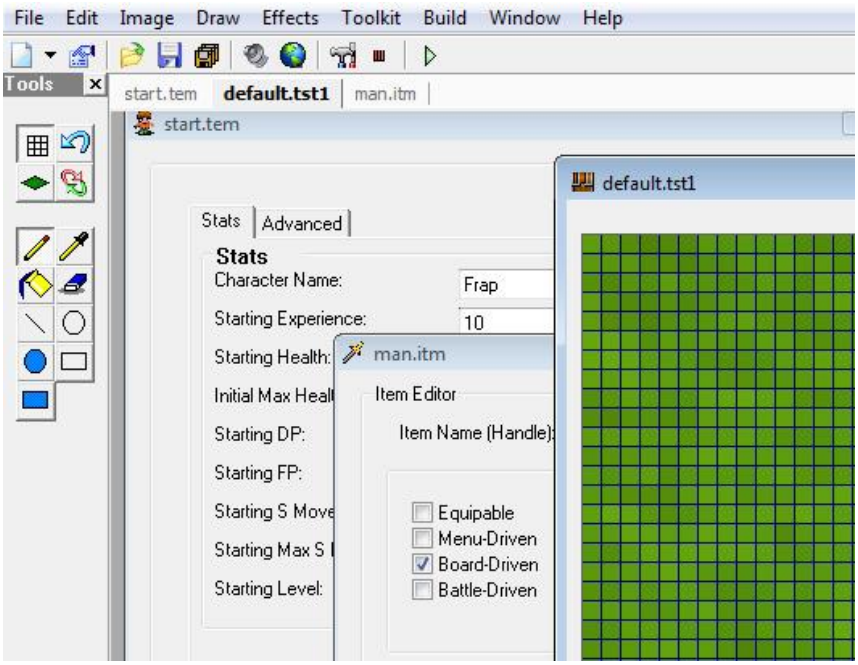


Ilustración 1. Editor gráfico de RPG Toolkit

2.3.2.1.1 Similitudes

RPG Toolkit presenta una única similitud significativa con el generador de videojuegos de Gade4All:

- Ambos editores gráficos permiten definir enemigos, movimientos y otros elementos con pocos clics del ratón.

[1] http://rpgtoolkit.net/

2.3.2.1.2 Diferencias

- RPG ToolKit viene respaldada por una amplia comunidad de programadores que mejoran cada cierto tiempo la aplicación. En Gade4All han trabajado muchos programadores implementando los diferentes tipos de videojuegos pero no se ha planteado tener una comunidad de usuarios que desarrollen y mejoren la aplicación a corto plazo tras su lanzamiento.

- Para usar RPG Toolkit es necesario tener conocimientos básicos de programación mientras que con el presente proyecto se podrá tener un juego completamente funcional sin escribir una línea de código.

- Para crear nuevos videojuegos se usará un lenguaje propio llamado *RPG Code.* Esto fomenta la baja reutilización del código pues sólo pocas personas lo conocen. Además fuerza a otros programadores a conocer un lenguaje de programación nuevo. En el caso de que se requiera programar la salida de la aplicación de Gade4All se hará en el lenguaje propio del dispositivo destino, todos lenguajes de programación ampliamente reconocidos a nivel mundial. En concreto, el presente proyecto se implementará en Java.

- RPG Toolkit devuelve un ejecutable que se podrá ejecutar sólo en PC's. Gade4All devolverá el ejecutable y el proyecto completo totalmente personalizable para diversas plataformas, no sólo para PC.

- Aunque permite una gran personalización de los juegos, se centra únicamente en aquellos de tipo rol en 2-D que con la transformación adecuada pueden llegar a ser de plataformas. Gade4All contará con videojuegos de todo tipo: trivial, física, plataformas, puzles, etc.

2.3.2.2 Game Editor

Game Editor [2] es un programa de diseño de videojuegos de código abierto que permite crear videojuegos de todo tipo en 2-D. Fue lanzado en 2002 por Makslane Rodrigues, que ha estado desarrollándolo desde su lanzamiento. Actualmente sigue en proceso de desarrollo para incluir nuevas funcionalidades y mejoras.

La interfaz gráfica que propone es relativamente sencilla y es la misma para Linux, Windows y Mac OS. Está basada en ventanas y está compuesta por un menú principal en la parte superior que muestra la posición actual de la pantalla que se está editando además de diferentes iconos para indicar al usuario el modo de trabajo en el que se encuentra.

[2] http://game-editor.com/

Ilustración 2. Interfaz gráfico del Game Editor

Game Editor se basa en el concepto de realizar una acción cuando se produce cierto evento de entre los predefinidos. Esa acción se programa mediante un lenguaje de *scripting* similar a C en un editor. Mediante código se puede acceder a los actores (objetos) por su nombre para que realicen diferentes acciones.

2.3.2.2.1 Similitudes

Game Editor posee bastantes semejanzas con el generador de videojuegos de Gade4All. Son las siguientes:

- Ambos editores gráficos permiten definir enemigos, movimientos y otros elementos con pocos clics del ratón.
- Ambos permiten editar y crear diferentes niveles con diferentes elementos y diferente grado de dificultad.
- Ambos permiten generar tipologías muy diversas de videojuegos.
- Ambos se centran en ofrecer una interfaz intuitiva y sencilla a los usuarios.

2.3.2.2.2 Diferencias

- Aunque ambos se caracterizan por ser una *cross-platform*[3]están orientados a diferentes sistemas operativos/dispositivos. Game Editor permite exportar a Windows, Pocket PC, Windows Phone, GP2X, Linux y recientemente ha incluido la exportación para Mac Os X, iPhone e iPad. Gade4all, tal y como comentamos anteriormente, se centra más en tecnología móvil y el que será el eventual lenguaje de los videojuegos en Internet: HTML5.

[3] Multi-plataforma. Todo software o dispositivo capaz de poder utilizarse en diferentes plataformas

- Game Editor también ofrece como salida final un ejecutable, en vez de un proyecto completo como Gade4All.

- Game Editor ofrece la posibilidad de añadir código desde el propio editor. En Gade4All este aspecto se debería de hacer una vez se cuente con el proyecto final.

- Los videojuegos finales de Game Editor contarán con una sucesión de niveles sin pantalla de bienvenida ni menús de configuración mientras que Gade4All permitirá crearlos, personalizarlos y añadirlos al videojuego.

2.3.2.3 Game Maker Studio

GameMaker Studio [4]o GM de modo abreviado es considerado un entorno de desarrollo programado en Delphi que fue desarrollado por Mark Overmars. En la actualidad es desarrollado por la empresa Yoyo Games. Esta gran aplicación que aligera en enorme medida la programación de software de entretenimiento pues se utiliza incluso para generar prototipos de futuros videojuegos.

Utiliza una interfaz gráfica muy intuitiva mediante el sistema *drag&drop*. En su web incluso se cita la posibilidad de usarla como herramienta de aprendizaje de realización de videojuegos a niños.

Para usuarios avanzados ofrece una alternativa al sistema de arrastrar y soltar. Este pasa por emplear un lenguaje de *scripting* propio llamado *Game Marker Language* orientado a objetos y que permite personalizar más si cabe el videojuego final.

[4] http://www.yoyogames.com/gamemaker/studio

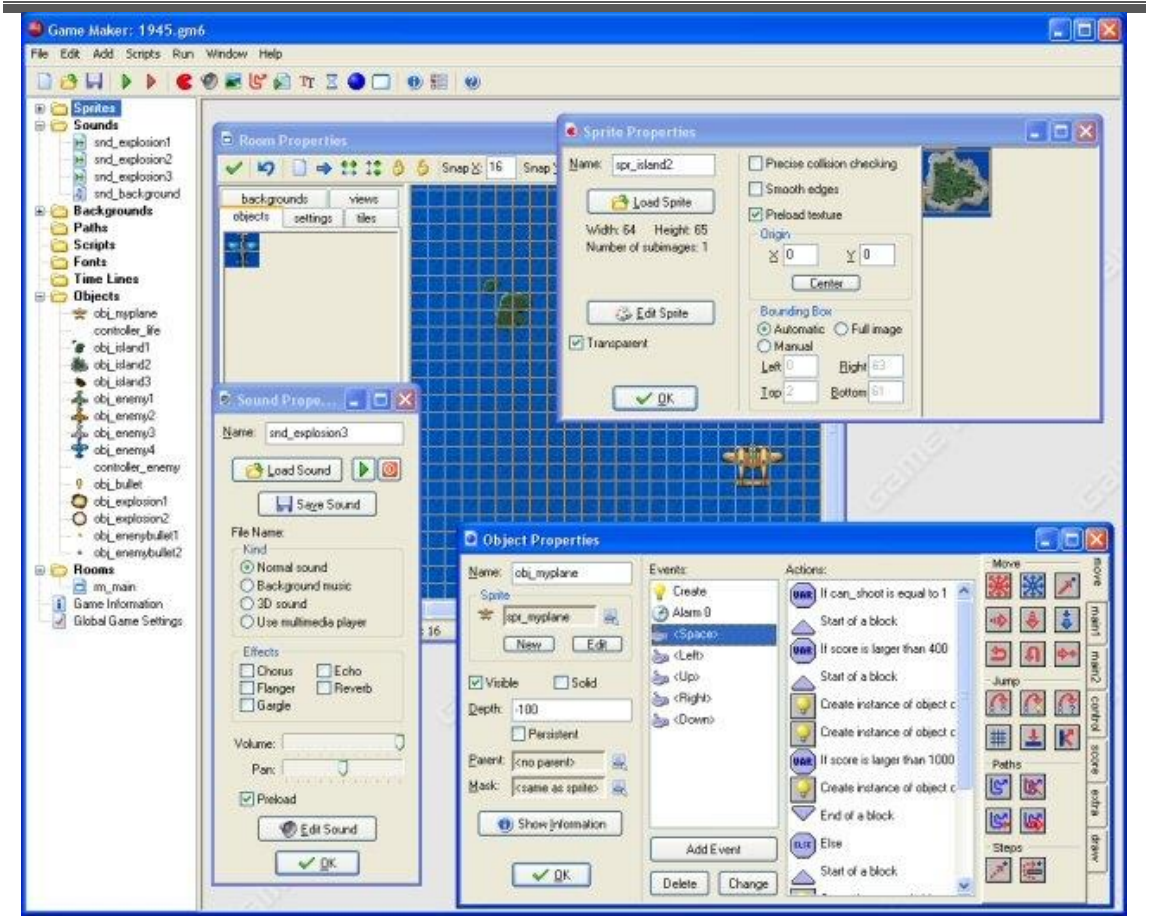


Ilustración 3. Interfaz de definición de videojuegos del GameMaker 8

2.3.2.3.1 Similitudes

- Ambos ofrecen una amplia variedad de tipología de videojuegos que se pueden crear. El catálogo de juegos posibles en Gade4All aún no está cerrado pero actualmente GameMaker también ofrece la posibilidad de generar *shooters*, juegos de estrategia, de acción, *arcade*, etc.

- Al ofrecer posibilidad de usar la interfaz gráfica o la interfaz textual se asemeja a Gade4All en tanto que las dos pueden servir como herramienta de generación de videojuegos tanto para usuarios inexpertos como para usuarios avanzados en programación.

2.3.2.3.2 Diferencias

- Ambas aplicaciones exportan a múltiples plataformas aunque GameMaker también incluye la posibilidad de hacer videojuegos para Facebook, IOS, Mac, Windows 8, Chrome, etc.

- GameMaker ofrece la posibilidad de implementar juegos en 3D. Gade4All se centrará exclusivamente en tecnología 2-D.

- La comunidad de usuarios es muy amplia e incluso existe un repositorio tipo Google Play donde descargarse juegos creados mediante la herramienta (http://sandbox.yoyogames.com/) Gade4All podría contar con uno cuando su uso se extienda pero no es uno de los objetivos del proyecto a corto plazo.

Capítulo 3. Aspectos Teóricos

3.1 Ingeniería dirigida por modelos

La ingeniería dirigida por modelos, más conocida como MDE, las siglas en inglés para *Model Driven Engineering* es una metodología de desarrollo que se centra en aumentar el nivel de abstracción en las especificaciones de los programas e incrementar la automatización del desarrollo software. La idea que propone MDE es utilizar modelos (puntos de referencia teóricos) en diferentes grados de abstracción y, a partir de ellos, generar programas de más bajo nivel que cumplan las características buscadas.

La ingeniería dirigida por modelos surge a la par que los programas van aumentando su complejidad y aparece la necesidad de industrializar el desarrollo software. Sus objetivos se centran, por tanto, en reducir la complejidad de las plataformas actuales generando familias de aplicaciones que posean diversas características en común. De esta manera, se evita codificar las mismas soluciones una y otra vez y se fomenta la reutilización de la arquitectura, lenguajes de dominio específico y código.

Otro concepto que aparece dentro de la MDE es el de meta modelo. A modo de ejemplo para entender qué es este concepto se podría pensar en un edificio, del cual se pueden crear modelos mediante diferentes técnicas, por ejemplo una maqueta, un plano, etc. Esos modelos representan por tanto un elemento del mundo real, en el ejemplo, un edificio. Normalmente esas dos ideas serían suficientes para entender el sistema pero desde el punto de vista del MDE es necesario introducir el concepto de meta modelo como nivel de abstracción superior que aporta los bloques básicos de construcción para crear los propios modelos.

Ya en 1984, Kotteman & Konsynski abogaron por la necesidad de cuatro niveles de instanciación para integrar el modelado en la evolución de los sistemas informáticos. De hecho, MDE está basado en la arquitectura de cuatro capas definida por el *Object Management Group* (OMG) que se muestra a continuación.

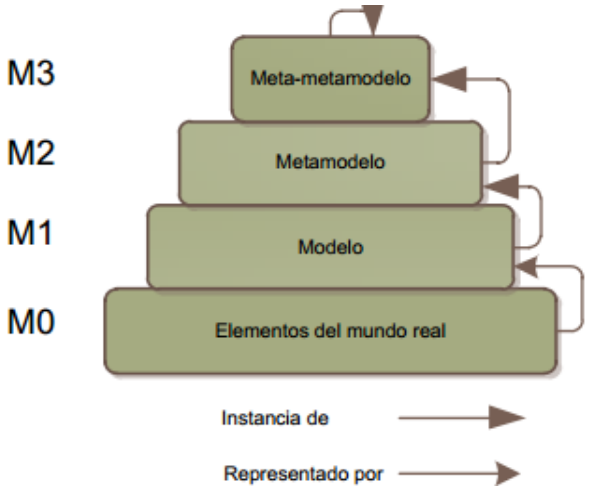


Ilustración 4. Arquitectura de cuatro capas en MDE

El presente proyecto sigue la ingeniería dirigida por modelos en tanto que se generará un meta modelo para la creación de videojuegos de vehículos en plataformas. De esta manera, cada videojuego diferente creado será un modelo generado a partir de la abstracción del meta modelo cumpliendo la reutilización de arquitectura y código y siendo generado a partir de un único DSL.

3.2 Lenguaje de dominio específico

DSL o *Domain Specific Language* es el nombre en inglés para lenguaje de dominio específico, un tipo de lenguaje de especificación o de programación orientado a resolver un problema concreto. Un ejemplo de DSL ampliamente reconocido es el HTML, pues se centra en el marcado de etiquetas para crear páginas web. En el lado contrario se encuentran los lenguajes de programación de propósito general como Java o C++ que sirven para programar cualquier tipo de aplicación sin centrarse en ninguna específicamente.

Los lenguajes de dominio específico se relacionan con la ingeniería dirigida por modelos en tanto que aportan el nivel de abstracción más alto dentro del dominio del problema. En concreto dentro de este proyecto, el DSL rebautizado como GSL (Game Specific Language), definirá los elementos comunes de los videojuegos (jugador principal, fondos, imágenes, sonidos, eventos, etc.) para que a partir de ese meta modelo genérico se puedan generar diferentes videojuegos-modelo personalizando esos elementos.

Antes de comenzar el presente proyecto, el GSL propio de Gade4All es un lenguaje muy amplio para abarcar las diferentes tipologías de juegos ya existentes. La intención del proyecto no será crear un DSL de cero sino aprovechar el que ya posee el proyecto Gade4All y utilizar sólo aquellos elementos definidos que se utilizarán en la plataforma de vehículos en plataformas.

3.3 Juegos basados en *Tiles*

Existen varias maneras de programar la lógica y los diferentes aspectos de un videojuego, desde sistemas de colisiones basados en *Pixel-perfect* hasta la perspectiva que se quiere presentar (2-D, 3-D, perspectiva en primera persona, proyecciones paralelas, etc.)

En nuestro caso, al tratarse de un videojuego de plataformas se ha usado la perspectiva 2-D y una tecnología de *Tiles* para la representación visual de los elementos. Este tipo de motor de juego se basa en dividir el área de la pantalla en pequeños rectángulos, cuadrados o hexágonos. En este caso se ha optado por *Tiles* rectángulos de 40x32 píxeles.

La ventaja principal de los motores de juego basados en *Tiles* es que permite a los desarrolladores crear estructuras dentro del juego más grandes y más complejas de manera sencilla, lo cual es una gran ventaja para que a posteriori el usuario del proyecto pueda definir los escenarios fácilmente. Otro de sus puntos fuertes es su alta eficiencia a la hora de detectar colisiones pues sólo es necesario monitorizar la posición del jugador principal y el tipo de los *Tiles* que rodean al mismo para programar la lógica del videojuego.

No en vano esta tecnología ha sido usada en videojuegos tan conocidos como PacMan, SimCity o las primeras entregas de Pokemon para GameBoy.

Ilustración 5. Videojuego basado en Tiles rectangulares

3.4 Android

Android es un sistema operativo basado en Linux diseñado para dispositivos táctiles, principalmente móviles y *tablets*. Fue inicialmente desarrollado por Android Inc. y comprado por Google en 2005. El primer dispositivo móvil en utilizarlo se vendió en Octubre de 2008.

El sistema operativo Android es de código abierto por lo que su software puede ser modificado y distribuido libremente lo que ha fomentado una amplia comunidad de desarrolladores de aplicaciones que han extendido la funcionalidad de los dispositivos Android. Estas aplicaciones (*apps*) se escriben en una versión personalizada de lenguaje de programación Java y pueden descargarse de *Google Play,* repositorio que contaba a finales de 2012 con más de 700.000.

Ilustración 6. Logotipo de Android

Uno de los motivos principales de centrar el desarrollo del videojuego para plataformas Android es el hecho de que a finales de 2012 Android ocupaba el 75% del mercado mundial de *smartphones,* lo cual, unido a las altas expectativas de mayor crecimiento, fomentará que nuestra aplicación sea utilizada por un mayor número de usuarios comparado con el resto de plataformas.

Capítulo 4. Planificación del Proyecto y Resumen de Presupuestos

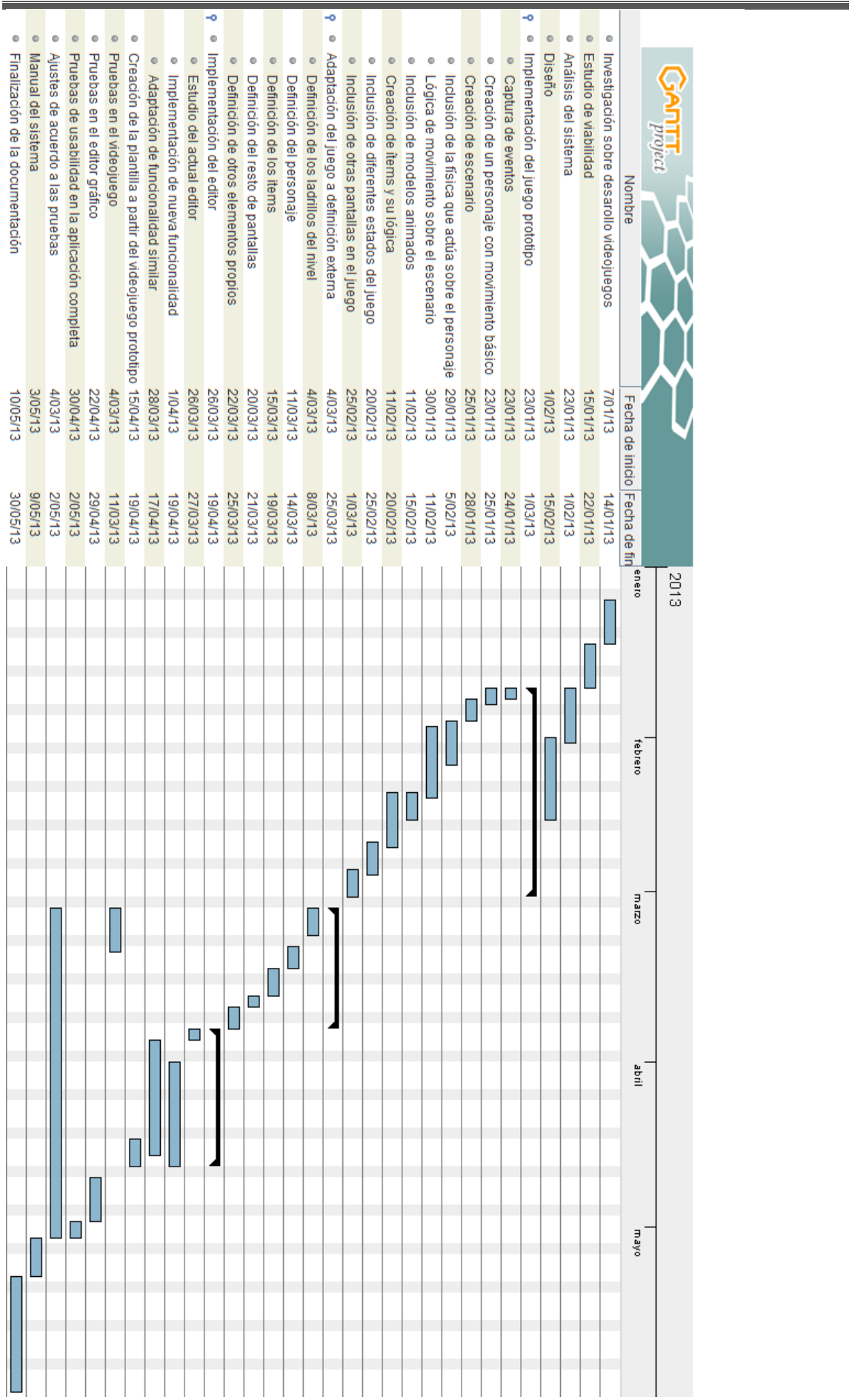
4.1 Planificación

En esta sección se detalla la planificación y el cronograma de actividades del presente proyecto. Su fecha de inicio se establece en el día 7 de enero de 2013 y su fecha estimada de finalización es a principios de junio del mismo año.

La dedicación diaria fue de cinco a seis horas diarias aunque este porcentaje se vio reducido durante el mes de prácticas en empresa (mediados de marzo a mediados de abril) y aumentó en las últimas fases del proyecto para poder cumplir con la planificación definida inicialmente.

Ya que la aplicación completa está formada por tres módulos principales: videojuego, analizador y editor gráfico, se dividirá la parte de implementación y pruebas de cada módulo, no así el resto de etapas que se efectuarán de manera conjunta tratando la aplicación como un todo.

A continuación se añade un diagrama de planificación de Gantt con la duración aproximada de las tareas.


GANTT project
Nombre
Fecha de inicio
Fecha de fin
2013
enero
febrero
marzo
abril
mayo
Investigación sobre desarollo videojuegos 7/01/13 14/01/13
Estudio de viabilidad 15/01/13 22/01/13
Análisis del sistema 23/01/13 1/02/13
Diseño 1/02/13 15/02/13
Implementación del juego prototipo 23/01/13 1/03/13
Captura de eventos 23/01/13 24/01/13
Creación de un personaje con movimiento básico 23/01/13 25/01/13
Creación de escenario 25/01/13 28/01/13
Inclusión de la física que actúa sobre el personaje 29/01/13 5/02/13
Lógica de movimiento sobre el escenario 30/01/13 11/02/13
Inclusión de modelos animados 11/02/13 15/02/13
Creación de ítems y su lógica 11/02/13 20/02/13
Inclusión de diferentes estados del juego 20/02/13 25/02/13
Inclusión de otras pantallas en el juego 25/02/13 1/03/13
Adaptación del juego a definición externa 4/03/13 25/03/13
Definición de los ladrillos del nivel 4/03/13 8/03/13
Definición del personaje 11/03/13 14/03/13
Definición de los ítems 15/03/13 19/03/13
Definición del resto de pantallas 20/03/13 21/03/13
Definición de otros elementos propios 22/03/13 25/03/13
Implementación del editor 26/03/13 19/04/13
Estudio del actual editor 26/03/13 27/03/13
Implementación de nueva funcionalidad 1/04/13 19/04/13
Adaptación de funcionalidad similar 28/03/13 17/04/13
Creación de la plantilla a partir del videojuego prototipo 15/04/13 19/04/13
Pruebas en el videojuego 4/03/13 11/03/13
Pruebas en el editor gráfico 22/04/13 29/04/13
Pruebas de usabilidad en la aplicación completa 30/04/13 2/05/13
Ajustes de acuerdo a las pruebas 4/03/13 2/05/13
Manual del sistema 3/05/13 9/05/13
Finalización de la documentación 10/05/13 30/05/13

4.2 Resumen del Presupuesto

Esta sección trata de estimar un cómputo presupuestario global de todos los recursos utilizados durante la elaboración de este proyecto. Conviene destacar que este presupuesto se centra únicamente en la inversión del módulo de un generador de videojuegos de vehículos y su correspondiente documentación, no en el proyecto general de Gade4All. Para el cálculo de los recursos humanos se ha hecho una estimación de los salarios medios de un ingeniero superior informático en España y para el cálculo de los recursos de ordenador y dispositivo móvil se ha elegido un factor de amortización de 0,40 € la hora.

Item	Concepto	Cantidad	Precio unitario / Amortización	TOTAL
1	Ordenador	1	(0,40 € * 900 h)	360,00 €
2	Dispositivo móvil Android	1	(0,40 € * 70 h)	28,00 €
3	Microsoft Windows 7	1	125 €	125,00 €
4	Visual Studio 2012 Profesional	1	675 €	675,00 €
5	Formación	20h	15€ / hora	300,00 €
6	Evaluación de viabilidad, análisis y diseño	100h	15€ / hora	1.500,00 €
7	Implementación	700h	10 € / hora	7.000,00 €
8	Pruebas	80h	10 € / hora	800,00€
			Subtotal	10.788,00 €
			IVA (21%)	2265,48 €
			TOTAL	**13053,48 €**

Capítulo 5. Análisis

Este apartado contiene toda la especificación de requisitos y toda la documentación del análisis de la aplicación, a partir de la cual se elaborará posteriormente el diseño.

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

En apartados anteriores se ha definido de manera general los objetivos y la funcionalidad del proyecto. Sin embargo, la parte principal del mismo consiste en desarrollar un videojuego basado en el control de vehículos por lo que este apartado se dedicará a estudiar las características de otros videojuegos populares en el mercado para Android en los cuales se basará la aplicación a desarrollar.

5.1.1.1 Moto X Mayhem

Personaje principal: Moto

Controles:

- Zona derecha de la pantalla para acelerar.
- Zona izquierda de la pantalla para frenar.
- Acelerómetro para controlar la inclinación hacia la derecha o izquierda.

¿Ofrece niveles de aprendizaje?: Sí.

Forma de puntuación: Los niveles se agrupan en etapas. La puntuación récord se obtiene sumando el tiempo empleado en completar todas las pantallas de la etapa. Además, la aplicación permite subir el resultado a Internet.

Forma de progresar en el juego: Se deben superar los obstáculos y llegar a la meta. Sólo se puede acceder a un nivel habiendo superado el anterior.

Forma de perder en el juego: El motorista se choca con alguna parte de su cuerpo contra el suelo, el techo o algún obstáculo.

Elementos que se utilizaran en el videojuego del proyecto:

- Se utilizarán controles táctiles para acelerar y para frenar.
- Para pasar de nivel el personaje principal tendrá que llegar a la meta superando los obstáculos y los récords se conseguirán batiendo el tiempo empleado en superar un nivel.

Ilustración 7. Captura de pantalla del videojuego Moto X Mayhem

5.1.1.2 Skater Boy

Personaje principal: Monopatín

Controles:

- Botón de acelerar.
- Botón de saltar / cambiar la posición del *skate.*

¿Ofrece niveles de aprendizaje?: No.

Forma de puntuación: Dentro de cada nivel se pueden obtener tres tipos de premio: obtener todos los trofeos, puntuación alta (ya sea recolectando estrellas o haciendo trucos con el skate) y llegar al final sin haber perdido ninguna vida.

Forma de progresar en el juego:

Cada vez que se llega a la meta de un nivel (aunque no se haya conseguido ninguno de los tres premios) se desbloquea el nivel siguiente.

- Para desbloquear un mundo (conjunto de varios niveles) se debe tener un mínimo de premios por lo que no basta llegar al final de los niveles sin conseguir puntos, trofeos, etc.

Forma de perder en el juego:

- El patinador se choca con alguno de los obstáculos del nivel.
- Si el patinador colisiona aparece en el último punto de control hasta un máximo de cinco veces aunque ya no obtendrá el premio de acabar el nivel sin muertes.

Elementos que se utilizaran en el videojuego del proyecto:

- Se utilizarán controles en la parte inferior de la pantalla para controlar el movimiento del personaje principal.
- Se utilizarán puntos de control en los que el personaje reaparecerá si pierde en un punto intermedio del nivel.
- El jugador tendrá un número finito de vidas para llegar a la meta del nivel.
- El videojuego poseerá varias maneras de pasar de nivel a definir por el usuario.
- Las pendientes utilizadas para definir los niveles serán rectas.

Ilustración 8. Captura de pantalla del videojuego Skater Boy

5.1.1.3 Dragon Fly!

Personaje principal: Dragón

Controles:

- Pulsar en la pantalla para coger inercia en los valles.

¿Ofrece niveles de aprendizaje?: Sí.

Forma de puntuación:

Forma de progresar en el juego: Se deben cumplir una serie de objetivos que aparecen al comienzo del juego (recolectar un número de diamantes, llegar a un nivel, hacer un número concreto de aterrizajes “perfectos”, etc.) y que cambian cada día.

Forma de perder en el juego: Como el juego se basa en la habilidad del usuario para avanzar rápido por el nivel, en el momento en el que tome demasiado tiempo para completarlo aparecerá "Mama dragón" y se habrá acabado el juego.

Elementos que se utilizaran en el videojuego del proyecto:

- Para aumentar las posibilidades de personalizar el videojuego el personaje principal no será exclusivamente un vehículo, sino cualquier personaje que defina el usuario.

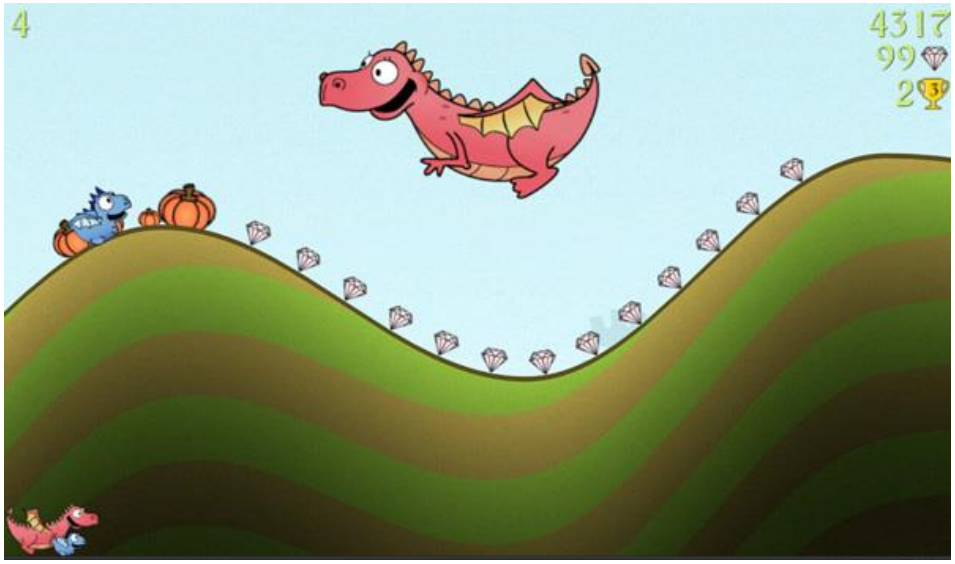


Ilustración 9. Captura de pantalla del videojuego Dragon Fly!

5.1.1.4 Alcance del videojuego a desarrollar

Teniendo en cuenta las características particulares de los videojuegos anteriores y otros que han sido objeto de estudio (Hill Climb Racing, Stickman Ski Racer, Stuntcar Challenge y Stunt2) se han extraído las siguientes características para el videojuego basado en el control de vehículos del presente proyecto. Son las siguientes:

- La interfaz fija del juego contará con dos botones de interacción con el usuario: botón de freno en la parte izquierda y botón de acelerar en la parte derecha. Asimismo, se incluirá un botón de pausa para parar momentáneamente el desarrollo del juego y un botón de reinicio en el caso de que el personaje principal se encuentre en un punto del nivel en el que no pueda avanzar más o no pueda alcanzar la meta. Se ha decidido no utilizar el acelerómetro para controlar el movimiento del personaje principal en el caso de una eventual exportación a HTML5. De esta manera los videojuegos generados tanto para dispositivos móviles como para ordenadores personales serán muy similares.

- A la hora de definir el videojuego, el usuario podrá elegir el objetivo a cumplir para pasar de nivel: llegar a la meta con un número concreto de vidas, llegar a la meta antes de un tiempo predefinido o una combinación de las anteriores.

- El usuario definirá los niveles disponiendo las superficies por las que avanzará el personaje principal, las rampas de diferentes inclinaciones, las trampas, los puntos de control en los que el usuario reaparecerá y la meta. También podrá elegir el personaje principal y personalizar la física que gobierna el juego (gravedad, índices de fricción, inercias, etc.) además de los sonidos del juego.

- La física se implementará desde cero por lo que no se utilizarán motores específicos para ello como AndEngine o jBox2D. Aunque la mayoría de los videojuegos del mercado actuales optan por el uso de algún *framework,* el objetivo del proyecto es aprender a desarrollar las diferentes facetas de un videojuego por lo que se optará por dicha opción para tener un mejor control del código existente y en aras de un mayor rango de aprendizaje.

5.2 Requisitos del Sistema

5.2.1 Obtención de los Requisitos del Sistema

5.2.1.1 Requisitos funcionales

5.2.1.1.1 Gestión de videojuegos

Código	Nombre Requisito	Descripción del Requisito
R1.1.1	Crear juego	La aplicación para generar el videojuego debe tener una opción para crear uno nuevo desde cero.
R1.1.2	Guardar juego	La aplicación debe contar con la opción de guardar un juego creado.
R1.1.3	Abrir juego	La aplicación debe tener la opción de abrir un juego previamente guardado.
R1.1.4	Exportar juego	La aplicación debe contar con la opción de exportar el juego a la plataforma Android.

5.2.1.1.2 Generación del videojuego

Código	Nombre Requisito	Descripción del Requisito
R1.2.1	Crear escenario	El usuario debe ser capaz de crear los niveles del videojuego disponiendo los diferentes elementos por el escenario.
R1.2.2	Asignar gráficos	El sistema debe permitir asignar gráficos a los diferentes elementos del juego: personaje principal, *Tiles,* botones, fondo, puntos de control, Trampas, etc.
R1.2.3	Asignar constantes físicas	El sistema debe permitir cambiar algunos parámetros físicos del juego como la gravedad, los factores de fricción, etc.

R1.2.4	Asignar condición fin de nivel	El usuario debe ser capaz de escoger la condición de fin de nivel: llegar a la meta con un número a escoger de vidas o dentro de un límite de tiempo.
R1.2.5	Asignar recursos sonoros	El sistema debe permitir asignar sonidos con determinados eventos importantes en el transcurso del videojuego.

5.2.1.1.3 Requisitos de exportación

Código	Nombre Requisito	Descripción del Requisito
R1.3.1	Exportación a DSL	El resultado de exportar el videojuego una vez creado será la personalización del DSL con los parámetros que introdujo el usuario. El formato de exportación será XML.
R1.3.2	Producto resultado de la exportación	Una vez se procese el XML, se deberá obtener un proyecto Android con un videojuego de plataformas basado en el control de vehículos totalmente funcional y con todos los elementos que definió el usuario en el editor.

5.2.1.2 Requisitos no funcionales

5.2.1.2.1 Requisitos tecnológicos

Código	**Nombre Requisito**	**Descripción del Requisito**
R2.1.1	Sistema operativo del editor	Es necesario el uso del sistema operativo Windows para la creación, edición y exportación de videojuegos.
R2.1.2	Ejecución de los videojuegos	Para probar los videojuegos desarrollados se necesitará un dispositivo móvil o tablet con el sistema operativo Android o el emulador que contiene el propio Eclipse.
R2.1.3	IDE para el videojuego generado	Para trabajar con el proyecto generado y/o modificar el resultado se precisa del entorno Eclipse además de las Android SDK para probar la aplicación en un emulador.

5.2.1.2.2 Requisitos del sistema

Código	**Nombre Requisito**	**Descripción del Requisito**
R2.2.1	Procesador	Debe utilizarse una CPU con procesador Pentium IV a 3 GHz o superior.
R2.2.2	Memoria RAM	Se requiere al menos 1GB de RAM en equipos de 32 bits o 2GB en uno de 64 bits.
R.2.2.3	Disco duro	Es recomendable disponer de un disco duro con al menos 3GB libres.
R.2.2.4	Tarjeta Gráfica	Es necesario poseer una tarjeta gráfica compatible con DirectX 9 o superior.

5.2.1.2.3 Requisitos de usabilidad

Código	Nombre Requisito	Descripción del Requisito
R2.3.1	Interfaz intuitiva	La interfaz de edición de videojuegos debe ser intuitiva y fácil de utilizar para que el usuario pueda definir un videojuego sin emplear tiempo en comprender cómo funciona el editor.
R2.3.2	Conocimiento sobre la implementación de videojuegos o sobre el entorno de desarrollo	El usuario debe ser capaz de generar un videojuego aunque no posea conocimientos específicos sobre el desarrollo de videojuegos para Android o sobre el IDE Eclipse.

5.2.1.2.4 Requisitos de tiempo de respuesta

Código	Nombre Requisito	Descripción del Requisito
R2.4.1	Tiempo de respuesta del videojuego	El videojuego generado deberá actualizar su lógica de manera que éste funcione de manera fluida y sin ralentizaciones en cualquier dispositivo Android.

5.2.2 Identificación de Actores del Sistema

- Usuario diseñador
- Usuario jugador

5.2.2.1 Usuario diseñador

Es el actor que utilizará la herramienta para crear, editar y exportar los videojuegos basados en el control de vehículos. Dichas tareas podrán llevarse a cabo a través del editor o directamente sobre el fichero DSL. El usuario diseñador de videojuegos no se le presupone ningún tipo de conocimiento sobre el desarrollo de videojuegos para dispositivos móviles

5.2.2.2 Usuario jugador

Es el actor que juega a los videojuegos definidos por el usuario diseñador. Podrá hacerlo a través de un dispositivo Android o a través del emulador del SDK. Tampoco es necesario que tenga conocimientos sobre el desarrollo de videojuegos o sobre el entorno en el cual se exportará el juego desarrollado.

5.2.3 Especificación de Casos de Uso

A lo largo de esta sección se estudiarán los casos de uso de la aplicación que involucran a los actores anteriormente citados.

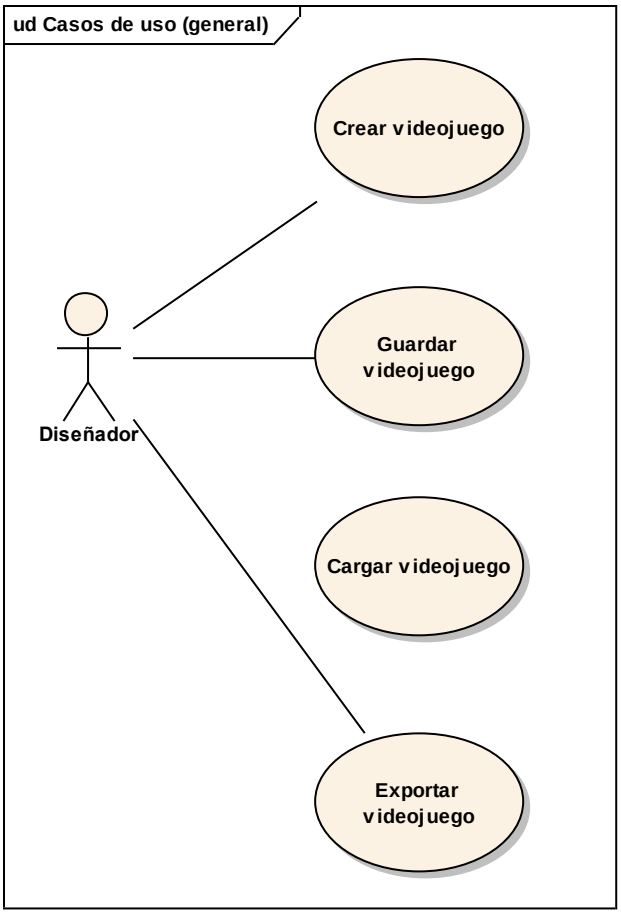


Ilustración 10. Diagrama general de los casos de uso del usuario diseñador

5.2.3.1 Crear videojuego

Nombre del Caso de Uso
Crear videojuego
Descripción
El usuario diseñador podrá crear un nuevo videojuego a través del editor gráfico correspondiente. Una vez que el proyecto Gade4All se haya cerrado, dicho usuario escogerá en primera instancia qué tipo de videojuego quiere diseñar. En el caso concreto de este proyecto escogería un videojuego de plataformas basado en el control de vehículos. El diseño del videojuego se basa en la elección y configuración de un variado número de elementos. Por ello se ha considerado conveniente dividir este caso de uso en los subcasos que se incluyen a continuación.

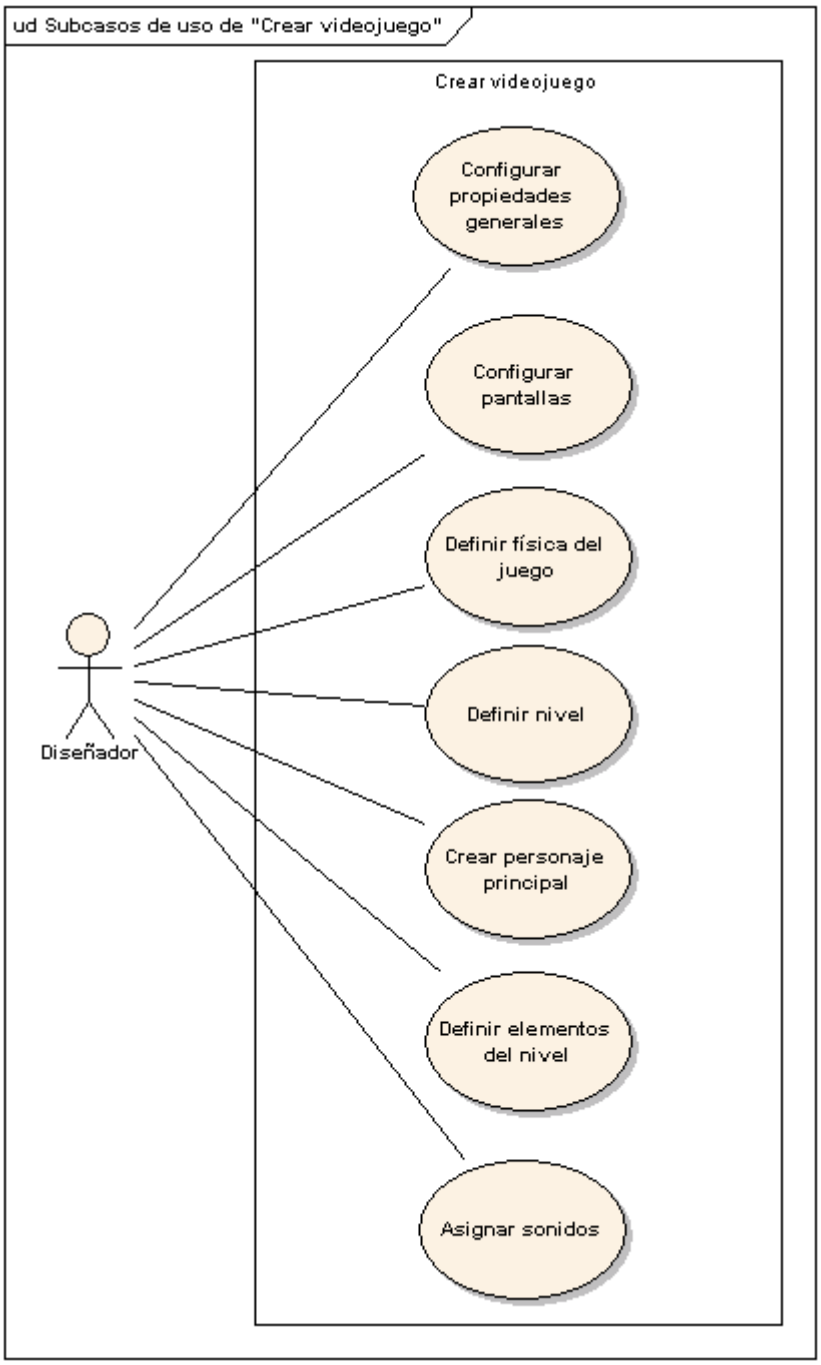


Ilustración 11. Subcasos de uso de "Crear videojuego"

5.2.3.1.1 Subcaso de uso: Configurar propiedades generales

Nombre del Caso de Uso
Configurar propiedades generales
Descripción
El usuario diseñador podrá configurar las propiedades generales del juego: tipo de juego, nombre, icono de la aplicación, posibilidad de incluir publicidad y música de los menús.

5.2.3.1.2 Subcaso de uso: Configurar pantallas

Nombre del Caso de Uso
Configurar pantallas
Descripción
El editor permitirá configurar el fondo, la apariencia y los botones (en el caso de que fueran necesarios) de las diferentes pantallas de la aplicación Android. El usuario diseñador podrá definir, por tanto, la pantalla principal, la pantalla de selección de nivel, la pantalla de opciones, la pantalla de pausa, la pantalla de victoria al finalizar un nivel, la pantalla de fin de juego y la pantalla de derrota en un nivel. Dentro de la pantalla propia del videojuego podrá seleccionar el botón de reinicio, la posición y color del marcador de tiempo y la forma de representar las vidas que le quedan al personaje principal.

5.2.3.1.3 Subcaso de uso: Definir física del juego

Nombre del Caso de Uso
Definir física del juego
Descripción
El usuario diseñador podrá cambiar algunos factores que influyen en la física del videojuego. El editor sugerirá unos valores predeterminados que harán que el juego posea una física realista pero el usuario podrá modificar los valores de la gravedad, la fricción en cuestas, el factor de aceleración o frenado cuando se pulsan los botones de interacción, la máxima aceleración en ambos sentidos y establecer si el personaje principal siente la acción de la gravedad cuando se sitúa en pendientes.

5.2.3.1.4 Subcaso de uso: Definir nivel

Nombre del Caso de Uso
Definir nivel
Descripción
A través del editor se podrán arrastrar los diferentes ladrillos o *Tiles* para definir la estructura física del nivel, es decir, el camino que podrá recorrer el personaje principal. Se seleccionarán otros parámetros del nivel como la imagen *thumbnail* que lo representa, la imagen de explicación, etc. También se deberá definir el punto de meta, el punto de inicio, la imagen para el fondo y situar el resto de elementos (puntos de control y trampas) en el escenario. Además, el diseñador será capaz de establecer la condición por la que el jugador pasa al siguiente nivel (llegar a la meta dentro de un límite de tiempo a definir, llegar a la meta antes de quedarse sin vidas o una combinación de los anteriores).

5.2.3.1.5 Subcaso de uso: Crear personaje principal

Nombre del Caso de Uso
Crear personaje principal
Descripción
El diseñador podrá elegir el personaje principal de cada nivel definiendo las animaciones (sucesión de imágenes) que se mostrarán ante determinados eventos: cuando el personaje esté parado, cuando se mueva a derecha o izquierda, cuando suba por un desnivel de determinada inclinación, etc. Se definirán también otros parámetros propios del personaje principal tales como tamaño y áreas de colisión.

5.2.3.1.6 Subcaso de uso: Definir elementos del nivel

Nombre del Caso de Uso
Definir puntos de control
Descripción
Para facilitar los niveles se podrán definir puntos de control o *checkpoints* en los cuales el personaje principal reaparecerá después de haber perdido una vida si previamente ha pasado por ese punto. El usuario diseñador podrá definir las animaciones a mostrar cuando el punto de control esté inactivo o activo. Para aumentar la complejidad y la versatilidad de los niveles, el diseñador podrá definir elementos trampa que harán perder vidas al personaje principal cuando éste colisione con ellas. Se podrá elegir su tamaño, la animación a mostrar y su posición dentro del nivel. Para definir la estructura de nivel se utilizarán *Tiles* de tipo sólido y de diferentes inclinaciones. Se podrá seleccionar la imagen y el tamaño de cada *Tile.* También se podrá definir la imagen de los elementos de meta (punto final) de los niveles.

5.2.3.1.7 Subcaso de uso: Asignar sonidos

Nombre del Caso de Uso
Asignar sonidos
Descripción
De manera opcional, se podrán asignar recursos sonoros ante determinados eventos que surjan durante el desarrollo del juego: música ambiente, sonidos de aceleración y freno al pulsar los botones de interacción, sonido al perder una vida, música al finalizar un nivel, etc.

5.2.3.2 Guardar videojuego

Nombre del Caso de Uso
Guardar videojuego
Descripción
El editor permitirá guardar el estado de los elementos que ya han sido definidos en el proceso de creación.

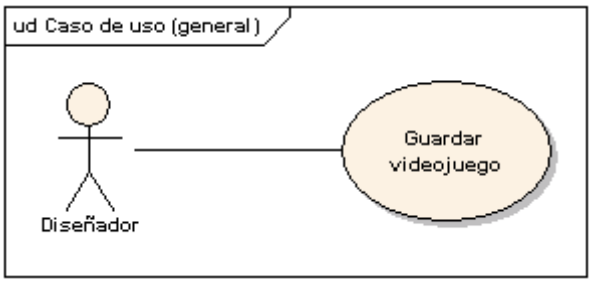


Ilustración 12. Caso de uso "guardar videojuego" del usuario diseñador

5.2.3.3 Cargar videojuego

Nombre del Caso de Uso
Cargar videojuego
Descripción
El editor recuperará el estado de aquellos videojuegos en proceso de creación que hayan sido guardados previamente.

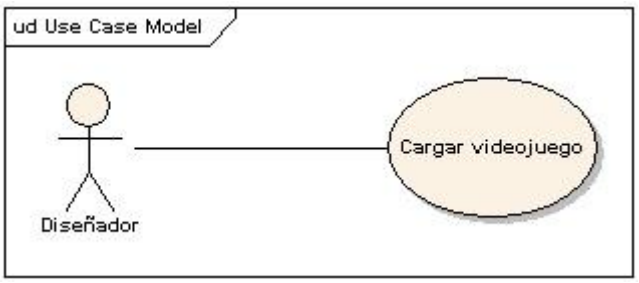


Ilustración 13. Caso de uso "cargar videojuego" del usuario diseñador

5.2.3.4 Exportar videojuego

Nombre del Caso de Uso
Exportar videojuego
Descripción
Una vez se hayan asignado los elementos obligatorios del juego, el usuario podrá exportar dicho videojuego a la plataforma Android, obteniendo como resultado de la operación un proyecto Android completamente funcional que podrá ser posteriormente importado al Eclipse. Una vez compilado dicho proyecto se obtendrá el el archivo .apk que se deberá instalar en el dispositivo para jugar al videojuego.

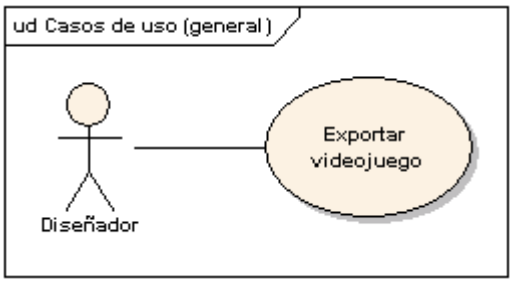


Ilustración 14. Caso de uso "exportar videojuego" del usuario diseñador

5.2.3.5 Jugar a videojuego

Nombre del Caso de Uso
Jugar a videojuego
Descripción
El jugador podrá interaccionar con el videojuego orientado a vehículos creado por el usuario diseñador. Este caso de uso se contempla desde el punto de vista de un usuario jugando desde un dispositivo Android y desde un jugador que lo hace desde el emulador.

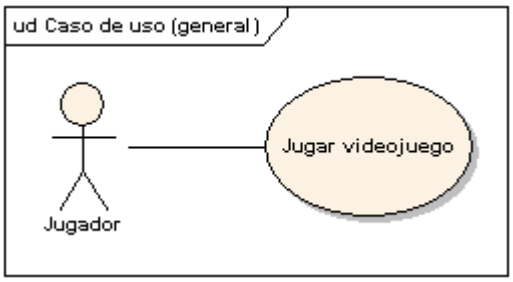


Ilustración 15. Único caso de uso del usuario jugador

5.3 Identificación de los Subsistemas en la Fase de Análisis

El objetivo de esta sección es analizar el sistema para poder descomponerlo en sistemas más pequeños (subsistemas) que faciliten su posterior análisis.

5.3.1 Descripción de los Subsistemas

Los subsistemas que abarcan la total funcionalidad del proyecto son:

- Subsistema editor.
- Subsistema analizador.
- Subsistema videojuego.

5.3.1.1 Subsistema editor

El subsistema editor abarca las tareas relativas a la definición de videojuegos por parte del usuario permitiendo desarrollar actualmente varias tipologías de juegos. Debido a la amplitud de dicho subsistema, el estudio de ahora en adelante se centrará en aquellas clases que deban ser modificadas o implementadas de cero para crear un videojuego de vehículos, obviando el resto que conforman la herramienta que no se utilizarán en el presente proyecto.

5.3.1.2 Subsistema analizador

El subsistema analizador es el encargado de procesar la información generada por el subsistema editor y transformar la instancia DSL resultante en un proyecto concreto de Android sustituyendo los valores en la plantilla del videojuego.

5.3.1.3 Subsistema videojuego

El subsistema videojuego es sobre el que recae toda la lógica del juego orientado a control de vehículos. Desde el punto de vista del analizador es una plantilla que se personalizará con los elementos necesarios de la instancia del DSL que definió el usuario diseñador mediante el editor. Posteriormente será un proyecto importado al Eclipse que incluirá todas las clases necesarias para implementar el videojuego.

5.3.2 Descripción de los Interfaces entre Subsistemas

Los subsistemas anteriormente citados se comunicarán entre sí de forma local por lo que no será necesaria la definición de interfaces.

5.4 Diagrama de Clases Preliminar del Análisis

5.4.1 Diagrama de Paquetes

Dada la amplitud del subsistema editor y analizador, se procede a separar los diagramas de clases de cada uno de los subsistemas. Para entender la interacción entre los tres subsistemas se incluye un diagrama de paquetes de la aplicación completa.

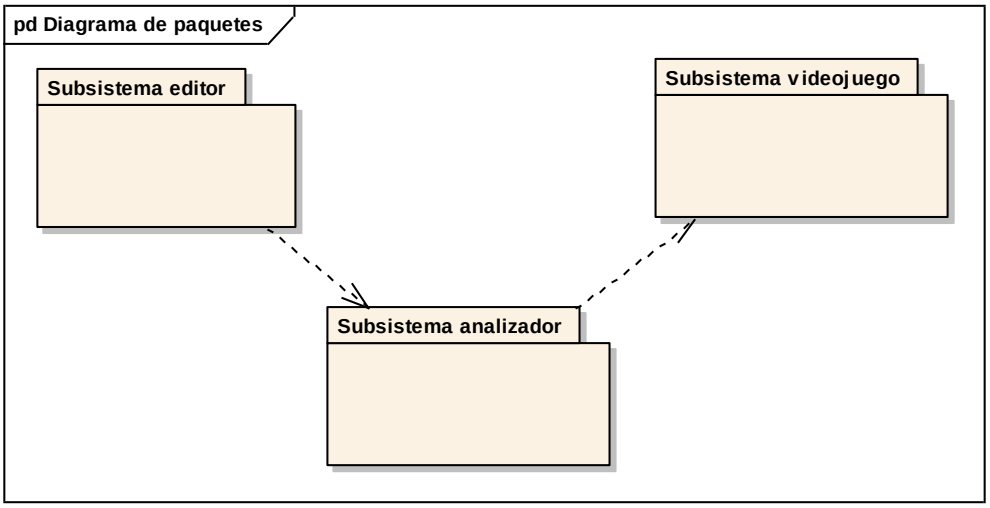


Ilustración 16. Diagrama de paquetes general de la aplicación

5.4.2 Diagramas de Clases

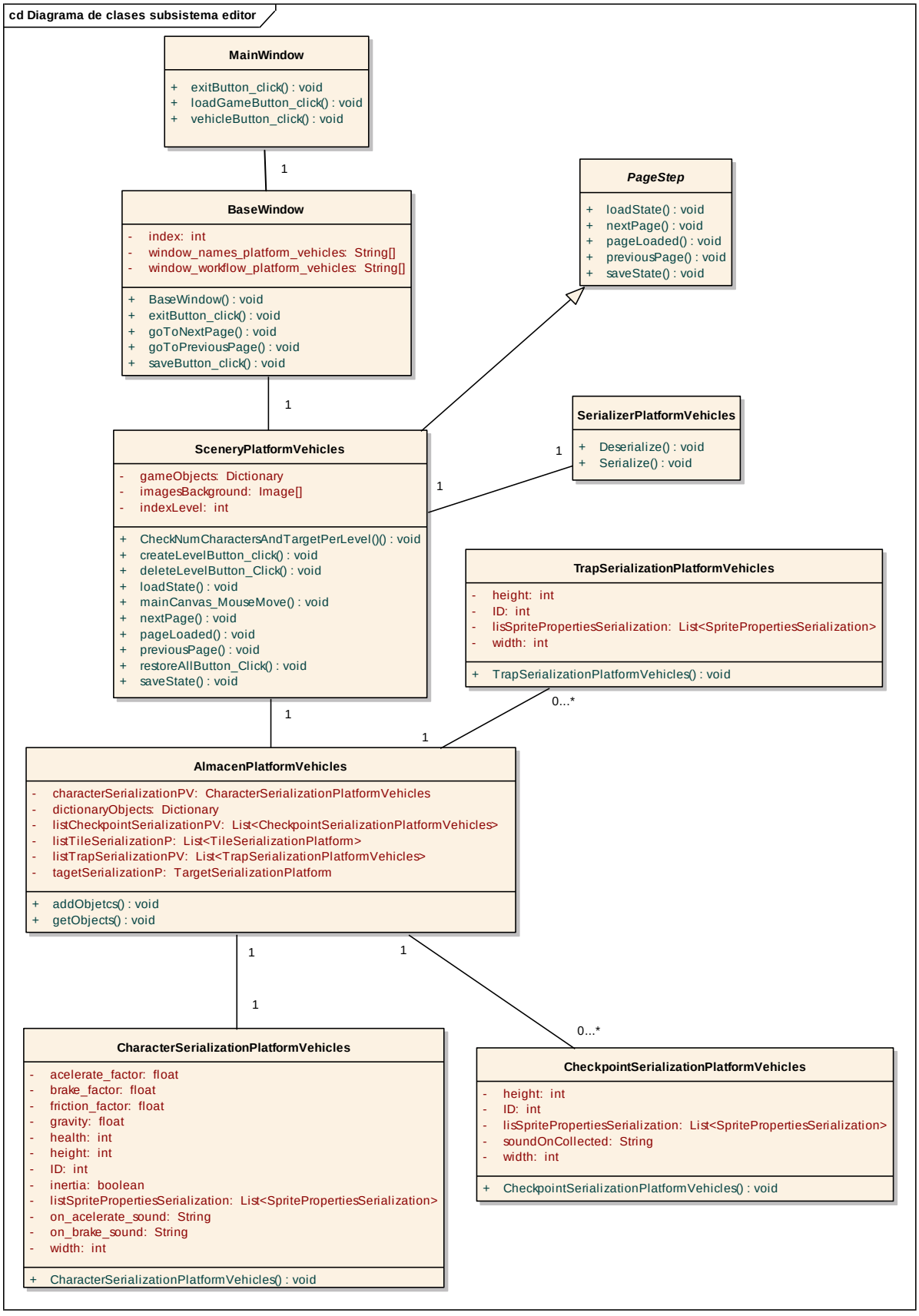


Ilustración 17. Diagrama de clases en la fase de análisis del subsistema editor

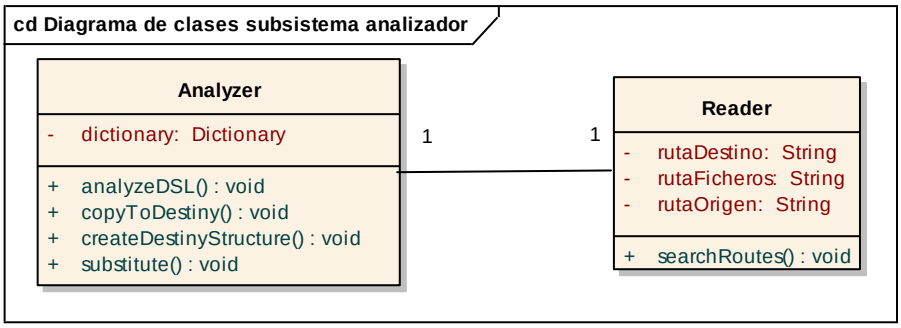


Ilustración 18 . Diagrama de clases en la fase de análisis del subsistema analizador

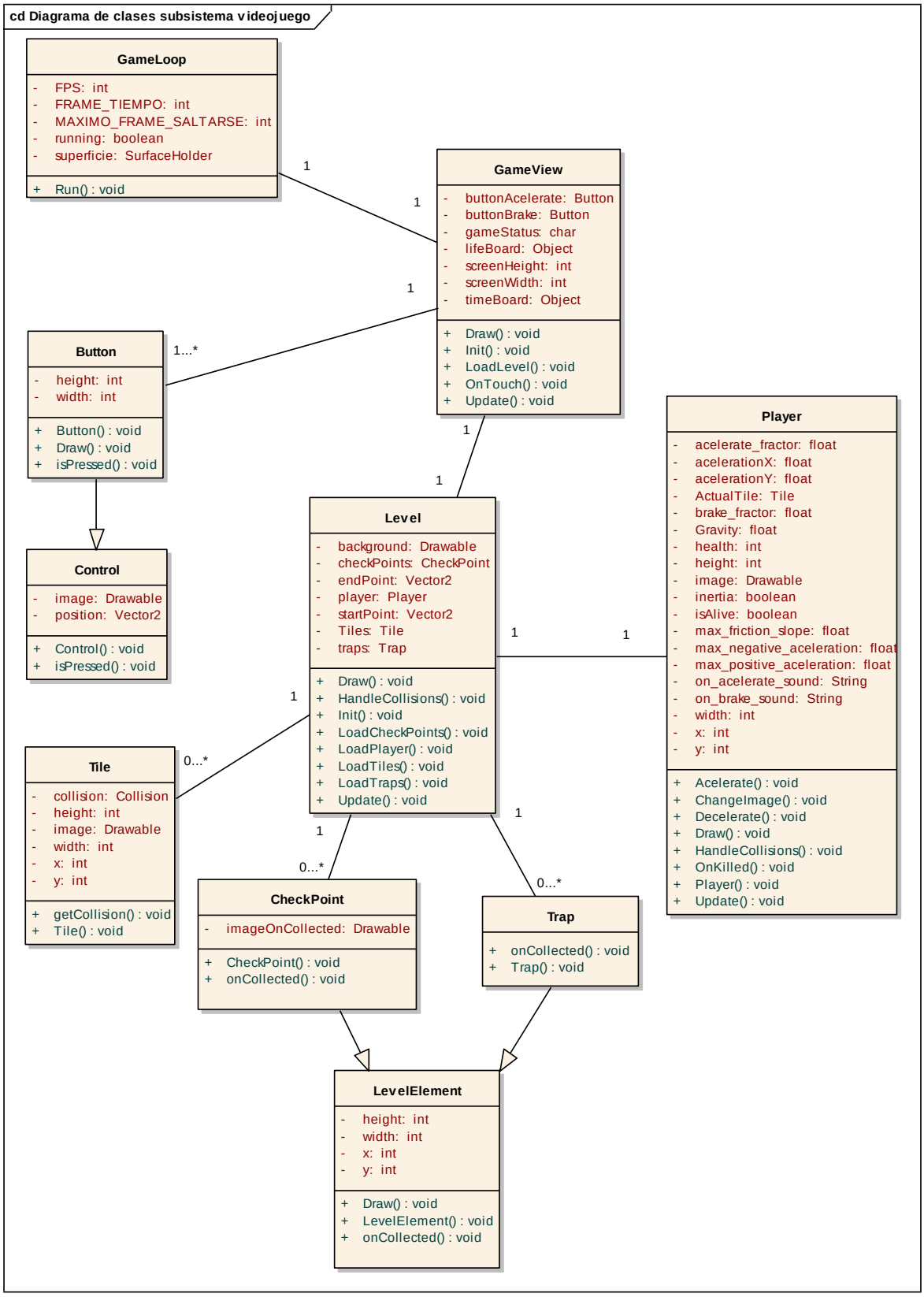


Ilustración 19. Diagrama de clases en la fase de análisis del subsistema videojuego

5.4.3 Descripción de las Clases

5.4.3.1 Subsistema editor

5.4.3.1.1 Clase MainWindow

Nombre de la Clase
MainWindow
Descripción
Alberga la primera pantalla de la aplicación
Responsabilidades
Muestra las diferentes tipologías a elegir con su descripción asignada. Permite elegir una de ellas y guardar el proyecto inicial en una ruta.
Atributos Propuestos
-
Métodos Propuestos
exitButton_click(): cierra la aplicación. **loadGameButton_click():** carga la edición de un videojuego guardado previamente. **vehicleButton_click():** accede a las pantallas de edición de un videojuego basado en el control de vehículos.

5.4.3.1.2 Clase BaseWindow

Nombre de la Clase
BaseWindow
Descripción
Contiene las pantallas de edición de cada tipología de juego
Responsabilidades
Dependiendo del parámetro que le llegue de MainWindow (tipología de juego elegida) carga unas pantallas u otras y permite la navegación entre ellas haciendo de contenedor.
Atributos Propuestos
Index: entero que alberga el número de la pantalla en la cual se encuentra el usuario. **window_names_platform_vehicles:** vector de *strings* que almacena los nombres de cada una de las pantallas de edición para la tipología de juego de control de vehículos. **window_workflow_platform_vehicles:** vector de *strings* que almacena los nombres de cada una de las clases que implementan las pantallas de edición para la tipología de juego de control de vehículos.
Métodos Propuestos
BaseWindow(): constructor de la clase que carga los vectores window_names o window_workflow dependiendo de la tipología de juego. **goToNextPage():** va a la siguiente pantalla respecto a la actual en la edición. **goToPreviousPage():** va a la pantalla anterior respecto a la actual en la edición. **saveButton_click():** guarda el estado de edición de la pantalla en el sistema de ficheros. **exitButton_click():** cierra la aplicación.

5.4.3.1.3 Clase PageStep

Nombre de la Clase
PageStep
Descripción
Interfaz abstracto para definir métodos comunes a las Scenery
Responsabilidades
Esta clase abstracta define diferentes funciones que deben implementar cada uno de los Scenery asociados a cada tipología de juego para conocer en qué estado se encuentra la página.
Atributos Propuestos
-
Métodos Propuestos
nextPage() **pageLoaded()** **previousPage()** **saveState()** **loadState()**

5.4.3.1.4 Clase SceneryPlatformVehicles

Nombre de la Clase
SceneryPlatformVehicles
Descripción
Carga el escenario de definición de la tipología de juegos de control de vehículos
Responsabilidades
Almacena los elementos del juego y dibuja el canvas sobre el cual el usuario definirá los niveles. Permite crear, eliminar, guardar y modificar los niveles.
Atributos Propuestos
gameObjects: diccionario que almacena los elementos del juego. **IndexLevel:** almacena el identificador del nivel actualmente en edición.
Métodos Propuestos
createLevelButton_click(): crea un nuevo nivel a editar. **deleteLevelButton_click():** elimina el nivel en edición. **restoreAllButton_click():** pone los parámetros por defecto de configuración de nivel. **CheckMandatoryElementsInLevel():** comprueba si el nivel tiene al menos un personaje principal, un punto de salida y un punto de meta. **Canvas_mouseMove():** permite arrastrar y soltar elementos en el canvas de edición de un nivel, modificar su posición, su tamaño, etc. **nextPage():** avanza a la página siguiente. Implementa la clase de PageStep. **pageLoaded():** llama a loadState y detecta cuando se ha cargado por completo la página. Implementa la clase de PageStep. **previousPage():** va a la página anterior. Implementa la clase de PageStep. **saveState():** guarda el estado de edición. Implementa la clase de PageStep. **loadState():** cargar un estado de edición guardado previamente. Implementa la clase de PageStep.

5.4.3.1.5 Clase AlmacenPlatformVehicles

Nombre de la Clase
AlmacenPlatformVehicles
Descripción
Clase con el atributo serializable que almacena los elementos del juego de control de vehículos.
Responsabilidades
Servir como contenedor de todos los elementos del juego a la hora de serializar y deserializar el estado de una edición.
Atributos Propuestos
dictionaryObjects: diccionario que almacena todos los elementos simples del juego de control de vehículos. **characterSerializationPV:** objeto serializable del personaje principal. **listTrapSerializationPV:** lista que contiene las trampas del nivel para que puedan ser serializadas. **listCheckPointSerializationPV:** lista que contiene los puntos de control del nivel para que puedan ser serializadas. **targetSerializationP**: objeto serializable de la meta del nivel. Se usará la clase de Platform por ser idéntica al de PlatformVehicles. **listTileSerializationP:** lista que contiene los ladrillos del nivel para que puedan ser serializados. Se usará la clase de Platform por ser idéntica al de PlatformVehicles.
Métodos Propuestos
addObjects(): añade un nuevo elemento al diccionario de objetos. **getObject():** obtiene el valor de un elemento del diccionario de objetos dada una clave.

5.4.3.1.6 Clase SerializerPlatformVehicles

Nombre de la Clase
SerializerPlatformVehicles
Descripción
Serializa y de serializa los elementos del juego de control de vehículos para guardar el estado de edición en el sistema de ficheros y poder recuperarlo posteriormente.
Responsabilidades
Serializa los elementos del juego de control de vehículos convirtiéndolos a un único fichero binario que se almacena en la ruta dada. También transforma un fichero binario válido en las diferentes variables del juego.
Atributos Propuestos
-
Métodos Propuestos
Serialize(): transforma de un objeto AlmacenPlatformVehicles a formato binario (.bat) **Deserialize():** transforma un archivo en formato binario (.bat) en un objeto AlmacenPlatformVehicles.

5.4.3.1.7 Clase CharacterSerializationPlatformVehicles

Nombre de la Clase
CharacterSerializationPlatformVehicles
Descripción
Clase serializable con las propiedades del personaje principal del juego orientado a control de vehículos.
Responsabilidades
Actúa como clase contenedora de las propiedades del personaje principal cuando llegue el momento de serializar el juego.
Atributos Propuestos
ID: identificador del personaje. **Width**: ancho del personaje. **Height:** altura del personaje. **Health**: vidas del personaje. **on_acelerate_sound:** sonido a reproducir cuando acelere. **on_brake_sound:** sonido a reproducir cuando frene. **acelerate_factor**: factor de movimiento hacia la derecha. **brake_factor:** factor de movimiento hacia la izquierda. **friction_factor:** factor de fricción en cuestas. **max_positive_aceleration**: máxima velocidad hacia la derecha. **max_negative_aceleration**: maxima velocidad hacia la izquierda. **max_friction_slope**: maxima velocidad en cuestas. **Gravity**: aceleración de la gravedad. **Inertia**: indica si el personaje debe rodar por las cuestas o no. **listSpritePropertiesSerialization:** lista de las diferentes animaciones serializadas que componen el personaje.
Métodos Propuestos
CharacterSerializationPlatformVehicles(): constructor con todos los parámetros.

5.4.3.1.8 Clase CheckpointSerializationPlatformVehicles

Nombre de la Clase
CheckpointSerializationPlatformVehicles
Descripción
Clase serializable con las propiedades del punto de control del juego orientado a control de vehículos.
Responsabilidades
Actúa como clase contenedora de las propiedades del punto de control cuando llegue el momento de serializar el juego.
Atributos Propuestos
Width: ancho del elemento. **Height:** alto del elemento. **SoundOnCollected:** sonido que se reproduce al colisionar con el personaje. **ListSpritePropertiesSerialization**: lista de las diferentes animaciones serializadas que componen el punto de control.
Métodos Propuestos
CheckpointSeralizationPlatformVehicles(): constructor con todos los parámetros.

5.4.3.1.9 Clase TrapSerializationPlatformVehicles

Nombre de la Clase
TrapSerializationPlatformVehicles
Descripción
Clase serializable con las propiedades de la trampa del juego orientado a control de vehículos.
Responsabilidades
Actúa como clase contenedora de las propiedades de la trampa cuando llegue el momento de serializar el juego.
Atributos Propuestos
Width: ancho del elemento. **Height:** alto del elemento. **ListSpritePropertiesSerialization**: lista de las diferentes animaciones serializadas que componen la trampa.
Métodos Propuestos
TrapSerializationPlatformVehicles(): constructor con todos los parámetros.

5.4.3.2 Subsistema analizador

5.4.3.2.1 Clase Analyzer

Nombre de la Clase
Analyzer
Descripción
Clase responsable de procesar los ficheros necesarios para generar el juego y también para sustituir los valores de la instancia del DSL en la plantilla.
Responsabilidades
Copia la plantilla a la carpeta de destino incluyendo los niveles definidos en formato XML. Procesa el XML del DSL insertando los valores en un hashMap. Substituye cada uno de los parámetros en el DSL.java y guarda ese fichero en su ruta correspondiente dentro del proyecto Android final.
Atributos Propuestos
dictionary: contiene las etiquetas de la instancia del DSL como clave y el valor concreto de dichas etiquetas como valor.
Métodos Propuestos
analyzeDSL(): abre el fichero xml y lo procesa, insertando cada uno de los valores en el diccionario. **copyToDestiny():** copia la plantilla original a la ruta de destino del proyecto Android. **createDestinyStructure():** crea la estructura del proyecto Android en la carpeta destino elegida por el usuario previamente en el editor. **substitute():** abre el fichero DSL.java y el manifest.xml en la ruta destino y sustituye cada string por su valor correspondiente de la instancia del editor.

5.4.3.2.2 Clase Reader

Nombre de la Clase
Reader
Descripción
Clase que hace algunas comprobaciones para que la clase Analyzer pueda trabajar correctamente.
Responsabilidades
Comprueba si se dan las siguientes condiciones: 1. Ruta raíz debe estar definida. 2. La ruta de destino debe ser creada sino existe. 3. El fichero dsl.xml debe existir dentro de la ruta de fichero.
Atributos Propuestos
rutaOrigen: ruta donde se almacena la plantilla en la cual se harás las sustituciones pertinentes. **rutaDestino**: ruta elegida por el usuario para guardar el proyecto Android. **rutaFicheros:** ruta donde se encuentra el proyecto del juego en proceso de creación.
Métodos Propuestos
searchRoutes: método que comprueba si se cumplen las condiciones anteriormente citadas.

5.4.3.3 Subsistema videojuego

5.4.3.3.1 Clase GameLoop

Nombre de la Clase
GameLoop
Descripción
Clase que maneja el hilo que ejecuta el videojuego.
Responsabilidades
Se encarga de inicializar la GameView y gestionar los *frames* del videojuego llamando a las funciones del GameView que necesitan ser actualizadas cada *frame.* También controla los *frames* saltados por el dispositivo para evitar "saltos" en el juego.
Atributos Propuestos
FPS: *frames* por segundo. **MAXIMO_FRAME_SALTARSE:** número máximo de *frames* que se pueden saltar sin actualizar. **FRAME_TIEMPO:** duración de un *frame.* **Running:** booleano que indica si el hilo está en ejecución. **Superficie:** superficie donde se dibujará el canvas.
Métodos Propuestos
Run: pone el hilo en ejecución y lleva a cabo toda la lógica de actualización de *frames* para que el juego se actualice.

5.4.3.3.2 Clase GameView

Nombre de la Clase
GameView
Descripción
Clase que gestiona todos los elementos que intervienen en la lógica del videojuego.
Responsabilidades
Se encarga de inicializar todos los botones en la pantalla y carga el nivel correspondiente. Maneja el ciclo del juego en tanto que dibuja en pantalla los elementos y actualiza su comportamiento. También se encarga de comprobar los estados del juego y de procesar la entrada táctil del usuario.
Atributos Propuestos
buttonAcelerate: botón para acelerar. **buttonBrake:** botón para frenar. **lifeBoard:** objeto para mostrar el número de vidas restantes. **timeBoard:** objeto para mostrar el tiempo que ha transcurrido desde el inicio del nivel. **gameStatus:** estado del juego. **screenWidth:** ancho de la pantalla. **screenHeight:** alto de la pantalla.
Métodos Propuestos
Init Inicializa todos los elementos del juego. **Draw:** Dibuja en pantalla todos los elementos del juego. **Update:** Actualiza la lógica del videojuego. **OnTouch:** Recibe el evento táctil que realizó el usuario y modifica el movimiento del jugador. **LoadLevel:** Carga el nivel seleccionado.

5.4.3.3.3 Clase Level

Nombre de la Clase
Level
Descripción
Clase que almacena todos los elementos propios de un nivel del juego.
Responsabilidades
Se encarga de inicializar los ladrillos del nivel, el fondo, el jugador principal, las trampas y los puntos de control además de actualizar la lógica de los mismos y pintarlos en la pantalla.
Atributos Propuestos
Tiles: diferentes estructuras de las que está compuesto la estructura del nivel. **Player:** jugador principal con el que interactuará el jugador moviéndolo por la pantalla. **Traps:** lista de trampas con las cuales si el usuario colisiona, perderá una vida. **CheckPoints:** lista de puntos de control con los cuales si el usuario colisiona, se guardará la posición en el caso de que posteriormente pierda una vida. **StartPoint:** punto de inicio para el personaje. **EndPoint:** punto objetivo al que tiene que llegar el personaje para finalizar el nivel. **Background:** fondo del nivel.
Métodos Propuestos
Init Inicializa todos los elementos del nivel. **Draw:** Dibuja en pantalla todos los elementos del nivel. **Update:** Actualiza la lógica propia del nivel. **LoadTiles:** Carga los ladrillos del nivel. **LoadPlayer:** Inicializa el jugador principal con todos sus parámetros. **LoadTraps:** Inicializa las trampas y las distribuye por el nivel dependiendo de sus parámetros. **LoadCheckPoints:** Inicializa los puntos de control y los distribuye por el nivel dependiendo de sus parámetros. **HandleCollisions:** Controla las colisiones del jugador principal con los elementos del nivel y le aplica a éste el comportamiento oportuno.

5.4.3.3.4 Clase Player

Nombre de la Clase
Player
Descripción
Clase que almacena todos los elementos propios del personaje principal.
Responsabilidades
Se encarga de inicializar el personaje y gestionar el movimiento hacia izquierda y derecha, el comportamiento cuando colisiona con los diferentes elementos del nivel, aplicarle el efecto de la gravedad y cambiar los estados del personaje.
Atributos Propuestos
X: posición en el eje horizontal. **Y:** posición en el eje vertical. **Width**: ancho del personaje. **Height:** altura del personaje. **Health**: vidas del personaje. **on_acelerate_sound:** sonido a reproducir cuando acelere. **on_brake_sound:** sonido a reproducir cuando frene. **acelerate_factor**: factor de movimiento hacia la derecha. **brake_factor:** factor de movimiento hacia la izquierda. **friction_factor:** factor de fricción en cuestas. **max_positive_aceleration**: máxima velocidad hacia la derecha. **max_negative_aceleration**: maxima velocidad hacia la izquierda. **max_friction_slope**: maxima velocidad en cuestas. **Gravity**: aceleración de la gravedad. **Inertia**: indica si el personaje debe rodar por las cuestas o no. **AcelerationX:** aceleración hacia derecha o izquierda. **AcelerationY:** aceleración hacia arriba o abajo. **IsAlive:** indica si el persona está vivo o muerto. **ActualTile:** indica el ladrillo actual en el que se sitúa el personaje.
Métodos Propuestos
Player Inicializa todas las propiedades del personaje. **Draw:** Dibuja el personaje en pantalla de acuerdo a sus propiedades. **Update:** Actualiza los valores del personaje. **ChangeImage:** Cambia la imagen del personaje cuando cambian algunas de sus propiedades. **Acelerate:** Añade la aceleration X a la x actual para conseguir que el personaje se mueva a lo largo del eje horizontal. **Decelerate:** Resta la aceleration X a la x actual para conseguir que el personaje frene. **HandleCollisions:** Mueve el personaje hacia izquierda y derecha e impide que traspase los *Tiles* sólidos. También se encarga de que suba las rampas ascendentes y baje las descendentes. **Gravity:** Hace que el personaje caiga cuando está en el aire. **OnKilled:** Pone IsAlive a false para indicar que el jugador ha perdido en el nivel.

5.4.3.3.5 Clase CheckPoint

Nombre de la Clase
CheckPoint
Descripción
Clase que define un elemento del nivel de tipo punto de control.
Responsabilidades
Se encarga de inicializar el punto de control con sus propiedades propias e indicarle el comportamiento cuando el personaje principal colisiona contra él.
Atributos Propuestos
ImageOnCollected: imagen a mostrar cuando el personaje pasa por el punto de control. **SoundOnCollected**: sonido que se reproduce cuando el personaje pasa por el punto de control.
Métodos Propuestos
CheckPoint: Inicializa todas las propiedades del punto de control. **OnCollected:** Cambia la imagen cuando el personaje principal colisiona con él y reproduce el sonido seleccionado para ese evento.

5.4.3.3.6 Clase Trap

Nombre de la Clase
Trap
Descripción
Clase que define un elemento del nivel de tipo trampa.
Responsabilidades
Se encarga de inicializar una trampa con sus propiedades y cambiar su comportamiento cuando el jugador colisiona con él.
Atributos Propuestos
-
Métodos Propuestos
Trap Inicializa todas las propiedades de la trampa.

5.4.3.3.7 Clase LevelElement

Nombre de la Clase
LevelElement
Descripción
Clase abstracta que encapsula los elementos de interacción del nivel.
Responsabilidades
Se encarga de inicializar el elemento con sus propiedades y dibujarlo en pantalla.
Atributos Propuestos
X: posición en el eje horizontal. **Y:** posición en el eje vertical. **Width:** ancho del elemento. **Height:** alto del elemento. **Image:** imagen que representa al elemento por defecto.
Métodos Propuestos
LevelElement Inicializa todas las propiedades del elemento del nivel. **Draw:** Dibuja el elemento en la pantalla de acuerdo a sus propiedades. **OnCollected:** Método abstracto que definirán sus hijos dependiendo de su comportamiento.

5.4.3.3.8 Clase Tile

Nombre de la Clase
Tile
Descripción
Clase que almacena todos los elementos propios de los ladrillos del nivel
Responsabilidades
Se encarga de inicializar el *Tile* con su imagen y su tipo de colisión. También almacena los tipos de colisión en un tipo enumerado.
Atributos Propuestos
X: posición en el eje horizontal. **Y:** posición en el eje vertical. **Width:** ancho del Tile. Será estático. **Height:** alto del Tile. Será estático. **Image:** imagen seleccionada para representar el *Tile* en el canvas. **Collision**: tipo de *Tile* (pasable, sólido, rampa de 22º, 45º o 67º hacia arriba o hacia abajo)
Métodos Propuestos
Tile Inicializa la textura y la colisión del ladrillo. **getCollision:** Obtiene el tipo de colisión.

5.4.3.3.9 Clase Control

Nombre de la Clase
Control
Descripción
Clase padre para cualquier elemento que sea pulsado en el juego.
Responsabilidades
Se encarga de inicializar las características típicas de un elemento susceptible de ser pulsado y de representarlo en la pantalla. Delega en los hijos las clases de isPressed y Draw.
Atributos Propuestos
Position: posición X, Y del control. **Image:** imagen que representará al control en el canvas.
Métodos Propuestos
Control: inicializa la posición y la imagen del control. **isPressed:** clase abstracta que implementarán los hijos.

5.4.3.3.10 Clase Button

Nombre de la Clase
Button
Descripción
Clase hija de Control. Elemento que se pulsa en el juego para conseguir un efecto.
Responsabilidades
Se encarga de inicializar su imagen y su posición pasándole los parámetros al padre y también inicializa su tamaño.
Atributos Propuestos
Width: ancho del botón. **Height:** alto del botón.
Métodos Propuestos
Button inicializa todos los parámetros del botón. **isPressed:** determina si el click del ratón está dentro de los límites del botón. **Draw:** dibuja el botón en el canvas.

5.5 Análisis de Casos de Uso y Escenarios

5.5.1 Caso de uso: Crear videojuego.

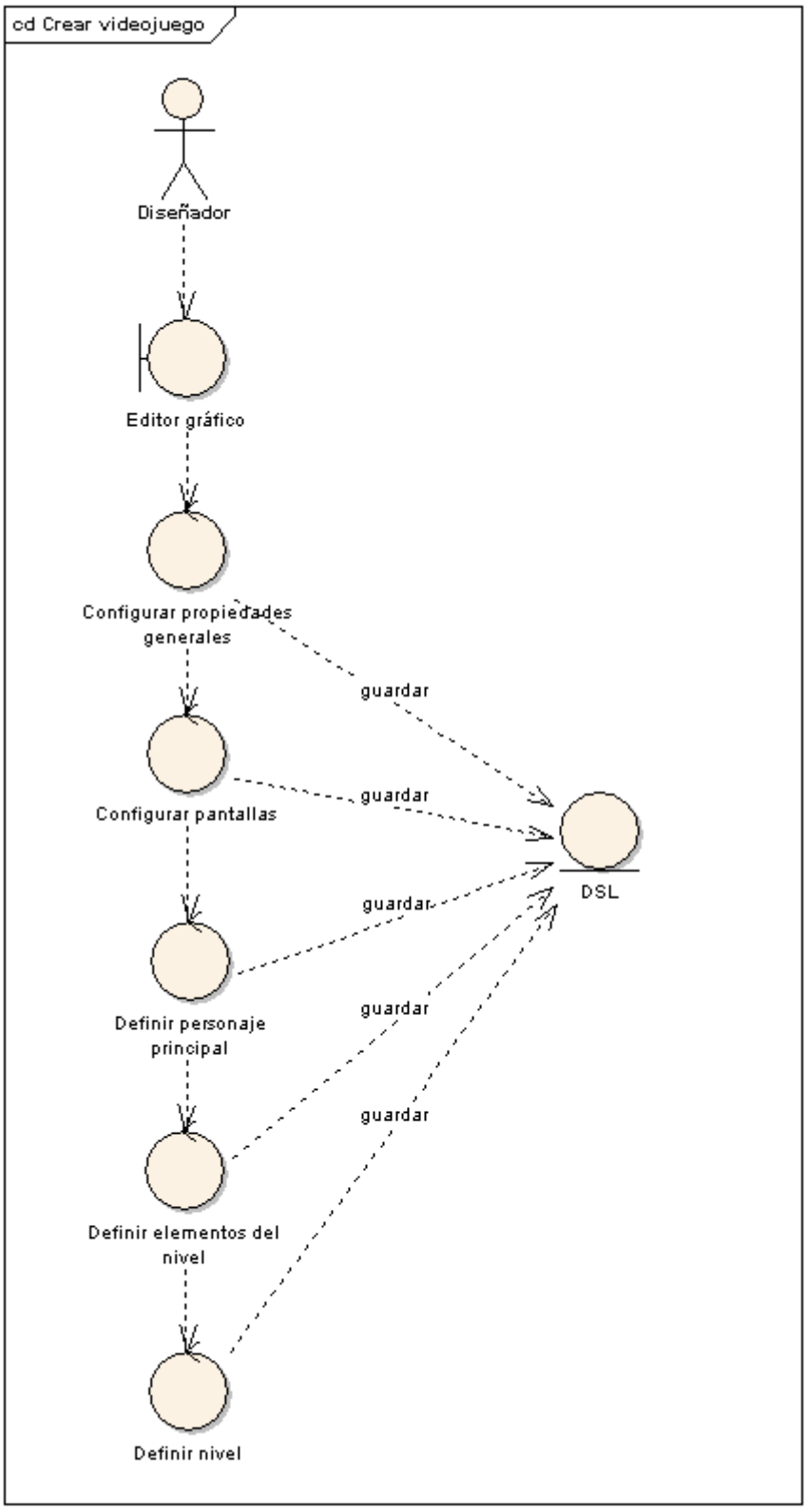


Ilustración 20. Diagrama de robustez para el caso de uso "Crear videojuego"

Crear videojuego	
Precondiciones	Tener instalado el editor Gade4All.
Poscondiciones	Se crea un proyecto Android con el videojuego definido con los elementos definidos por el usuario en el editor.
Actores	Iniciado y finalizado por el usuario diseñador
Descripción	El usuario diseñador: 1. Abrirá el editor, escogerá la tipología y elegirá una ruta donde se generará el videojuego al finalizar el proceso. 2. Configurará las propiedades generales del videojuego (ver subcaso de uso "Configurar propiedades generales". 3. Configurará las diferentes pantallas y los elementos principales de la pantalla de juego (ver subcaso de uso "Configurar pantallas") 4. Definirá los elementos pertinentes del personaje principal (caso de uso "Crear personaje principal") además de la física que gobernará sobre el mismo (caso de uso "Definir física del juego") 5. El usuario definirá diversas trampas, puntos de control, *Tiles* y metas en con las características que desee (ver casos de uso "Definir elementos del nivel") 6. En la pantalla de creación de niveles, seleccionará crear un nuevo nivel y definirá su estructura y la condición que permite al personaje pasar de nivel (ver subcaso de uso "Definir nivel"). También dispondrá el personaje y los elementos por el escenario además de los sonidos pertinentes de manera opcional (caso de uso "Asignar sonidos")
Variaciones (escenarios secundarios)	Si el usuario no define alguna de las pantallas comentadas anteriormente, se tomarán las pantallas por defecto. No habrá, en cambio, personajes y elementos del escenario por defecto.
	Si el usuario referencia algún elemento audiovisual que no existe, el videojuego funcionará igualmente aunque no sonará ni se mostrará dicho sonido/imagen.
	Si el usuario quiere añadir un nivel completamente vacío, el editor mostrará un mensaje de alerta. Es necesario incluir, al menos, un personaje y una meta.
Excepciones	Ante cualquier deficiencia en el proceso de creación del videojuego, el editor mostrará el mensaje pertinente para evitar que el juego final contenga errores.
Notas	-

5.5.2 Caso de uso: Guardar videojuego.

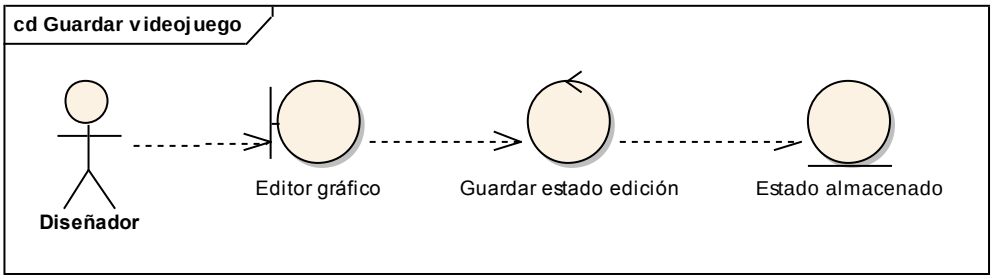


Ilustración 21. Diagrama de robustez para el caso de uso "Guardar videojuego"

Guardar videojuego	
Precondiciones	El usuario debe haber iniciado el editor gráfico y debe haber indicado una ruta de destino de la aplicación
Poscondiciones	Se guardará el estado de cada pantalla por la que haya pasado el usuario así como los recursos audiovisuales implicados.
Actores	Iniciado y finalizado por el usuario diseñador
Descripción	El usuario diseñador podrá guardar el estado de la edición en cualquier momento pulsando el botón "Guardar". También se efectuará un guardado automático al pasar de una pantalla a otra de tal manera que se cree un archivo por cada pantalla del editor dentro de la ruta que escogió el usuario al comenzar la edición.
Variaciones (escenarios secundarios)	Si el usuario no escoge una ruta de guardado no podrá continuar con la edición.
	Si el usuario guarda un juego que ya había guardado previamente la información se reescribe en los archivos .dat ya existentes.
Excepciones	El explorador de archivos no encuentra la ruta o existe algún problema que impide que se guarde la copia del estado. El explorador de Windows se encargará de informar del error pertinente.
Notas	El estado de cada pantalla se almacenará en un fichero binario .dat. También se almacenan los ficheros de imágenes y sonidos a medida que se vayan definiendo en cada pantalla.

5.5.3 Caso de uso: Cargar videojuego.

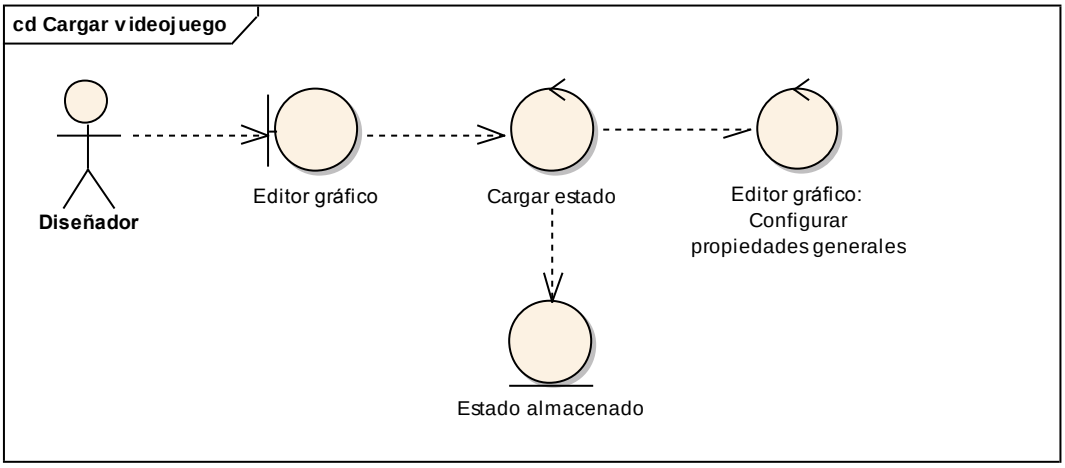


Ilustración 22. Diagrama de robustez para el caso de uso "Cargar videojuego"

Cargar videojuego	
Precondiciones	El usuario debe haber iniciado el editor gráfico y debe haber guardado previamente el juego para poder cargarlo.
Poscondiciones	Se cargará el estado de cada pantalla que se hubiera personalizado a la hora de guardar.
Actores	Iniciado y finalizado por el usuario diseñador
Descripción	1. El usuario diseñador podrá cargar el estado de un juego únicamente desde la pantalla inicial. 2. Escogerá un archivo .dat de un videojuego previamente guardado. 3. El videojuego en proceso de edición se cargará en el editor.
Variaciones (escenarios secundarios)	-
Excepciones	Se produce un error en el proceso de carga por la falta de algún recurso audiovisual. El editor debe informar del error debidamente.
	Se intenta cargar un .bat que no almacena ficheros guardados por el editor. Éste debe informar del error debidamente.
Notas	-

5.5.4 Caso de uso: Exportar videojuego.

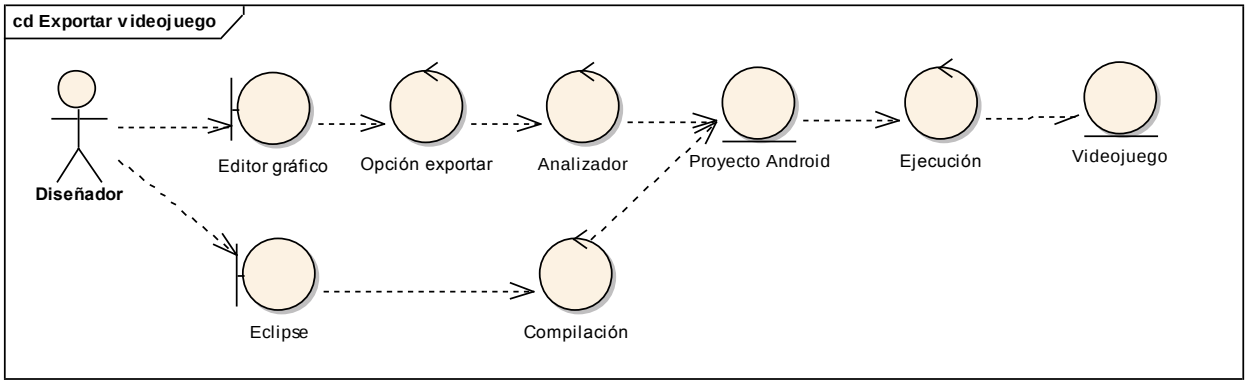


Ilustración 23. Diagrama de robustez para el caso de uso "Exportar videojuego"

Exportar videojuego	
Precondiciones	El usuario debe haber creado un videojuego completo y correcto.
Poscondiciones	Se creará un proyecto Android que simplemente compilando y ejecutando dentro del Eclipse genere un videojuego orientado a vehículos con los parámetros que introdujo el usuario.
Actores	Iniciado y finalizado por el usuario diseñador
Descripción	El usuario diseñador: 1. Accederá a la pantalla de exportación de videojuego. 2. Seleccionará la ruta de exportación y esperará el tiempo necesario. 3. Importará el proyecto generado en Eclipse. 4. Compilará y ejecutará dicho proyecto en Eclipse. 5. Obtendrá un videojuego orientado a vehículos completamente funcional.
Variaciones (escenarios secundarios)	El usuario podrá definir la instancia del DSL manualmente antes de realizar la exportación.
Excepciones	El videojuego final funciona pero de manera muy poco intuitiva porque el usuario ha elegido parámetros válidos pero que no funcionan correctamente en la práctica.
	El analizador ha fallado y se obtiene un proyecto con errores. El videojuego no puede ejecutarse.
	La plantilla presenta fallos y se obtiene un proyecto con errores. El proyecto no puede ejecutarse.
Notas	El usuario deberá tener instalado el Eclipse para importar el proyecto generado.

5.5.5 Caso de uso: Jugar videojuego

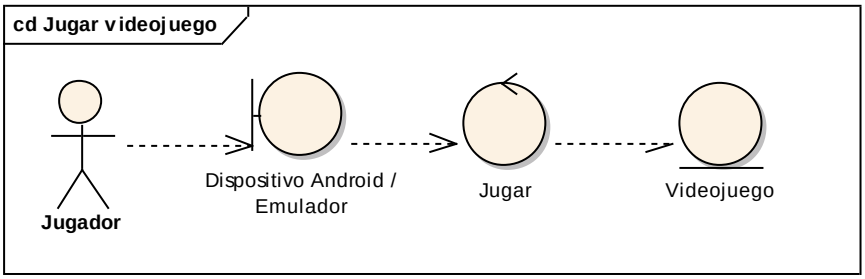


Ilustración 24. Diagrama de robustez para el caso de uso "Jugar videojuego"

Cargar videojuego	
Precondiciones	El usuario diseñador debe haber creado un juego correcto previamente.
Poscondiciones	El usuario ejecutará y jugará al videojuego de plataformas orientado a vehículos.
Actores	Iniciado y finalizado por el usuario jugador
Descripción	1. El jugador instalará y ejecutará el videojuego en su dispositivo móvil. 2. El usuario jugará al videojuego.
Variaciones (escenarios secundarios)	El usuario también podrá jugar al videojuego a través del emulador que incorpora el Android SDK para Eclipse.
Excepciones	El videojuego no puede ejecutarse debido a fallos de compilación. El Eclipse mostrará el error correspondiente en su consola.
	El videojuego se ejecuta hasta determinado momento donde se produce una excepción.
Notas	-

5.6 Análisis de Interfaces de Usuario

5.6.1 Pantallas de la aplicación

La aplicación completa poseerá dos interfaces de interacción con el usuario: el editor gráfico y el videojuego propiamente dicho por lo que se definirán las pantallas de ambos subsistemas por separado.

5.6.1.1 Pantallas del editor gráfico

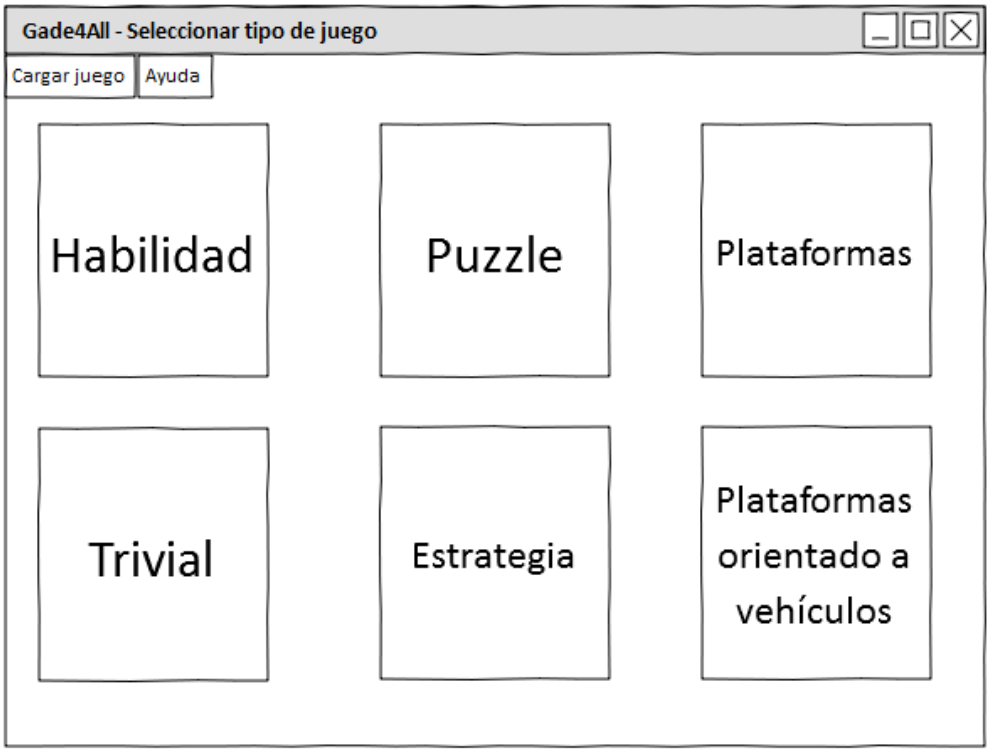


Ilustración 25. Pantalla principal del editor gráfico donde se selecciona el tipo de juego a crear

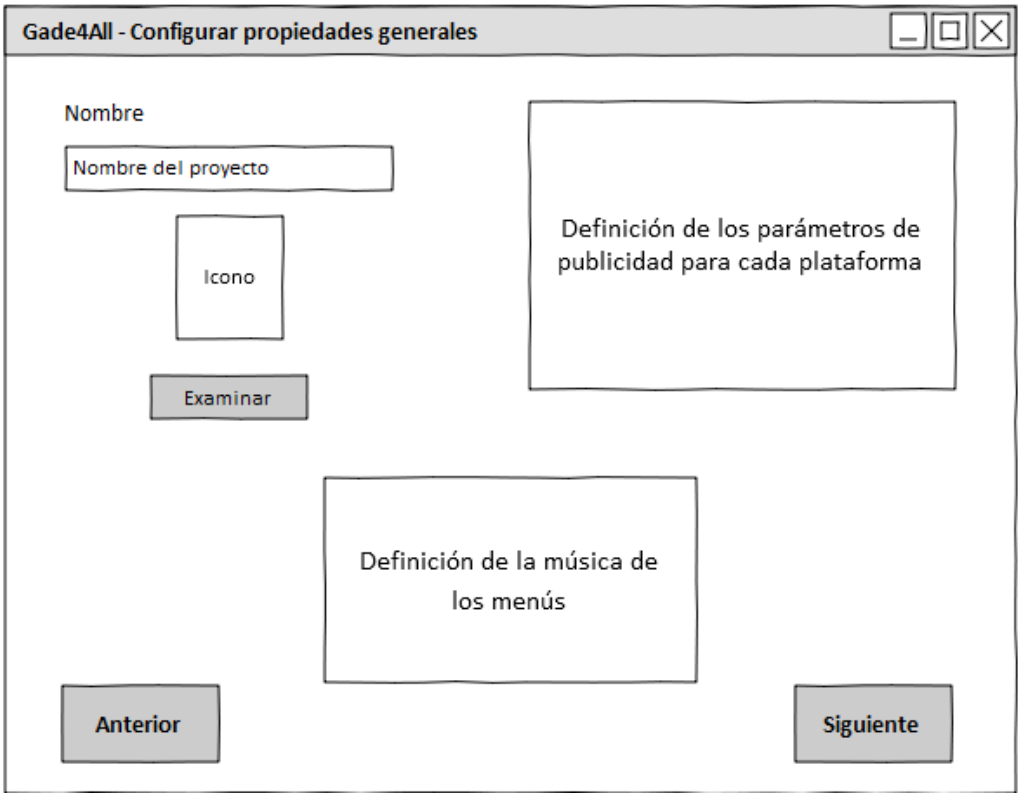


Ilustración 26. Pantalla de configuración de propiedades globales

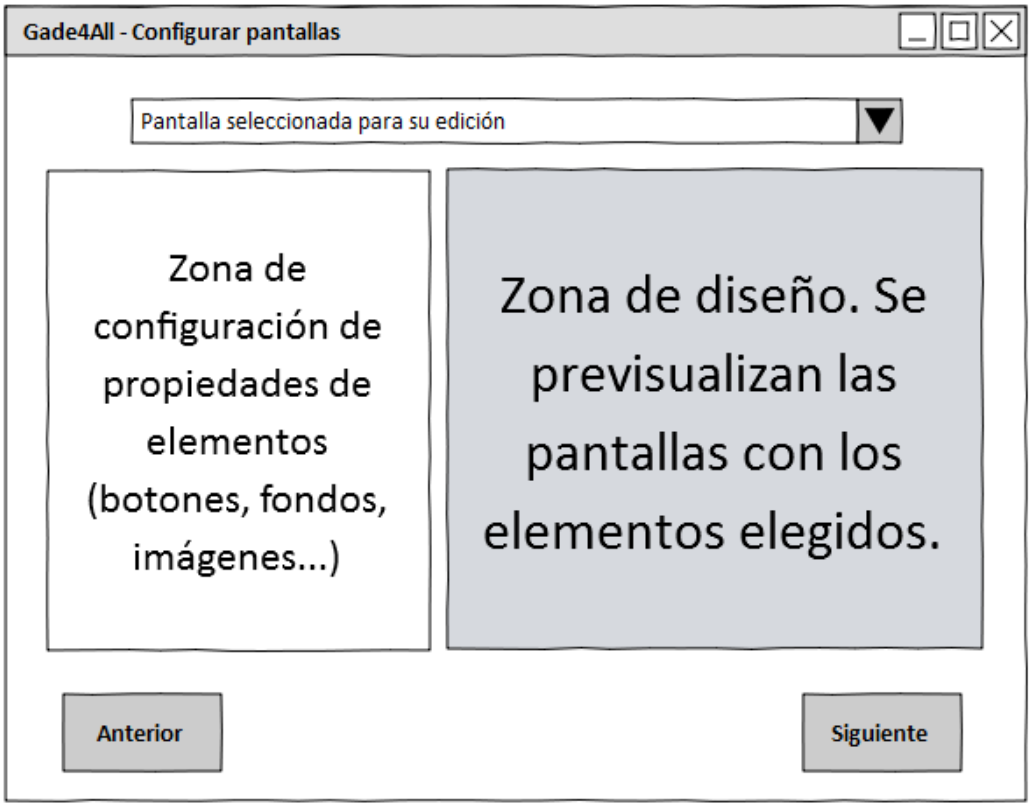


Ilustración 27. Pantalla de configuración de pantallas

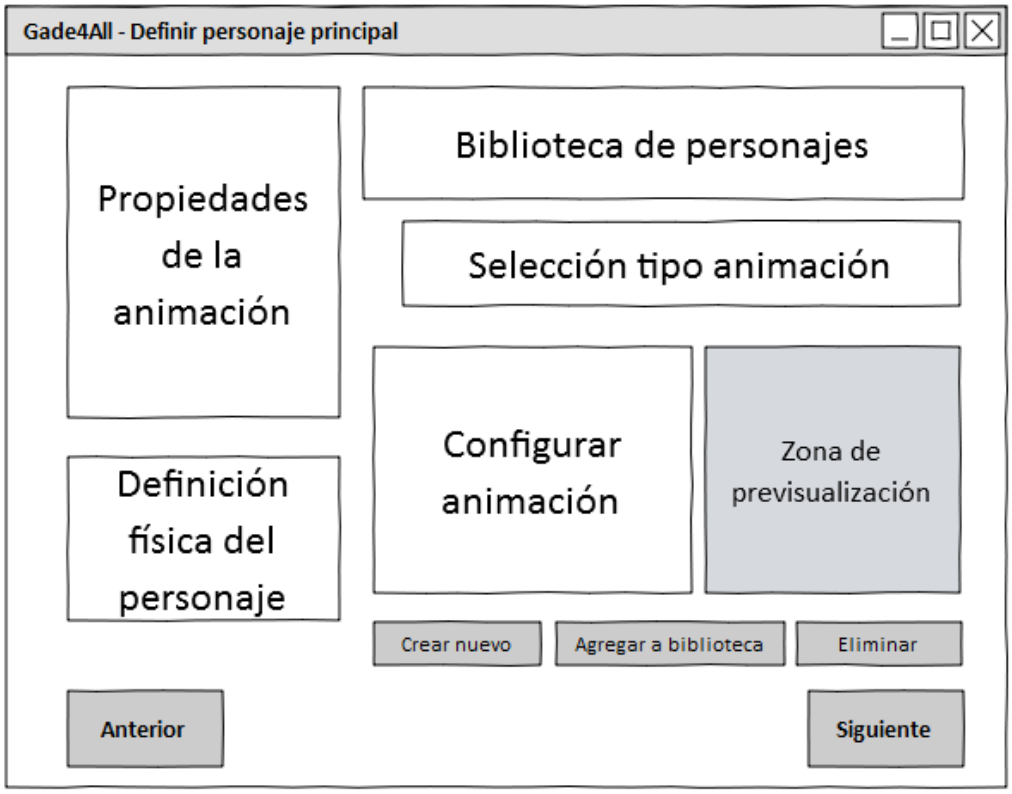


Ilustración 28. Pantalla de definición de personaje principal

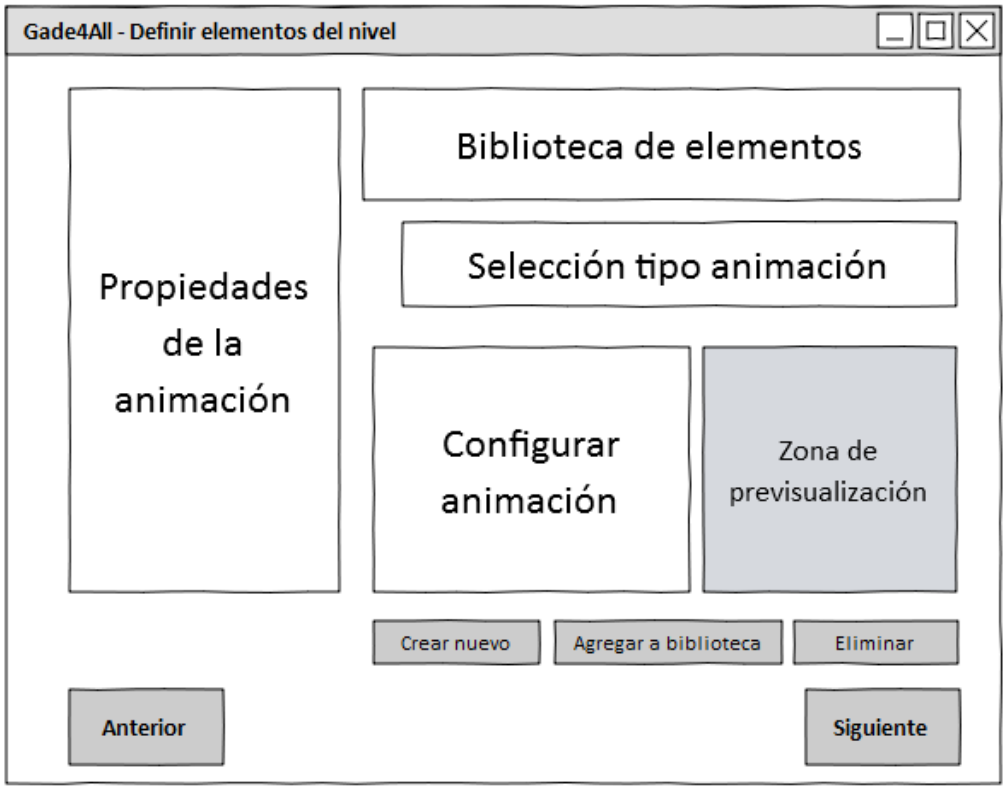


Ilustración 29. Pantalla de definición de elementos del nivel

Esta pantalla será análoga para configurar las trampas, la meta, los puntos de control y los *Tiles.*

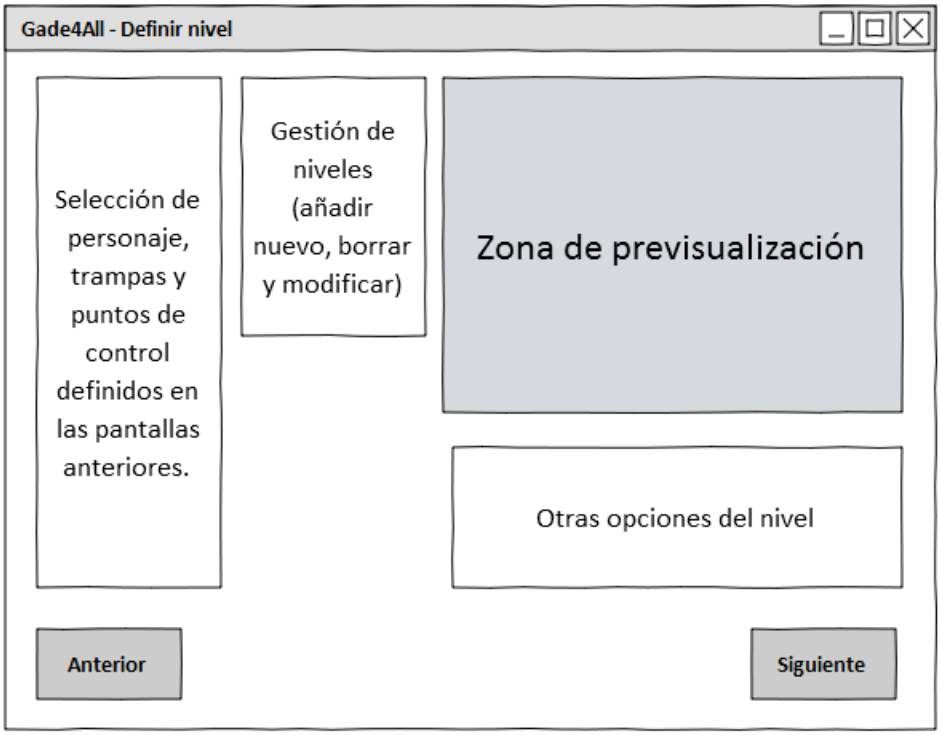


Ilustración 30. Pantalla de definición de un nivel

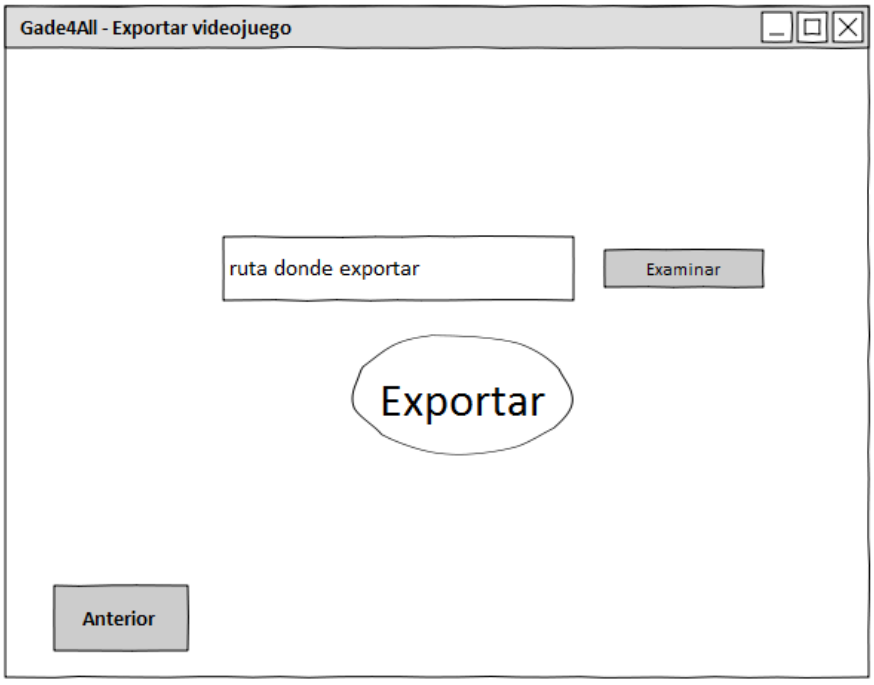


Ilustración 31. Pantalla de exportación de videojuego

5.6.1.2 Pantallas del videojuego

Aunque las pantallas del videojuego sean personalizables (al contrario que la navegabilidad entre pantallas que será inmutable), el editor tendrá los siguientes modelos de diseño por defecto.

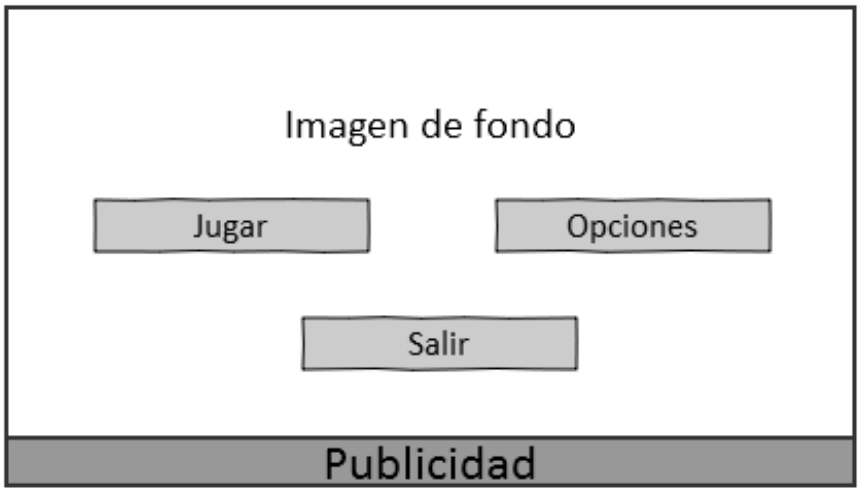


Ilustración 32. Pantalla principal

Ilustración 33. Pantalla de opciones

Los botones de música y efectos aparecerán alternativamente de acuerdo a la opción seleccionada. Por ejemplo, si la música está activada y los efectos también, sólo aparecerán esos dos botones.

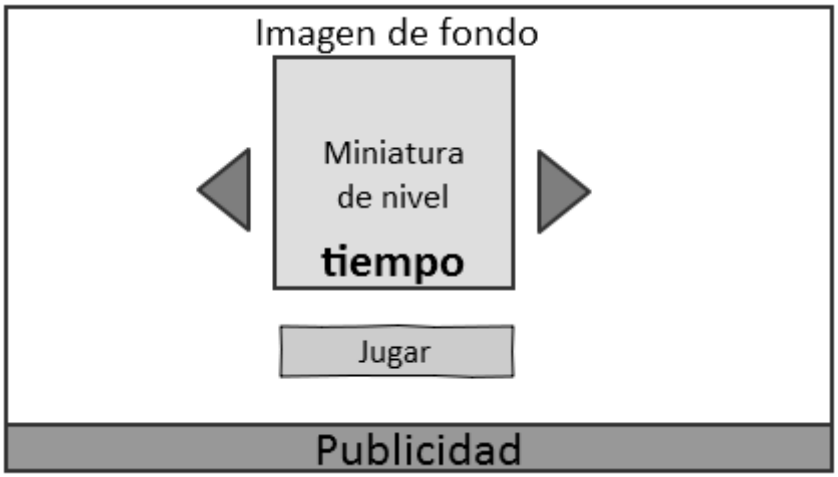


Ilustración 34. Pantalla de selección de nivel

En cada nivel se indicará el tiempo necesario para completarlo en el caso de que se hubiera definido este parámetro en el editor.

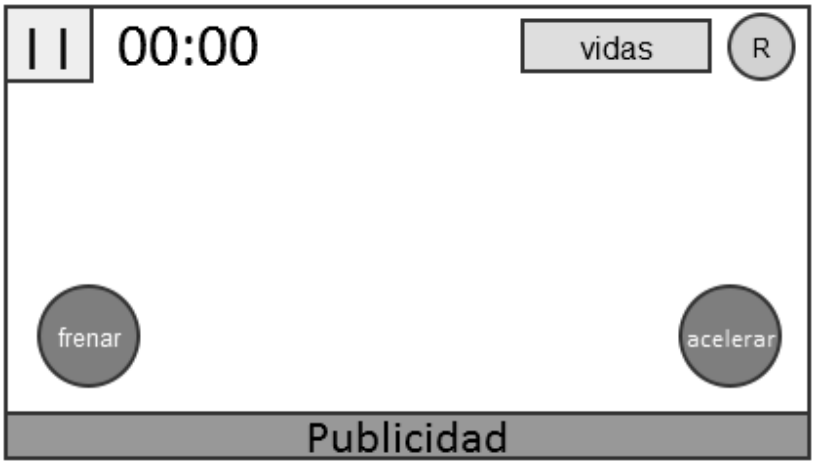


Ilustración 35. Pantalla de juego

Ilustración 36. Pantalla de pausa

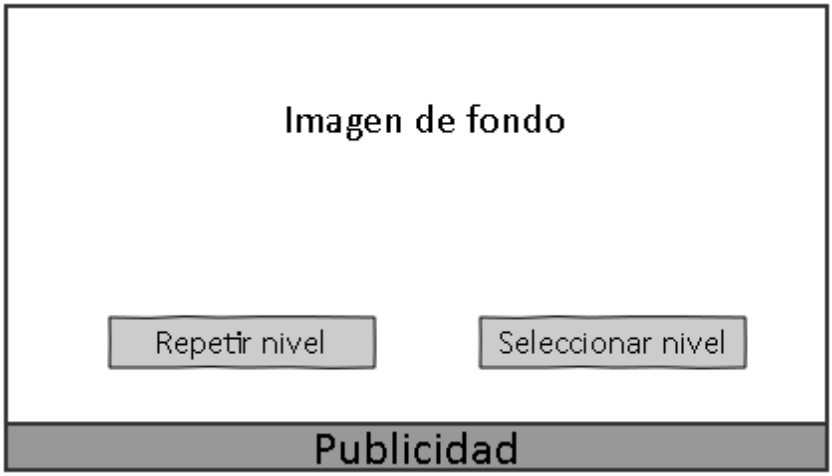


Ilustración 37. Pantalla de juego finalizado sin éxito (Game Over)

Ilustración 38. Pantalla de nivel finalizado con éxito

Ilustración 39. Pantalla de juego finalizado con éxito

5.6.2 Descripción del Comportamiento de la Interfaz

5.6.2.1 Editor gráfico

Las diferentes pantallas del editor gráfico que impliquen el tratamiento de recursos visuales se diseñarán de manera muy parecida para facilitar la usabilidad y de tal manera que la zona de visualización siempre se sitúe a la derecha y la zona de definición de propiedades y elementos se sitúe a la izquierda.

Los mensajes de error que se mostrarán se identificarán en la especificación del plan de pruebas pero se seguirá el convenio de resaltar en rojo aquellos campos problemáticos a la hora de crear el videojuego. Asimismo, los mensajes de error deben ser lo suficientemente explicativos y aclarativos para que el usuario pueda resolver sin problemas la causa que lo activó.

5.6.3 Diagrama de Navegabilidad

5.6.3.1 Editor gráfico

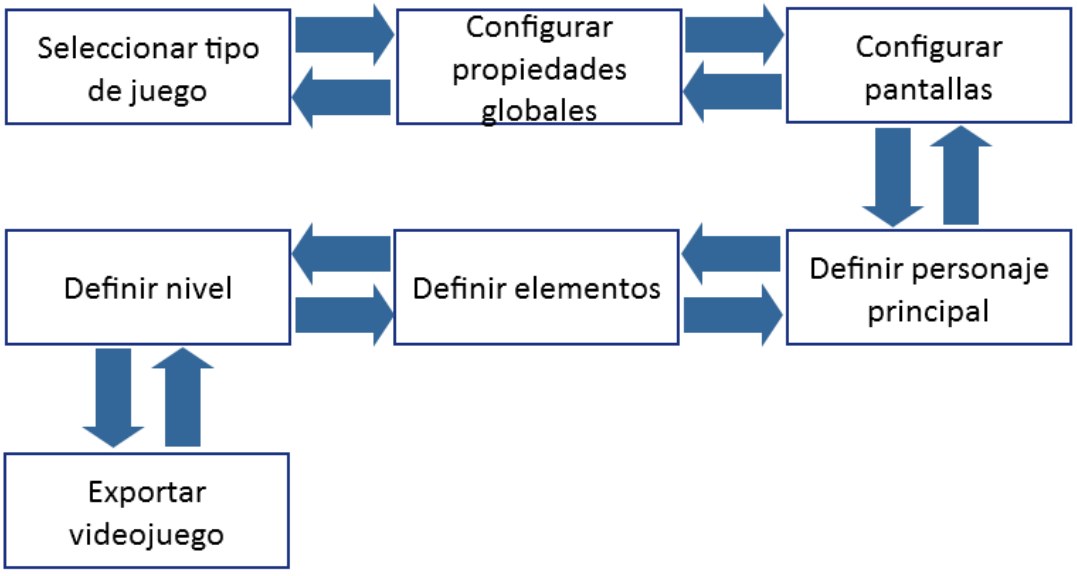


Ilustración 40. Mapa de pantallas del editor gráfico

5.6.3.2 Videojuego

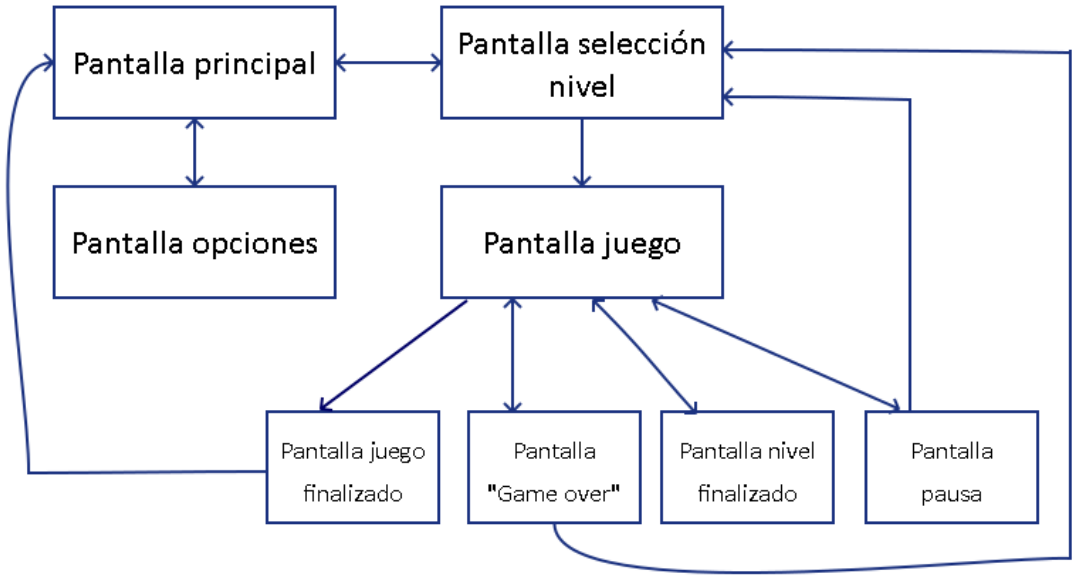


Ilustración 41. Mapa de pantallas del videojuego

5.7 Especificación del Plan de Pruebas

El plan de pruebas se centrará en testear los puntos más críticos de la aplicación. Por ello, se realizarán pruebas unitarias, de integración, del sistema y de usabilidad.

5.7.1 Pruebas unitarias

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código, o en este caso una clase individual que cumple con una función concreta.

En el presente proyecto se efectuarán pruebas unitarias tanto al subsistema editor como al subsistema videojuego pues es crítico que todas las partes de ambos subsistemas funcionen correctamente.

5.7.2 Pruebas de integración y del sistema

Las pruebas de integración comprenden verificaciones asociadas a grupos de componentes, verificando que éstos funcionan correctamente cuando estos son ensamblados para cumplir con una función concreta.

Ya que la aplicación final contará con tres módulos principales se deberá comprobar que todos funcionan correctamente y de manera coordinada. Para ello se comprobará que diferentes configuraciones del DSL desde el editor desembocarán en videojuegos finales igualmente aptos.

5.7.3 Pruebas de usabilidad

Este tipo de pruebas determinan la satisfacción del cliente con el producto final.

Ya que uno de los principales objetivos del proyecto Gade4All reside en generar una aplicación usable e intuitiva para la generación automática de videojuegos a personas no expertas en el tema, se realizarán diversas pruebas de usabilidad a personas de diferentes perfiles para comprobar que esto se cumple.

5.7.3.1 Caso de uso: Crear videojuego

5.7.3.1.1 Subcaso de uso: Configurar propiedades generales

Caso de Uso 1: Crear videojuego. Subcaso de uso: Configurar propiedades generales	
Prueba	**Resultado Esperado**
Insertar un nombre inválido para el videojuego.	El sistema informa de un error y no deja seguir avanzando en la edición.
Prueba	**Resultado Esperado**
Insertar un nombre válido para el videojuego.	Cuando finalice la edición se obtiene un videojuego con el nombre asignado previamente.
Prueba	**Resultado Esperado**
Escoger un icono para la aplicación que no es formato .png.	El editor no permite seleccionar imágenes con formato distinto a .png. El explorador de carpetas para escoger el icono tiene un filtro, así que solamente mostrará imágenes con esta extensión.
Prueba	**Resultado Esperado**
Escoger un icono .png para la aplicación.	El editor muestra una pre visualización de la imagen escogida y se asignará al videojuego.
Prueba	**Resultado Esperado**
Marcar la opción de publicidad y no se añade ningún ID.	El sistema visa de que se necesita un ID para mostrar la publicidad.
Prueba	**Resultado Esperado**
Marcar la opción de publicidad e incluye un ID.	Se asigna el valor al identificador de la publicidad y se muestra dicha publicidad en el videojuego final.

Prueba	**Resultado Esperado**
Pulsar en el botón "siguiente"	Se crean las carpetas necesarias en la ruta escogida por el usuario para almacenar los diferentes elementos globales que ya se han definido.

5.7.3.1.2 Subcaso de uso: Configurar pantallas

Caso de Uso 1: Crear videojuego. Subcaso de uso: Configurar pantallas	
Prueba	**Resultado Esperado**
Definir la pantalla principal.	Es la primera pantalla que aparecerá una vez se cargue el juego en el dispositivo o en el emulador.
Prueba	**Resultado Esperado**
No definir la pantalla principal.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
Prueba	**Resultado Esperado**
No definir la pantalla de opciones.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
Prueba	**Resultado Esperado**
No definir la pantalla de victoria.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
Prueba	**Resultado Esperado**
No definir la pantalla de derrota.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
Prueba	**Resultado Esperado**
No definir la pantalla de fin.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
Prueba	**Resultado Esperado**
Disponer los elementos de la pantalla de juego en el medio de la misma.	El editor no dará ningún error aunque es lógico que de esta manera no se visualice correctamente el juego. Este aspecto de disposición de los elementos se deja a elección del diseñador.

Prueba	**Resultado Esperado**
Seleccionar un botón.	El botón se selecciona dentro de un recuadro
Prueba	**Resultado Esperado**
Escoger un fondo no .png	El editor no permite seleccionar imágenes con formato distinto a .png. El explorador de carpetas para escoger el icono tiene un filtro, así que solamente mostrará imágenes con esta extensión.
Prueba	**Resultado Esperado**
Cambiar la posición de un botón.	El botón se mueve y aparece en dicha posición en el juego final.
Prueba	**Resultado Esperado**
Ocultar un botón	El botón baja de opacidad y no aparece en el juego final

5.7.3.1.3 Subcaso de uso: Definir física del juego

Caso de Uso 1: Crear videojuego. Subcaso de uso: Definir física del juego	
Prueba	**Resultado Esperado**
Insertar un valor negativo para aquellos factores que no pueden serlo (gravedad, factor de fricción, etc.)	El valor negativo no se asigna. Se preparará el editor para que no acepte números negativos para determinados elementos.
Prueba	**Resultado Esperado**
No modificar ninguno de los valores que afecten a la física del juego.	La física del videojuego será la que está definida por defecto en el editor.
Prueba	**Resultado Esperado**
Insertar un valor de gravedad igual a cero.	El editor no permitirá asignar valores iguales o menores que cero para la gravedad.
Prueba	**Resultado Esperado**
Insertar un valor de factor de aceleración igual a cero.	El editor permitirá asignar esos valores pero mostrará un mensaje indicando que los factores de aceleración que sean igual a cero impedirán que el personaje se mueva en una u otra dirección.

Prueba	**Resultado Esperado**
Insertar un valor con letras en uno de los campos.	El editor emitirá un mensaje de que el número introducido es incorrecto y no asignará dicho valor.
Prueba	**Resultado Esperado**
Insertar valores válidos en todos los *textbox.*	El editor asignará dichos valores a la instancia del DSL.

5.7.3.1.4 Subcaso de uso: Definir nivel

Caso de Uso 1: Crear videojuego. Subcaso de uso: Definir nivel	
Prueba	**Resultado Esperado**
Definir un nivel vacío.	El editor mostrará un error y no permitirá al usuario definir un nivel vacío. El personaje principal y el punto final son los únicos elementos obligatorios a la hora de definir un nivel.
Prueba	**Resultado Esperado**
Definir el punto de meta en una posición anterior al punto de salida.	Se asignan ambos valores a la instancia del DSL. Una vez se defina el videojuego, el personaje principal podrá ir de derecha a izquierda para completar el nivel.
Prueba	**Resultado Esperado**
Se sitúa un Tile en la posición X,Y.	Si la posición es válida dentro de los parámetros del nivel, se situará el Tile correspondiente en el escenario.
Prueba	**Resultado Esperado**
Definir un escenario compuesto por Tiles de tal manera que es imposible que el personaje principal llegue a la meta.	El editor asignará igualmente los recursos a la instancia del DSL. Es competencia del usuario diseñador definir niveles que puedan ser finalizados.
Prueba	**Resultado Esperado**
Se pulsa el botón “Siguiente” sin haber definido la meta y el personaje principal	Es obligatorio que exista un personaje principal y una meta para que el nivel se guarde. Se mostrará el mensaje de error correspondiente.

Prueba	**Resultado Esperado**
No se escoge ninguna imagen para el fondo.	El fondo por defecto es negro
Prueba	**Resultado Esperado**
Se pulsa el botón de "Siguiente"	Los parámetros elegidos se agregan a la instancia del DSL y los recursos gráficos se almacenan en la carpeta Drawable.
Prueba	**Resultado Esperado**
Situar un punto de control en un punto posterior a la meta.	El punto de control se agrega sin problemas. Es responsabilidad del usuario diseñador definir puntos de control que realmente se vayan a utilizar.
Prueba	**Resultado Esperado**
Situar una trampa en un punto posterior a la meta.	La trampa se agrega sin problemas. Es responsabilidad del usuario diseñador definir trampas que sean activas en el desarrollo del videojuego.
Prueba	**Resultado Esperado**
Situar una trampa en una posición que impide al personaje llegar a la meta.	La trampa se agrega sin problemas. Es responsabilidad del usuario diseñador definir un nivel factible.
Prueba	**Resultado Esperado**
No seleccionar número de vidas.	Para seleccionar un número de vidas se utilizará un selector numérico que ya tendrá un valor por defecto.
Prueba	**Resultado Esperado**
Seleccionar un número de vidas inferior a uno.	El selector numérico sólo permite seleccionar números entre uno y cinco.
Prueba	**Resultado Esperado**
Desmarcar el *checkbox* de tiempo y no definir un tiempo máximo para finalizar el nivel.	Si no se define un valor de tiempo mayor que cero, aunque la casilla de "sin límite de tiempo" esté desmarcada, el nivel se ejecutará sin límite de tiempo.
Prueba	**Resultado Esperado**
Desmarcar el *checkbox* de tiempo, definir un tiempo válido y escoger un número de vidas entre uno y cinco.	Se asignarán dichos parámetros al .xml del nivel y, posteriormente, el usuario jugador tendrá que llegar a la meta del nivel en el tiempo fijado y sin gastar las vidas definidas.

5.7.3.2 Subcaso de uso: Crear personaje principal

Caso de Uso 1: Crear videojuego. Subcaso de uso: Crear personaje principal	
Prueba	**Resultado Esperado**
Definir la misma imagen para todos los eventos.	El editor asignará la misma imagen para todos los eventos sin errores.
Prueba	**Resultado Esperado**
Seleccionar un personaje ya creado de la biblioteca.	El editor cargará dicho personaje y sus animaciones.
Prueba	**Resultado Esperado**
Definir una animación para algún evento del personaje.	Se mostrará una vista previa en el editor que recoja los parámetros introducidos para definir la animación (imágenes, número de frames, si debe reproducirse en bucle, etc.)
Prueba	**Resultado Esperado**
Pulsar el botón "Añadir a biblioteca" cuando no hay ningún personaje creado.	El personaje incompleto no se agrega a la biblioteca.
Prueba	**Resultado Esperado**
Pulsar el botón "Añadir a biblioteca" cuando se ha finalizado la creación de un personaje.	El personaje se agrega correctamente a la biblioteca para poder ser usado posteriormente en otros desarrollos.
Prueba	**Resultado Esperado**
Pulsar el botón "Borrar" seleccionando un personaje de la biblioteca.	El personaje se borra de la biblioteca si no está vinculado a ningún escenario.
Prueba	**Resultado Esperado**
Seleccionar un personaje de la biblioteca	Se cargan todas las variables propias del personaje y se visualiza una muestra previa de sus animaciones.

Prueba	**Resultado Esperado**
Seleccionar un personaje de la biblioteca, cambiar alguna de sus propiedades y hacer clic en el botón de "Actualizar".	Si las propiedades son válidas, se actualiza dicha información en la base de datos de la biblioteca.
Prueba	**Resultado Esperado**
Intentar asignarle al personaje un tamaño menor que 35.	El editor sólo permitirá añadir valores para el tamaño del personaje principal comprendidos entre 35 y 70.
Prueba	**Resultado Esperado**
Intentar asignarle al personaje un tamaño mayor que 70.	El editor sólo permitirá añadir valores para el tamaño del personaje principal comprendidos entre 35 y 70.

5.7.3.2.1 Subcaso de uso: Definir elementos del nivel

Caso de Uso 1: Crear videojuego. Subcaso de uso: Definir elementos de nivel	
Prueba	**Resultado Esperado**
Elegir un recurso gráfico para un punto de control que no sea .png	El editor no permite seleccionar imágenes con formato distinto a .png. El explorador de carpetas para escoger el icono tiene un filtro, así que solamente mostrará imágenes con esta extensión.
Prueba	**Resultado Esperado**
Definir la imagen, el número de frames y el tamaño de un punto de control.	Se muestra la vista previa del punto de control.
Prueba	**Resultado Esperado**
Seleccionar un punto de control de la biblioteca	Los parámetros del punto del control ya creado se cargan y se muestran en el editor con su vista previa incluida.

Prueba	**Resultado Esperado**
Agregar un punto de control nuevo a la biblioteca	Si todos sus parámetros son correctos se agrega ese nuevo punto de control a la biblioteca.
Prueba	**Resultado Esperado**
Elegir un recurso gráfico para una trampa que no sea .png	El editor no permite seleccionar imágenes con formato distinto a .png. El explorador de carpetas para escoger el icono tiene un filtro, así que solamente mostrará imágenes con esta extensión.
Prueba	**Resultado Esperado**
Definir la imagen y la posición de una trampa	Se muestra la vista previa de la trampa si los valores son válidos.
Prueba	**Resultado Esperado**
Seleccionar una trampa de la biblioteca	Los parámetros de la trampa ya creada se cargan y se muestran en el editor con su vista previa incluida.
Prueba	**Resultado Esperado**
Agregar un tipo de trampa nueva a la biblioteca	Si todos sus parámetros son correctos se agrega esa nueva trampa a la biblioteca

5.7.3.2.2 Subcaso de uso: Asignar sonidos

Caso de Uso 1: Crear videojuego. Subcaso de uso: Asignar sonidos	
Prueba	**Resultado Esperado**
Elegir una melodía para los menús del juego con formato .mp4	La aplicación no permite seleccionar archivos de sonido con formato distinto a .mp3 o .wav. El explorador de carpetas para escoger el sonido tiene un filtro, así que solamente mostrará archivos con dicha extensión.
Prueba	**Resultado Esperado**
Seleccionar un archivo de sonido en formato .mp3	La aplicación permite escucharlo y lo asigna ante un determinado evento o como música ambiente.

Prueba	**Resultado Esperado**
Pulsar el botón de reproducir sin haber escogido previamente un sonido.	El sonido no se reproduce.
Prueba	**Resultado Esperado**
Pulsar el botón de reproducir habiendo escogido previamente un sonido.	El sonido se reproduce sin problemas.
Prueba	**Resultado Esperado**
Pulsar sobre el botón de pausa una vez el sonido esté reproduciéndose.	El sonido se para.
Prueba	**Resultado Esperado**
No seleccionar ningún sonido para un determinado evento.	El editor permite no asignar sonidos a eventos. Se trata de un parámetro no obligatorio.
Prueba	**Resultado Esperado**
Definir todos los sonidos necesarios y se continúa la edición.	Todos los sonidos seleccionados se incluirán en una carpeta *sounds.* La correspondencia entre sonidos y eventos se verá reflejada en el xml del nivel.

5.7.3.3 Caso de uso: Guardar videojuego

Caso de Uso 2: Guardar videojuego	
Prueba	**Resultado Esperado**
Navegar entre pantallas	En la carpeta donde se almacena el juego se crea un archivo .dat por cada pantalla que se configura y se pulsa sobre “Siguiente”. Si ya existe se actualiza.

Prueba	**Resultado Esperado**
Pulsar sobre la opción "Guardar" del menú.	Actualiza los archivos .dat existentes o creo los nuevos necesarios guardando los parámetros del videojuego en desarrollo.

5.7.3.4 Caso de uso: Cargar videojuego

Caso de Uso 3: Cargar videojuego	
Prueba	**Resultado Esperado**
Cargar un videojuego guardado previamente	El videojuego se cargará correctamente recuperando todos los elementos que se habían salvado en su momento.
Prueba	**Resultado Esperado**
Cargar un juego con errores en el momento que se guardó.	El editor mostrará los errores que impiden la edición del videojuego.
Prueba	**Resultado Esperado**
Pulsar el botón "anterior" en algún momento de la edición	En la pantalla anterior aparecerán los valores que introdujo el usuario al acceder a dicha pantalla.

5.7.3.5 Caso de uso: Exportar videojuego

Caso de Uso 4: Exportar videojuego	
Prueba	**Resultado Esperado**
Se finaliza el proceso de crear videojuego y se pulsa "siguiente" en la última pantalla.	Se muestra la pantalla de selección de ruta de despliegue.
Prueba	**Resultado Esperado**
Se finaliza el proceso de crear videojuego dejando errores en el proceso y se pulsa "siguiente"	El editor muestra un diálogo de error informando que no se puede exportar un juego con errores.

Prueba	**Resultado Esperado**
Se elige una ruta donde almacenar el videojuego correcto y se pulsa el botón de generar.	Tras el tiempo necesario, el proyecto Android con todos los elementos definidos aparecerá en la ruta seleccionada.
Prueba	**Resultado Esperado**
Se importa el proyecto generado en el Eclipse.	El proyecto se importa sin dar errores.
Prueba	**Resultado Esperado**
Se compila el proyecto.	El proyecto se compila sin errores y se genera el .apk, archivo a instalar posteriormente en el dispositivo del usuario jugador.

5.7.3.6 Caso de uso: Jugar videojuego

Caso de Uso 5: Jugar videojuego	
Prueba	**Resultado Esperado**
Se instala el fichero .adk generado en Eclipse en el dispositivo.	El videojuego se instala correctamente en el dispositivo y muestra en primera instancia la pantalla principal.
Prueba	**Resultado Esperado**
Se ejecuta el videojuego en el emulador del Eclipse	El videojuego se instala correctamente y funciona. No se asegura que la velocidad de refresco iguale a la del dispositivo.
Prueba	**Resultado Esperado**
Se ejecuta el videojuego en diferentes dispositivos Android con diferentes resoluciones	La lógica del videojuego se comporta exactamente igual en todas las resoluciones.
Prueba	**Resultado Esperado**
Se selecciona un nivel	Tras el tiempo de espera requerido, el personaje principal apareceré en el nivel seleccionado

Prueba	**Resultado Esperado**
Se procede a seleccionar un nivel sin límite de tiempo	En el selector de niveles no aparecerá ningún indicador del tiempo necesario para superar el nivel.
Prueba	**Resultado Esperado**
Una vez superado un nivel se vuelve a la pantalla de selección de nivel.	En la pantalla de selección de nivel pueden elegirse sólo aquellos niveles que se hayan superado.
Prueba	**Resultado Esperado**
Se llega a la meta del nivel cumpliendo el objetivo predefinido	Se muestra un mensaje de fin de nivel con el tiempo invertido en completarlo y eventualmente se pasa al siguiente nivel si lo hubiera.
Prueba	**Resultado Esperado**
El personaje principal pierde todas las vidas	Se muestra una imagen tipo “Game Over” y se reinicia el nivel.
Prueba	**Resultado Esperado**
Se finaliza el último nivel	Se muestra una imagen de juego finalizado y se vuelve al menú principal del juego.
Prueba	**Resultado Esperado**
El personaje principal finaliza un nivel	Aparece la pantalla de fin de nivel con el tiempo que le llevó al usuario completarlo.
Prueba	**Resultado Esperado**
El personaje choca contra un ladrillo sólido.	El personaje principal debe de frenar aunque el botón de avance esté pulsado.
Prueba	**Resultado Esperado**
El personaje principal colisiona con una rampa ascendente	El personaje principal debe ascender por la rampa con la inclinación correspondiente.
Prueba	**Resultado Esperado**
El personaje principal colisiona con una rampa descendente	El personaje principal debe descender por la rampa con la inclinación correspondiente.

Prueba	**Resultado Esperado**
El personaje principal colisiona contra una trampa	El personaje principal pierde una vida.
Prueba	**Resultado Esperado**
El personaje principal cae por la parte inferior de la pantalla.	El personaje principal pierde una vida.
Prueba	**Resultado Esperado**
El personaje principal muere por alguna de las razones anteriores habiendo alcanzado algún *checkpoint* del nivel previamente.	El personaje principal pierde una vida y aparece en el último punto de control por el que haya pasado si sus vidas no han llegado a cero.
Prueba	**Resultado Esperado**
El personaje principal no supera el tiempo mínimo definido para completar el nivel	El marcador de tiempo debe de parpadear cuando el tiempo esté cerca de agotarse y una vez que lo haga el nivel se reinicia.
Prueba	**Resultado Esperado**
Al personaje principal sólo le queda una vida y la pierde.	El nivel se reinicia.
Prueba	**Resultado Esperado**
Se pausa el juego	Aparece la pantalla de pausa y la lógica del juego deja de actualizarse.
Prueba	**Resultado Esperado**
Se cambian los parámetros por defecto de los sonidos en la pantalla de opciones desactivando la música y los efectos.	Si los efectos y la música ambiental están desactivados, no se escuchará ningún tipo de sonido al comenzar cualquiera de los niveles.

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

6.1.1 Diagramas de Paquetes

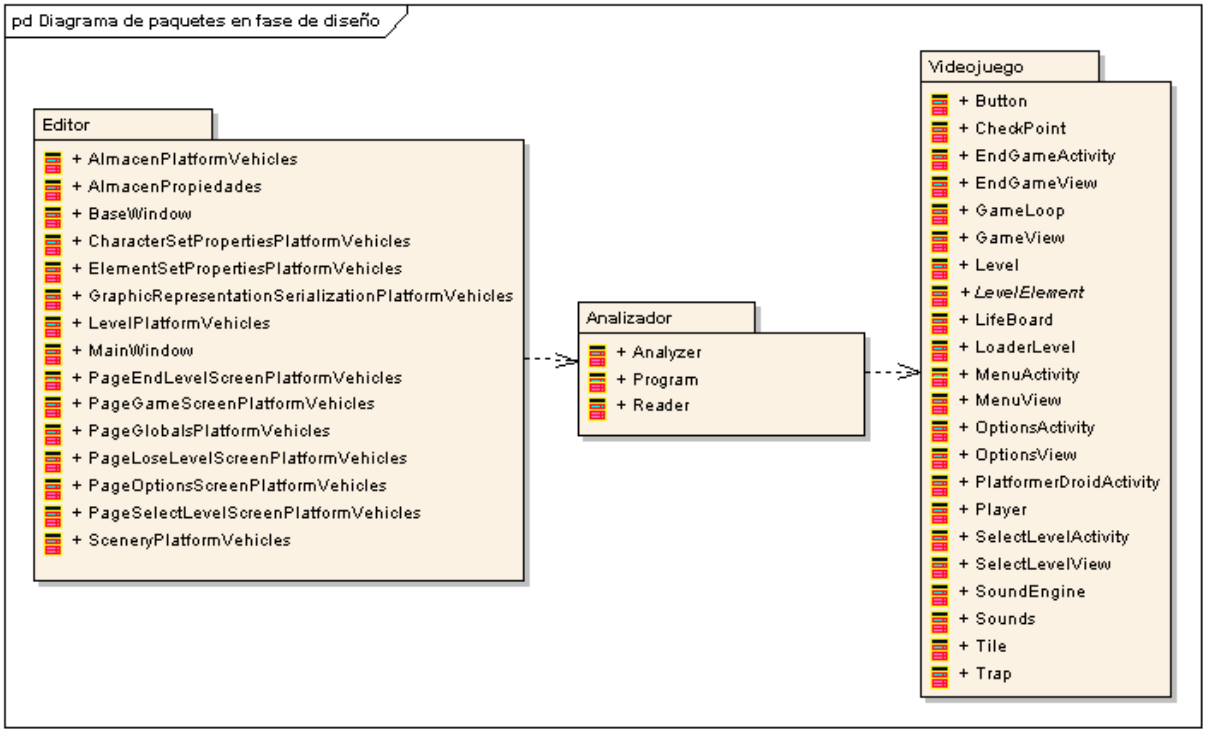


Ilustración 42. Diagrama de paquetes general de la aplicación

6.1.1.1 Paquete editor

El paquete editor abarca toda la funcionalidad del editor de videojuegos. Dado que el editor ya poseía la implementación de la edición de las diferentes tipologías, el diseño se centrará en las partes del editor relacionadas con la tipología de plataformas orientado al control de vehículos.

El editor está formado por las clases que gestionan el comportamiento de las diferentes pantallas (código .cs) y su vista (código .xaml), clases auxiliares, clases contenedoras que almacenan la información de cada unas de esas pantallas y clases que permiten serializar el contenido de las contenedoras para guardar/recuperar el estado de edición.

6.1.1.2 Paquete analizador

Este paquete recibe toda la información recogida por el editor (archivo XML con los parámetros generales del juego, XML con la definición de los niveles y los recursos audiovisuales). A continuación se encarga de copiar la plantilla incompleta del videojuego en un directorio y copiar dentro de ella todos los ficheros de definición de niveles y recursos audiovisuales. Después procesa todos los elementos del dsl.xml y sustituye los valores en el código Java de la plantilla (clase DSL.java) El resultado de todas las sustituciones y copias de archivos es un proyecto de aplicación Android para Eclipse.

En analizador está formado por clases programadas en C#.

6.1.1.3 Paquete videojuego

Este paquete aporta toda la lógica de funcionamiento del videojuego y constituye la plantilla en la cual se efectuarán las sustituciones pertinentes por parte del analizador. Contiene las clases en Java que implementan la funcionalidad de las pantallas previas al videojuego (principal, opciones y selección de nivel) además de la pantalla de juego. También posee las clases que encapsulan los niveles, los elementos del nivel y el personaje principal donde se gestiona la física del juego. Otras clases útiles reciben el xml del nivel, obtienen los parámetros del mismo y lo cargan en el juego. Existen otras clases auxiliares que se encargan de las animaciones de los elementos del nivel, de mostrar una resolución adecuada de los recursos visuales, de reproducir los sonidos y la música del videojuego, etc. Finalmente, la clase DSL es la encargada de almacenar todos los parámetros parametrizables que se definieron en el editor.

6.2 Diseño de Clases

6.2.1 Diagrama de Clases

6.2.1.1 Diagrama de Clases paquete Editor

El subsistema editor abarca la funcionalidad de edición para cada tipología de juego. Dada su inmensidad, los posteriores diagramas se centrarán únicamente en aquellas clases programadas para conseguir un editor de videojuegos de la tipología orientada a vehículos, en concreto aquellas clases y funciones significativamente diferentes a las ya existentes en el editor de manera previa al presente proyecto.

Para una mejor visibilidad se ha decidido separar los diagramas de clases del subsistema editor dependiendo de su funcionalidad.

6.2.1.1.1 Diagrama de clases paquete editor: funcionalidad pantallas.

El siguiente diagrama de clases resume aquellas clases que implementan la funcionalidad y la interacción con el usuario de las diferentes pantallas del editor.

La clase ElementSetPropertiesPlatformVehicles representa a las clases **CheckpointSetPropertiesPlatformVehicles**, **TrapSetPropertiesPlatformVehicles**, **TileSetPropertiesPlatformVehicles** y **TargetSetPropertiesPlatformVehicles** simultáneamente. Se han omitido del diagrama para ganar claridad y porque todas siguen el mismo esquema de atributos y operaciones que resume ElementSetPropertiesPlatformVehicles.

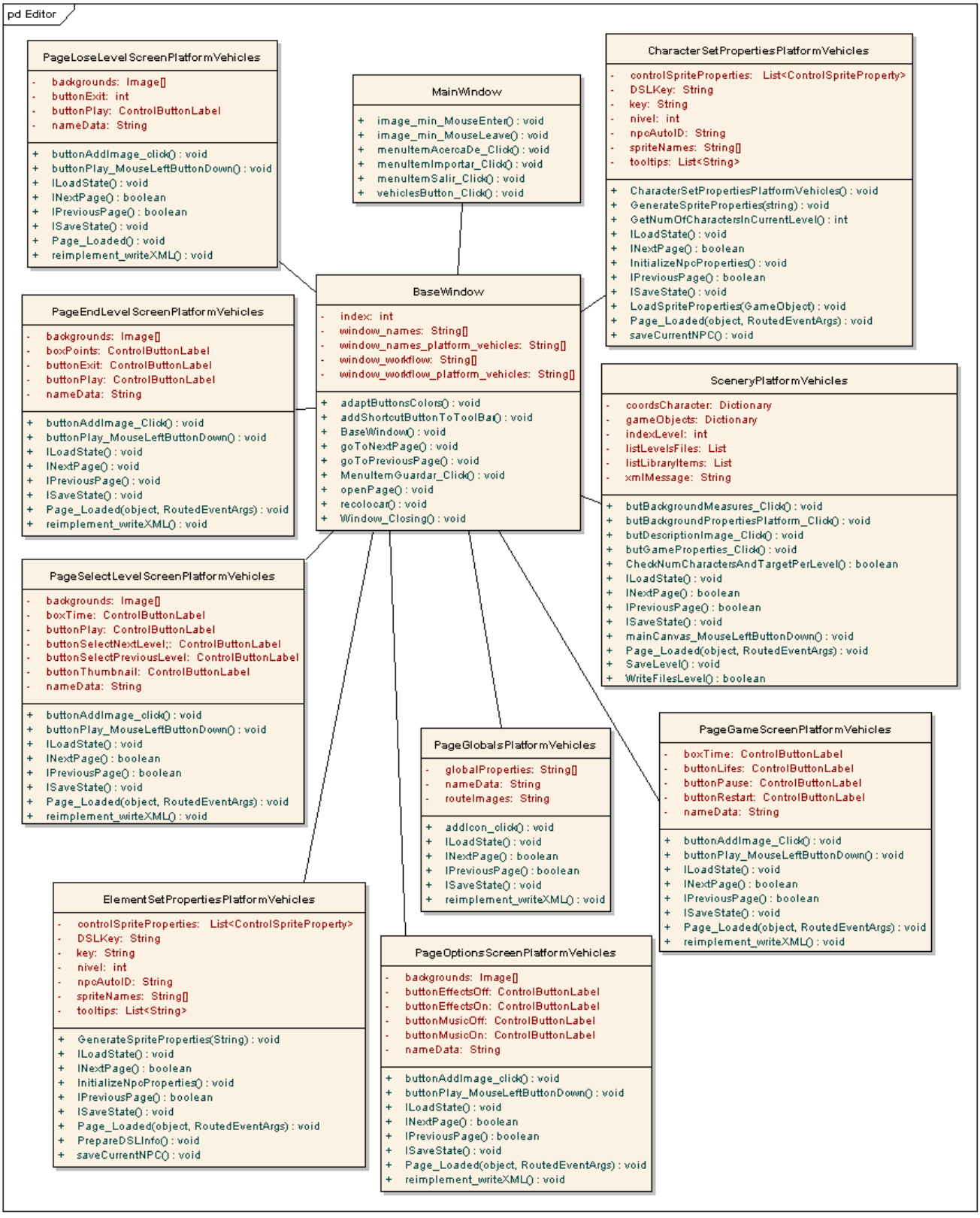


Ilustración 43. Diagrama de clases en la fase de diseño para la funcionalidad de las pantallas del paquete editor.

6.2.1.1.2 Diagrama de clases paquete editor: StorePlatformVehicles

Esta clase implementada siguiendo el patrón *singleton* almacena los xml de las diferentes pantallas y se encarga de crear un fichero donde vuelca toda esa información una vez se finalice la edición.

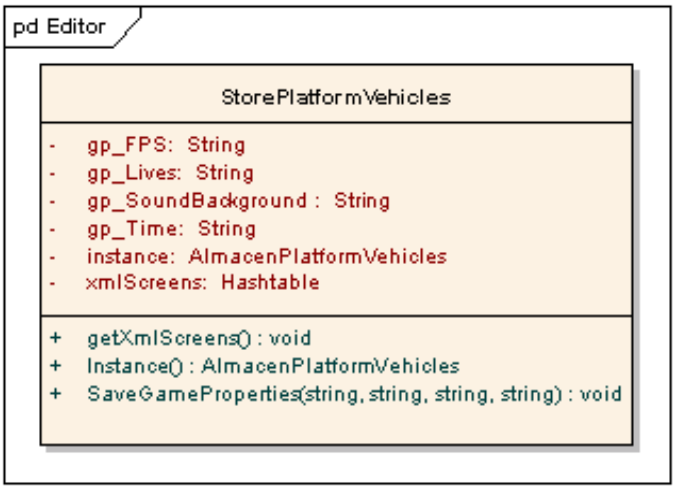


Ilustración 44. Diagrama de clases en la fase de diseño para la clase StorePlatformVehicles del paquete editor.

6.2.1.1.3 Diagrama de clases paquete editor: funcionalidad de serialización

Una parte importante de la funcionalidad del editor se basa en clases contenedoras que almacenan los datos de cada pantalla para volcar al finalizar la edición todo el contenido en los ficheros XML. Algunas de estas clases también se encargan de serializar esta información para guardar el estado de edición y recuperarlo.

En el diagrama posterior se han obviado las clases **TileSerializationPlatform** y **TargetSerializationPlatform** ya que no han sido implementadas en el ámbito de este proyecto sino que se han reaprovechado de la tipología de plataformas.

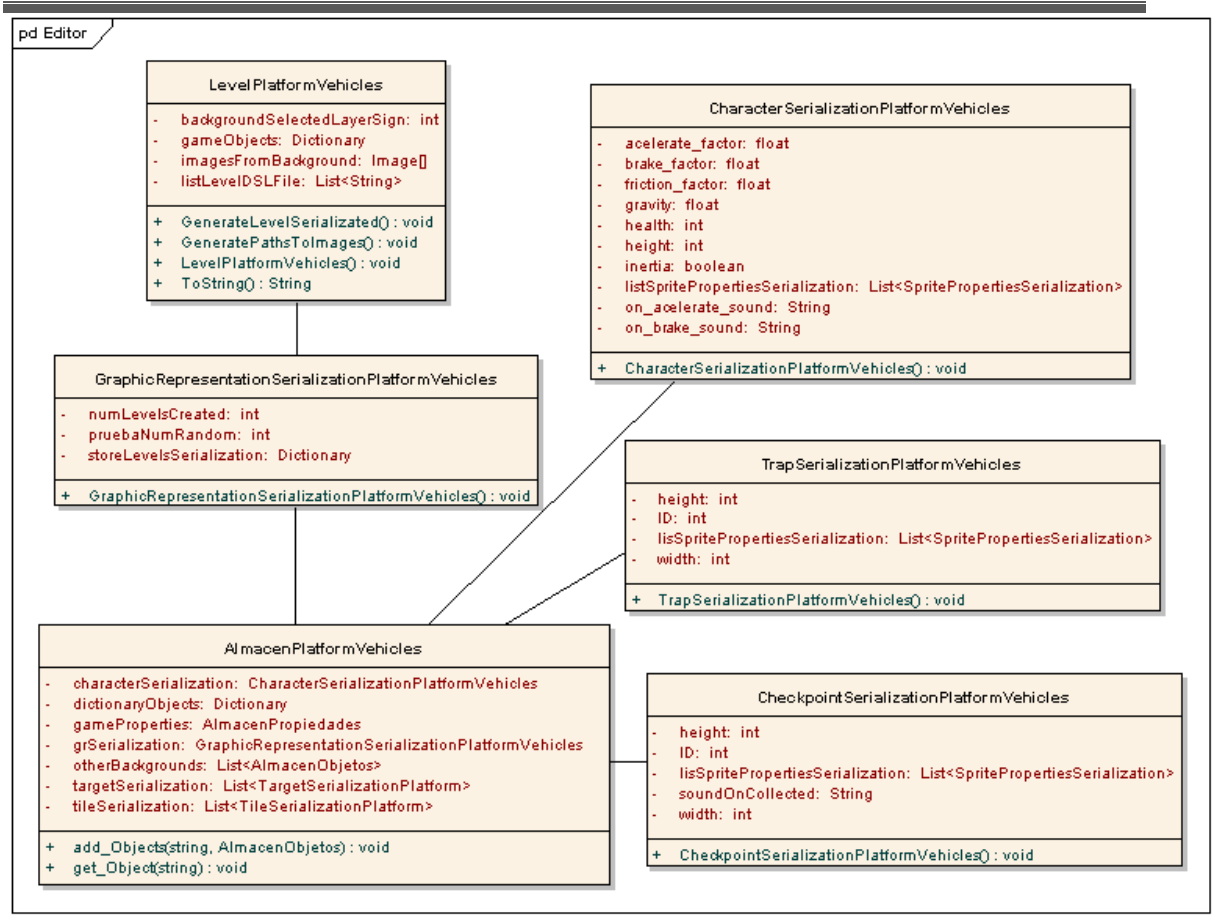


Ilustración 45. Diagrama de clases en la fase de diseño para la funcionalidad relacionada con la serialización del paquete editor.

6.2.1.2 Diagrama de Clases paquete Analizador

Clase encargada de recibir la definición el juego realizada mediante el GSL que se genera a partir del editor y sustituir esa información en la plantilla para obtener como salida un videojuego orientado al control de vehículos.

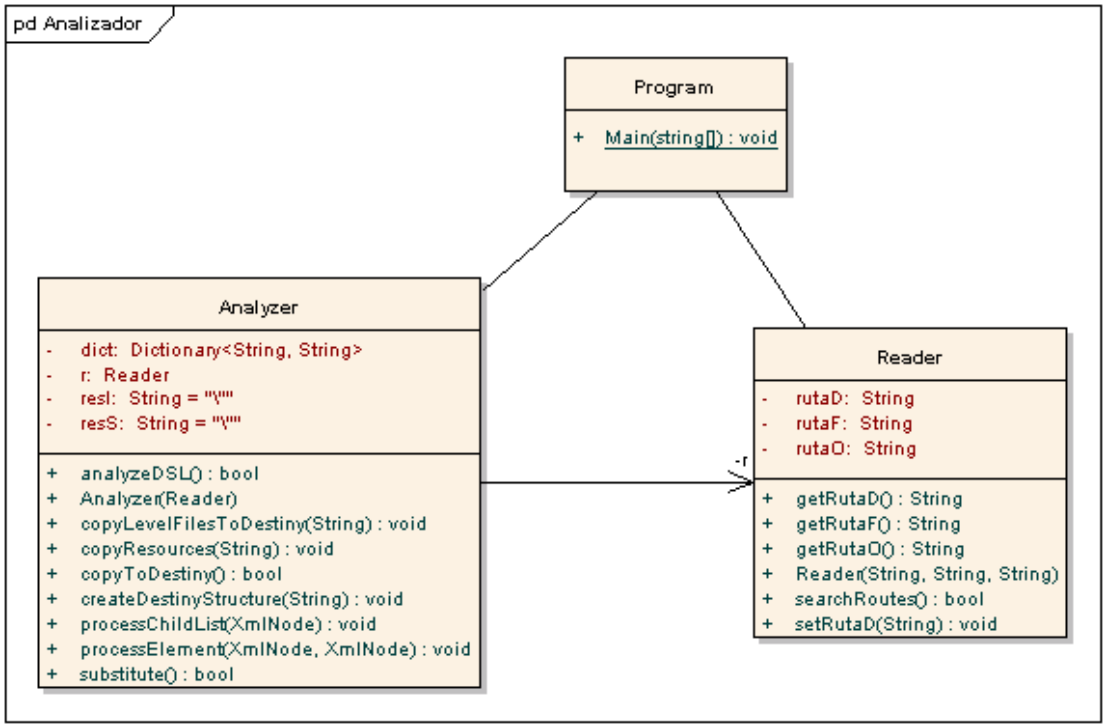


Ilustración 46. Diagrama de clases en la fase de diseño del analizador

6.2.1.3 Diagrama de Clases paquete Videojuego

Al igual que ocurría con el paquete del editor, el diagrama de clases general del paquete videojuego es demasiado extenso por lo que se separará en diagramas de clases más pequeños atendiendo a la funcionalidad de cada módulo, obviando aquellas clases de menor importancia o con implementación similar al resto de tipologías.

La clase DSL es la encargada de recibir los parámetros personalizables que sustituyó el analizador en la plantilla. No se incluye en el diagrama de clases por el alto número de atributos que posee, los cuales se analizarán en una posterior subsección de la sección de diseño.

6.2.1.3.1 Diagrama de clases paquete videojuego: controles

Los controles son aquellos elementos con los que interacciona el usuario durante el desarrollo del videojuego. Se definen en cada nivel.

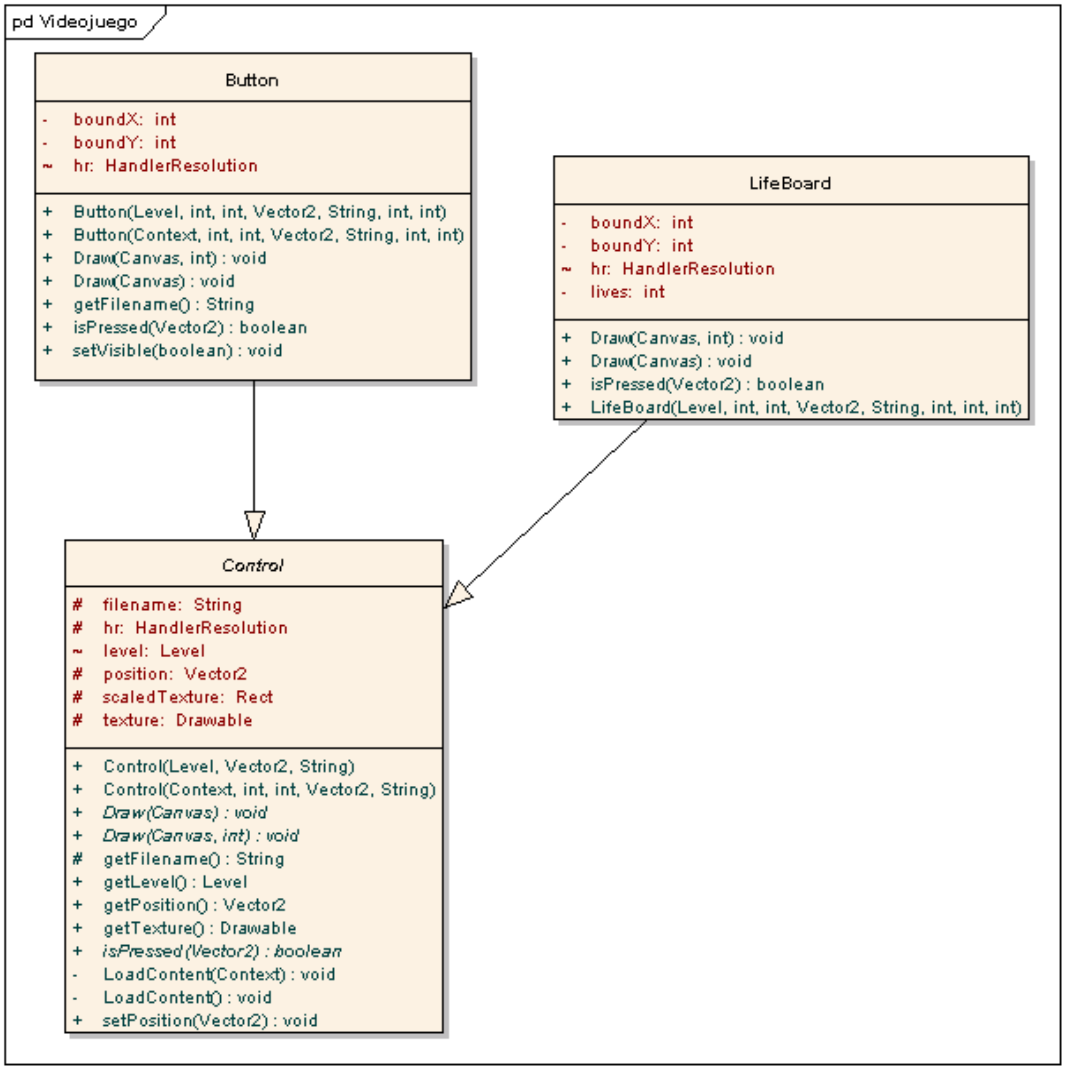


Ilustración 47. Diagrama de clases en la fase de diseño para los controles del paquete videojuego.

6.2.1.3.2 Diagrama de clases paquete videojuego: GameView, GameLoop, Level y LoaderLevel

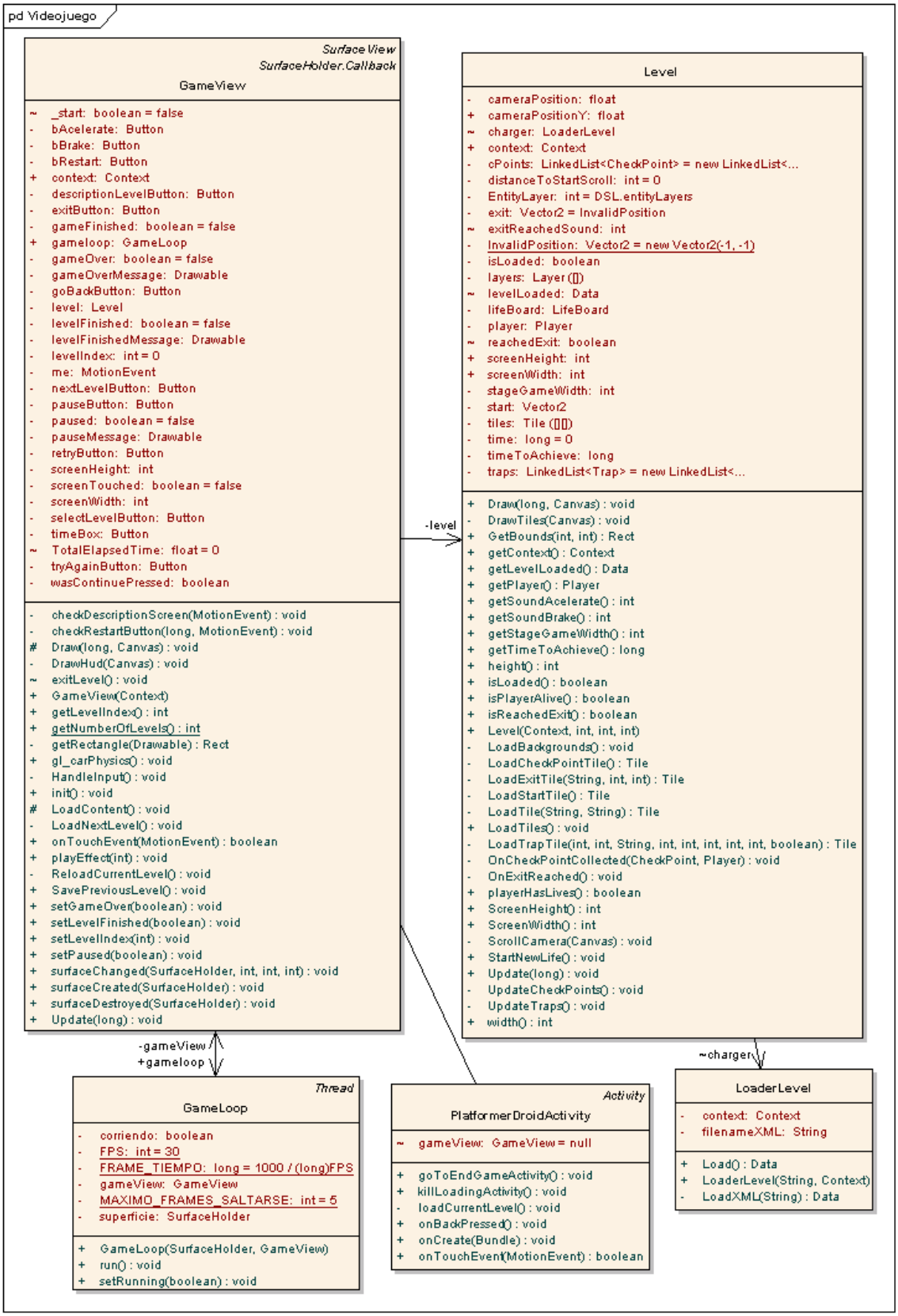


Ilustración 48. Diagrama de clases en la fase de diseño para GameView y Level

La clase **Level** encapsula todo el comportamiento de cada uno de los niveles que conforman el juego y almacena todos sus elementos (personaje principal, puntos de control, ladrillos, trampas, etc.). **LoaderLevel** se encarga de cargar toda la definición de un nivel desde su especificación XML.

El **GameLoop** es el hilo encargado de hacer que se actualice la lógica del juego mientras que el **GameView** es la vista del juego que recibe las pulsaciones de usuario y se encarga de gestionar los diferentes niveles para conocer el estado del juego (pausa, nivel perdido, nivel finalizado o juego finalizado). La actividad que lanza **GameView** es la **PlatformerDroidActivity.**

6.2.1.3.3 Diagrama de clases paquete videojuego: funcionalidad de Player

La clase **Player** es una de las más importantes del videojuego pues encapsula toda la lógica de movimiento del personaje principal. En el diagrama se han obviado algunos *getters* y *setters* de dicha clase para evitar que sea demasiado grande.

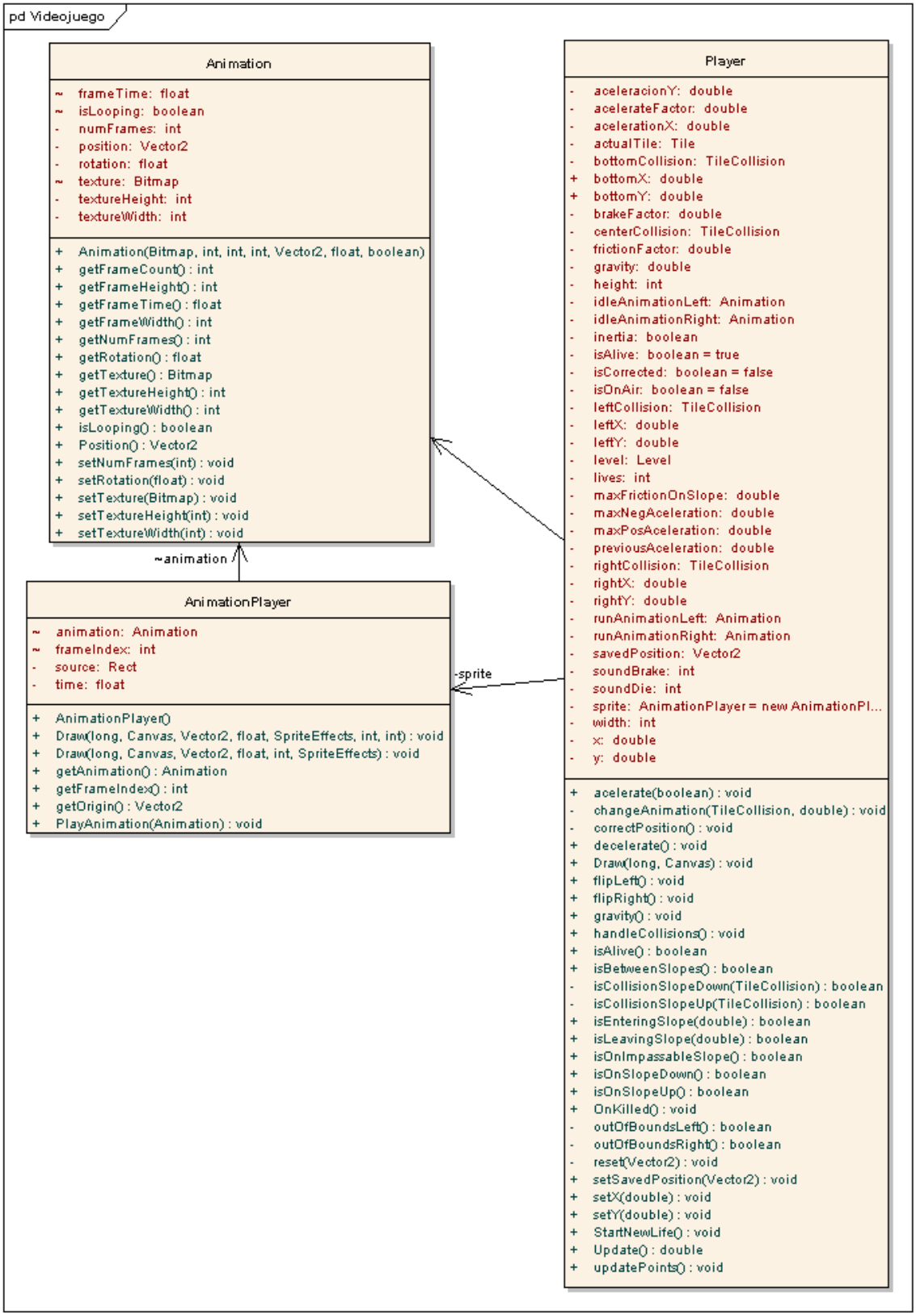


Ilustración 49. Diagrama de clases en la fase de diseño para Player en el videojuego

6.2.1.3.4 Diagrama de clases paquete videojuego: elementos del nivel

El siguiente diagrama de clases representa aquellos elementos que se disponen por el nivel (*Tiles,* trampas y puntos de control).

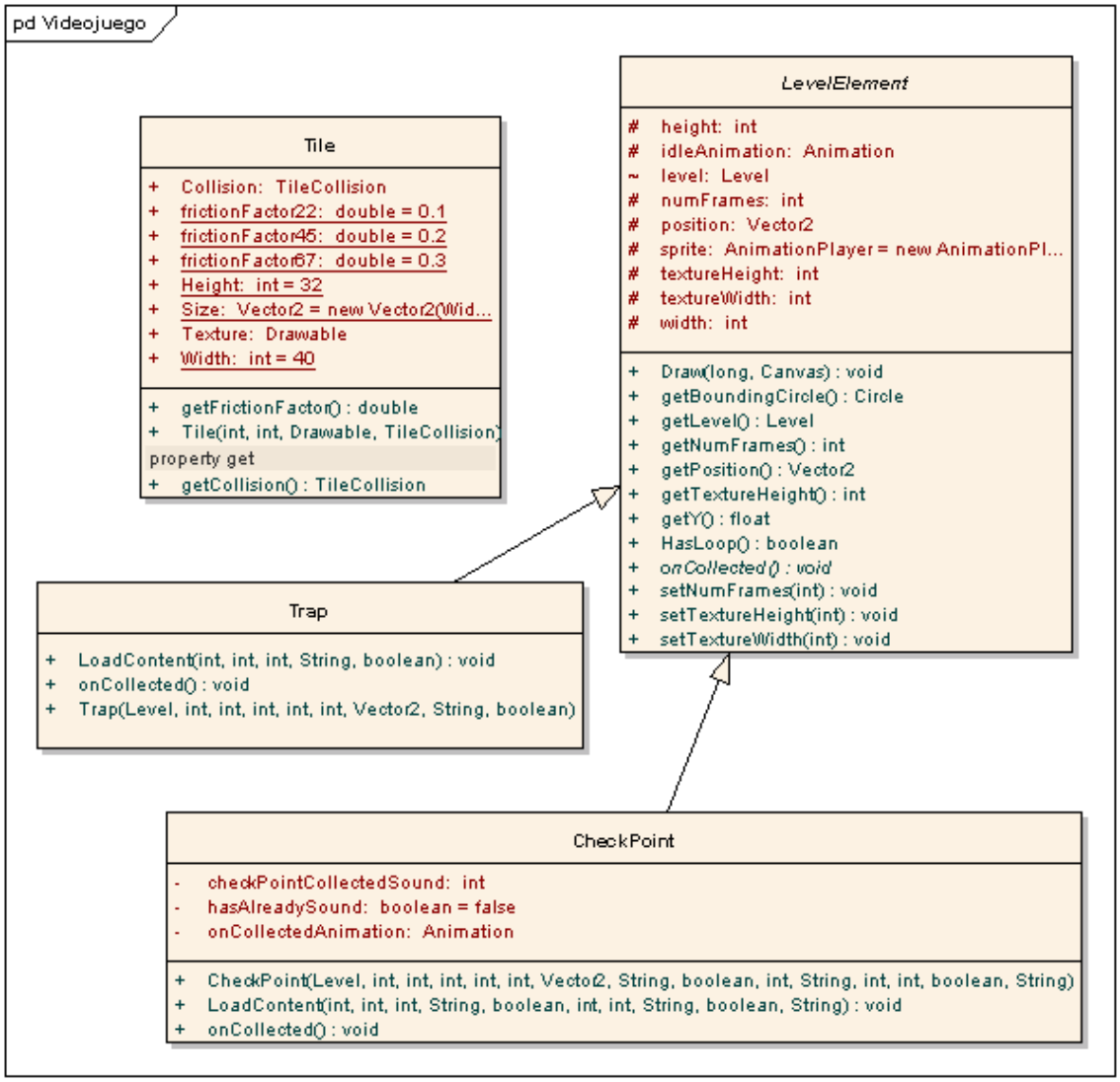


Ilustración 50. Diagrama de clases en la fase de diseño para los elementos del nivel del videojuego

6.2.1.3.5 Diagrama de clases paquete videojuego: pantallas

El siguiente diagrama de clases se corresponde con los *View* y *Activities* que implementan la vista y la funcionalidad de las pantallas previas al videojuego: pantalla principal, opciones, selección de nivel y la pantalla de fin de juego.

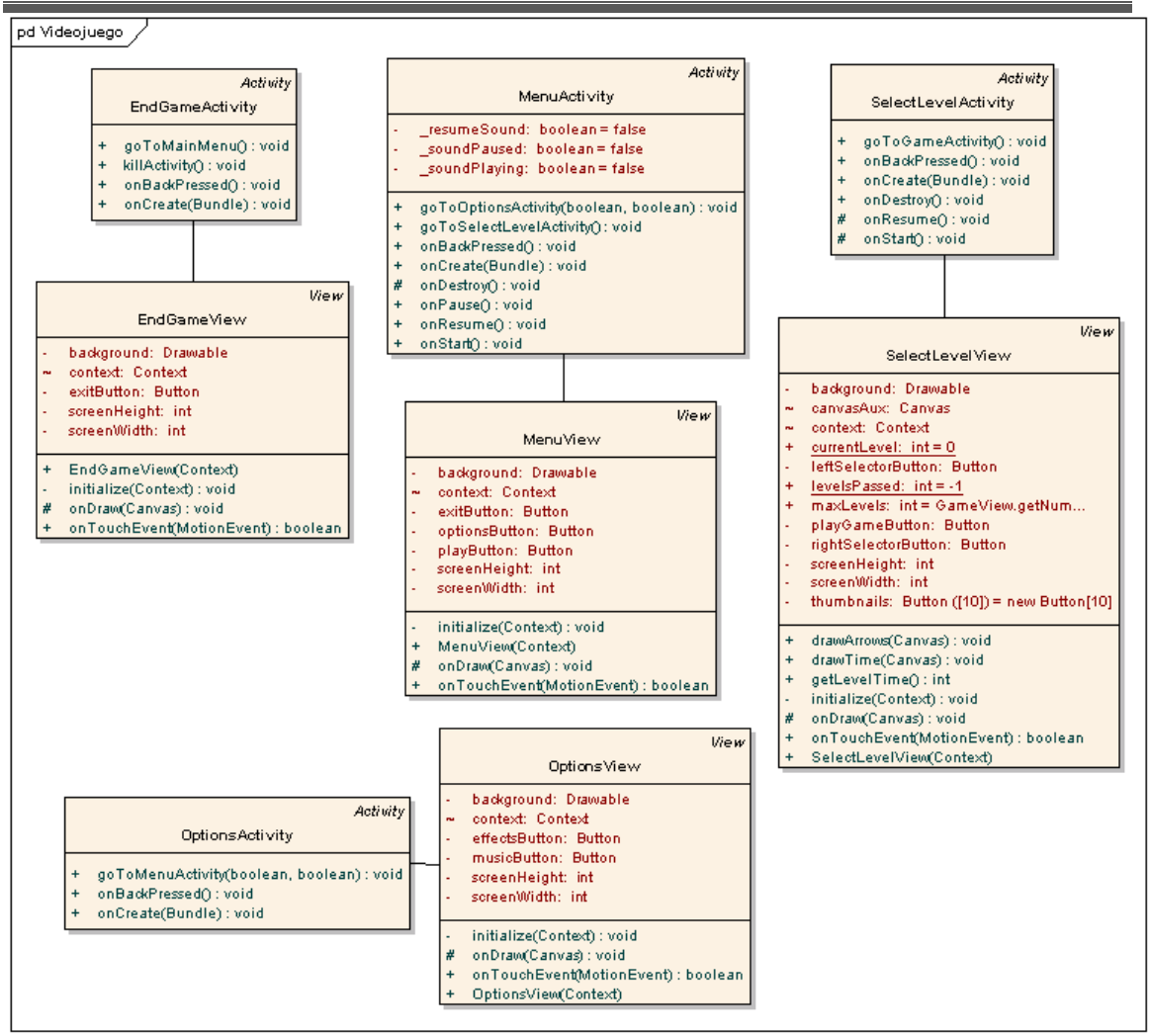


Ilustración 51. Diagrama de clases en la fase de diseño para las pantallas del videojuego

6.2.1.3.6 Diagrama de clases paquete videojuego: funcionalidad del sonido.

El siguiente diagrama de clases se corresponde con las clases encargadas de reproducir, guardar y gestionar los diferentes sonidos y músicas que se escucharán durante el transcurso del videojuego.

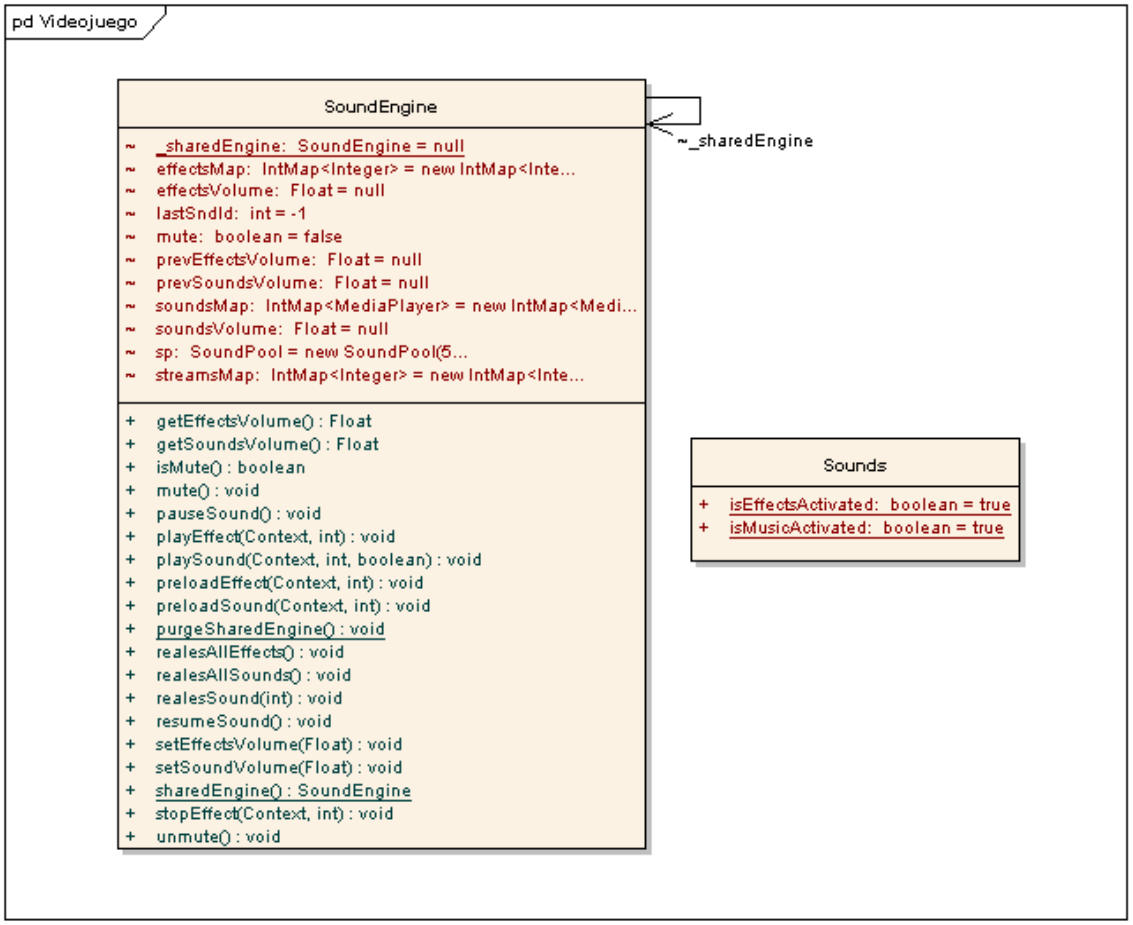


Ilustración 52. Diagrama de clases en la fase de diseño para la funcionalidad del sonido del videojuego

6.3 Diagramas de Interacción y Estados

6.3.1 Caso de Uso: Crear Videojuego

6.3.1.1 Diagramas de Interacción

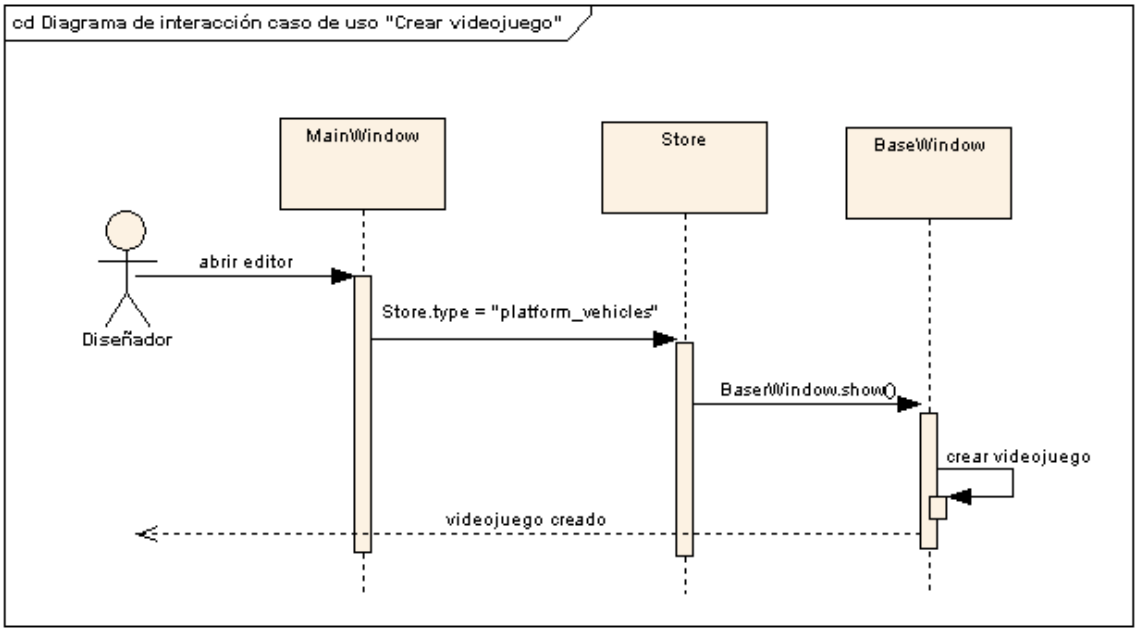


Ilustración 53. Diagrama de interacción para el caso de uso "Crear videojuego"

Cuando el usuario diseñador abre la aplicación y se muestra la pantalla de selección de tipología la clase **MainWindow** se encarga de gestionar la elección del usuario. Al hacer clic en la tipología de videojuegos orientados al control de vehículos se guarda en la **Store** el tipo de juego para que en **BaseWindow** se muestra una u otra lista de pantallas dependiendo de la tipología.

Ya que el caso de uso de crear un videojuego posee muchas fases, se ha decidido encapsular todo el proceso en el mensaje “crear videojuego” que se puede ver en el diagrama. Dicho proceso se detalla en el diagrama de actividad correspondiente en la sección posterior a la actual.

6.3.2 Caso de Uso: Guardar Videojuego

6.3.2.1 Diagramas de Interacción

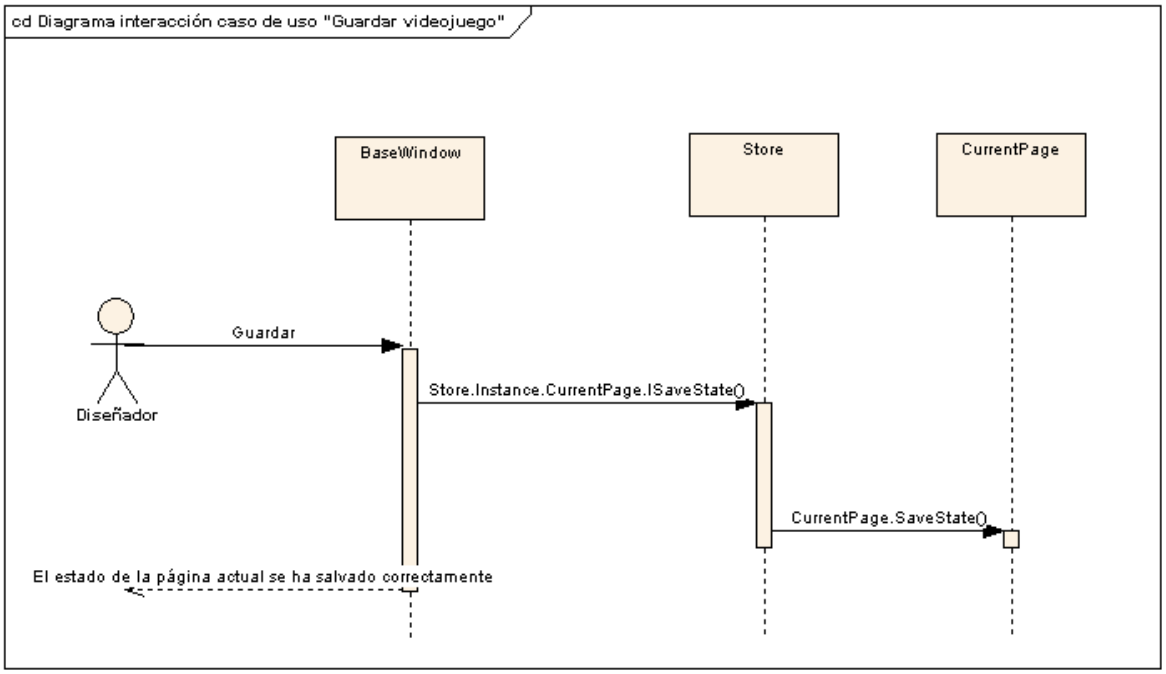


Ilustración 54. Diagrama de interacción para el caso de uso "Guardar videojuego"

El usuario diseñador puede guardar en cualquier momento el estado de la edición por lo que la **BaseWindow** se encargaría de acceder a la instancia del **Store** que almacena en la variable CurrentPage la pantalla actual. Cada una de las pantallas hereda de la clase **IPageStep** lo que las fuerza a implementar la función SaveState que es a la que se llama en última instancia para guardar el estado de la pantalla actual.

6.3.3 Caso de Uso: Cargar Videojuego

6.3.3.1 Diagramas de Interacción

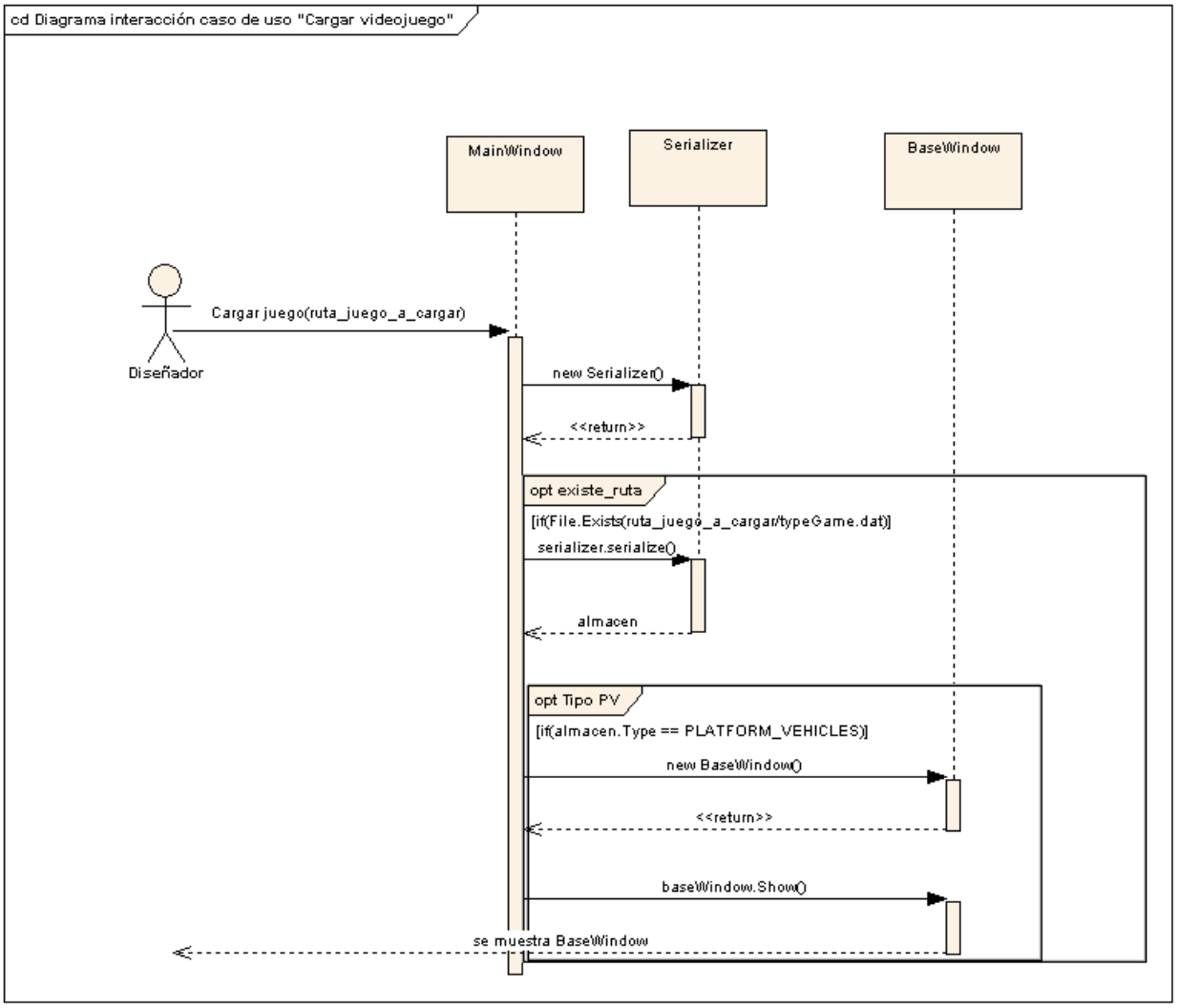


Ilustración 55. Diagrama de interacción para el caso de uso "Cargar videojuego"

En este caso la opción de cargar el estado de un videojuego previamente guardado sólo puede efectuarse desde la primera pantalla (MainWindow) Para ello el usuario escogerá la ruta donde se almacena un juego guardado y el Serializer obtendrá esta información a partir del archivo binario correspondiente.

6.3.4 Caso de Uso: Exportar Videojuego

6.3.4.1 Diagramas de Interacción

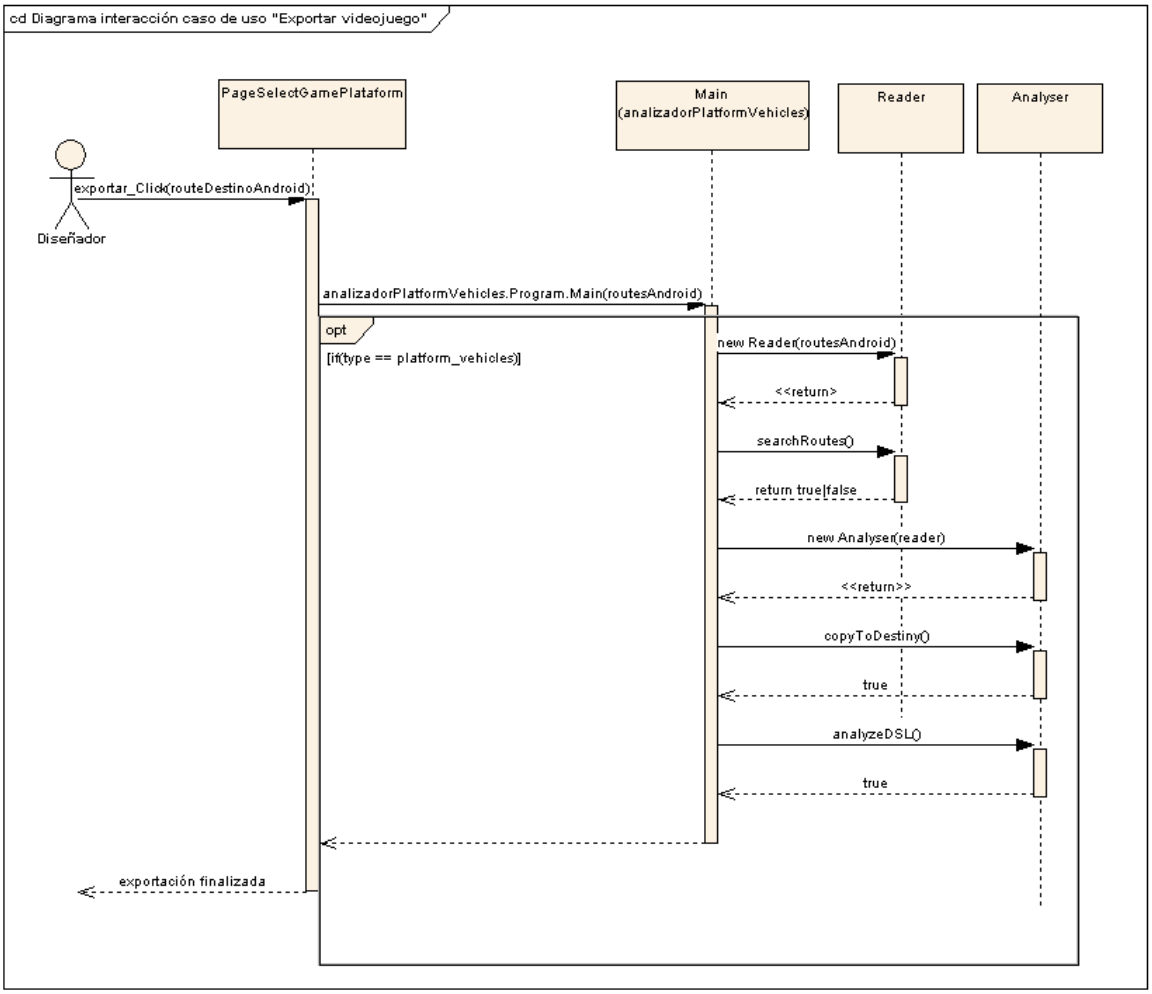


Ilustración 56. Diagrama de interacción para el caso de uso "Exportar videojuego"

Dependiendo de la tipología de juego (dato almacenado en la clase **Store**) se llama a uno u otro proyecto analizador de la solución global. En el caso del presente proyecto se llamaría a la clase main del proyecto **AnalizadorPlatformVehicles** donde se crearía una clase **Reader** que gestione la ruta donde se almacenan los archivos generados por el editor, la ruta donde se almacenan las plantillas de dicha tipología y la ruta donde se guardará el proyecto Android del videojuego. Con esas rutas, las plantillas y los datos guardados del editor el **Analyzer** a través de la clase analyseDSL procederá a realizar la sustitución donde sea necesario para obtener un proyecto Eclipse con el videojuego orientado al control de vehículos.

6.3.5 Caso de Uso: Jugar Videojuego

6.3.5.1 Diagramas de Interacción

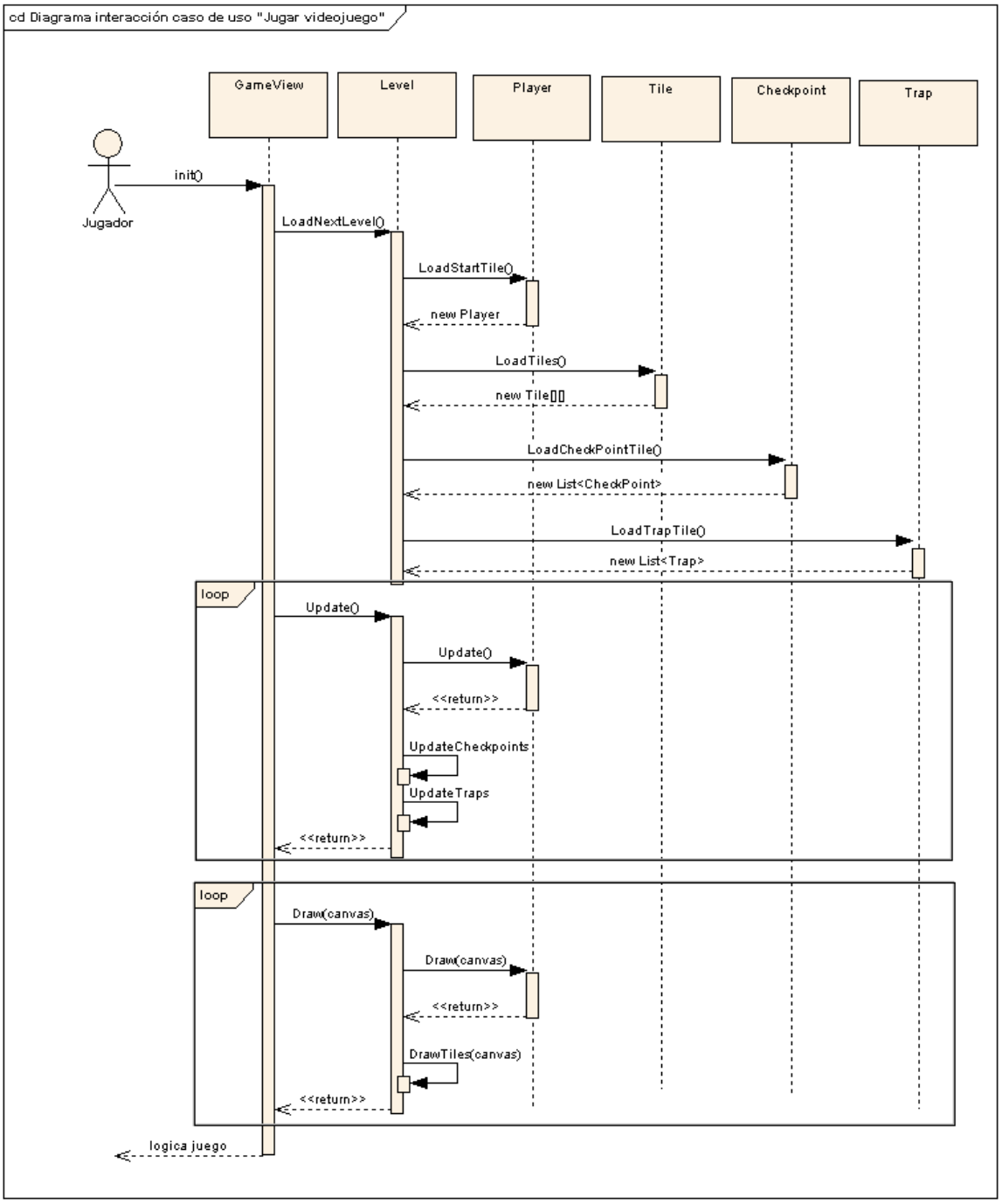


Ilustración 57. Diagrama de interacción para el caso de uso "Jugar videojuego"

El caso de uso “Jugar videojuego” involucra a todas las clases del subsistema videojuego. Dado que sería muy complejo contemplar todas ellas con sus posibilidades en un diagrama de interacción se ha decidido incluir solamente las más importantes.

Cuando el actor jugador inicia el videojuego, la clase **GameView** inicializa todos los elementos necesarios del nivel y carga el siguiente nivel a jugar. Dentro de la clase **Level** se inicializan los elementos del mismo: personaje con su clase **Player,** las trampas del nivel en su clase **Trap**, los puntos del control del nivel en la clase **Checkpoint** y los ladrillos, clase **Tile**. Una vez inicializados todos los elementos, la clase GameView actualiza periódicamente el nivel (que a su vez actualiza el personaje principal) y dibuja la situación actual en pantalla mediante la función Draw, que también se ejecuta en el nivel y en el personaje.

6.4 Diagramas de Actividades

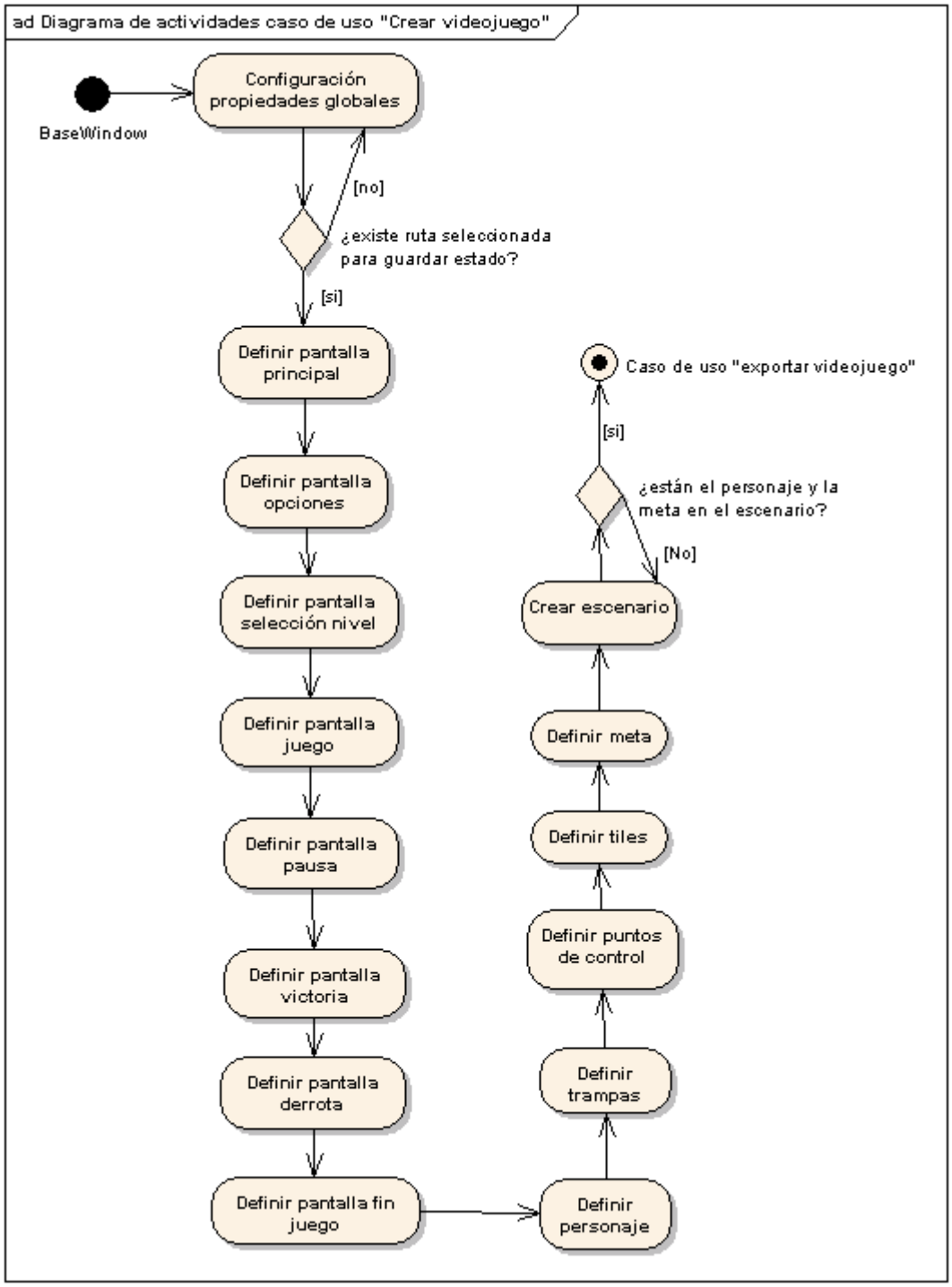


Ilustración 58. Diagrama de actividades del caso de uso "Crear videojuego"

6.5 Diseño del DSL

Esta sección describe la parte del DSL que define los niveles del juego. Éstos se representan a través de ficheros XML que deben ser alojados en la carpeta */assets* dentro del proyecto Eclipse final con el nombre Li.xml siendo i el número que indica el orden del nivel (de 0 a 9). Cada fichero XML representa un nivel del juego donde cada línea del mismo se corresponde con las características del nivel que se analizarán a continuación.

6.5.1 Elementos del DSL

6.5.1.1 Elemento level

Constituye la raíz del XML por lo que sólo puede aparecer una vez para delimitar el inicio y el fin de la definición del nivel en el DSL.

6.5.1.2 Elemento stage

Esta etiqueta define los parámetros generales del nivel. A continuación se enuncian y describen cada uno de los elementos que alberga.

6.5.1.2.1 Stage_id

Es el identificador del nivel.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.2 Stage_description

Descripción textual del nivel. Por defecto se inicializa con el nombre del juego.

Valores aceptados	Valor por defecto	Opcional
Alfanumérico	*Nombre del juego*	Sí

6.5.1.2.3 Stage_description_image_source

Contiene el nombre de la imagen que se muestra al inicio del nivel para ofrecer ayuda al usuario sobre cómo jugar.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	Sí

6.5.1.2.4 Stage_description_image_width

Ancho de la imagen que se presenta como ayuda al comenzar un nivel.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.5 Stage_description_image_height

Alto de la imagen que se presenta como ayuda al comenzar un nivel.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.6 Stage_description_image_x_position

Posición en el eje horizontal de la imagen que se presenta como ayuda al comenzar un nivel.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.7 Stage_description_image_y_position

Posición en el eje vertical de la imagen que se presenta como ayuda al comenzar un nivel.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.8 Stage_thumbnail_image_source

Nombre de la imagen de vista previa que aparece en la pantalla de selección de nivel representando a dicho nivel.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	Sí

6.5.1.2.9 Stage_thumbnail_image_width

Anchura de la imagen de vista previa del nivel.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.10 Stage_thumbnail_image_height

Altura de la imagen de vista previa del nivel.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.11 Stage_thumbnail_image_has_text

Indica si la imagen de vista previa del nivel va acompañada de un texto.

Valores aceptados	Valor por defecto	Opcional
YES| NO	Ninguno	Sí

6.5.1.2.12 Stage_game_width

Anchura total del nivel

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.13 Stage_game_height

Altura total del nivel

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.14 Stage_background_layer_back_image_source

Ruta de archivo de la imagen que define el fondo de la pantalla del videojuego. Se trata del fondo más alejado del primer plano del juego, efecto que se consigue dándole un movimiento lento a la capa.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.15 Stage_background_layer_back_image_width

Anchura de la imagen que define el fondo del juego más alejado.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.16 Stage_background_layer_back_image_height

Altura de la imagen que define el fondo del juego más alejado.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.17 Stage_background_layer_back_x_speed

Velocidad de la capa de la imagen que define el fondo del juego más alejado. El fondo se moverá con la velocidad indicada cuando lo haga el personaje principal en el eje horizontal.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.18 Stage_background_layer_medium_image_source

Ruta de archivo de la imagen que define el fondo de la pantalla del videojuego. Se trata del fondo de velocidad media respecto al primer plano del juego.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	Sí

6.5.1.2.19 Stage_background_layer_medium_image_width

Anchura de la imagen que define el fondo del juego de profundidad media.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.20 Stage_background_layer_medium_image_height

Altura de la imagen que define el fondo del juego de profundidad media.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.21 Stage_background_layer_medium_x_speed

Velocidad de la capa de la imagen que define el fondo de profundidad media. El fondo se moverá con la velocidad indicada cuando lo haga el personaje principal en el eje horizontal.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.22 Stage_background_layer_front_image_source

Ruta de archivo de la imagen que define el fondo de la pantalla del videojuego. Se trata del fondo más próximo respecto al primer plano del juego.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	Sí

6.5.1.2.23 Stage_background_layer_front_image_width

Anchura de la imagen que define el fondo del juego de menor profundidad.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.24 Stage_background_layer_front_image_height

Altura de la imagen que define el fondo del juego de menor profundidad.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.25 Stage_background_layer_front_x_speed

Velocidad de la capa de la imagen que define el fondo de menor profundidad. El fondo se moverá con la velocidad indicada cuando lo haga el personaje principal en el eje horizontal.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	Sí

6.5.1.2.26 Stage_time_to_achieve

Tiempo en milisegundos que tiene el personaje principal para ir desde el punto de salida a la meta. Si este valor es igual a cero significará que el personaje principal no tiene que cumplir ningún límite de tiempo, le bastará con llegar a la meta del nivel para superarlo.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.27 Stage_fps

Número de fotogramas por segundo que definirán cuándo actualizar la lógica del juego y cuándo dibujar y mover los personajes dentro del mismo.

Valores aceptados	Valor por defecto	Opcional
Numérico	15	No

6.5.1.2.28 Stage_lives

Número de vidas totales que tendrá el personaje para llegar a la meta del nivel. Su valor sólo puede estar entre uno y cinco.

Valores aceptados	Valor por defecto	Opcional
Numérico	3	No

6.5.1.2.29 Stage_sound_music_background_source

Ruta de archivo del sonido que sonará como música ambiente a lo largo del desarrollo del juego.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	Sí

6.5.1.2.30 Stage_sound_character_on_exit_reached_source

Ruta de archivo del sonido que sonará cuando el personaje principal llegue a la meta del nivel.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	Sí

6.5.1.2.31 Elemento character

Este elemento engloba todas las propiedades que definen al personaje principal, es decir, el elemento del nivel con el que interacciona el usuario. Se definen sus subelementos a continuación. Es único dentro del nivel y tiene que aparecer de forma obligatoria.

6.5.1.2.32 Character_name

Nombre identificativo del jugador. No aparece en el juego.

Valores aceptados	Valor por defecto	Opcional
Alfanumérico	Ninguno	No

6.5.1.2.33 Character_x_position

Posición en el eje horizontal del que partirá el personaje principal al comenzar el nivel.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.34 Character_y_position

Posición en el eje vertical del que partirá el personaje principal al comenzar el nivel.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.35 Character_width

Anchura del personaje principal.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.36 Character_height

Altura del personaje principal.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.2.37 Character_collision_area_left

Sirve para definir el punto de colisión izquierdo del personaje principal en el caso de que el Drawable elegido para el mismo no esté ajustado.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.38 Character_collision_area_top

Sirve para definir el punto de colisión superior del personaje principal en el caso de que el Drawable elegido para el mismo no esté ajustado.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.39 Character_collision_area_right

Sirve para definir el punto de colisión derecho del personaje principal en el caso de que el Drawable elegido para el mismo no esté ajustado.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.40 Character_collision_area_down

Sirve para definir el punto de colisión inferior del personaje principal en el caso de que el Drawable elegido para el mismo no esté ajustado.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.41 Character_sound_acelerate_source

Ruta del archivo de sonido que sonará cuando se pulse el botón de acelerar.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	Sí

6.5.1.2.42 Character_sound_brake_source

Ruta del archivo de sonido que sonará cuando se pulse el botón de frenado.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	Sí

6.5.1.2.43 Character_sound_die_source

Ruta del archivo de sonido que sonará cuando el personaje principal pierda una vida.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	Sí

6.5.1.2.44 Character_acelerate_factor

Factor de aceleración que se aplicará al personaje principal cuando se mueva a la derecha. Cuanto mayor sea este valor más rápido avanzará a la derecha cuando se pulse el botón de aceleración.

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.45 Character_brake_factor

Factor de desaceleración que se aplicará al personaje principal cuando se mueva a la izquierda. Cuanto mayor sea este valor más rápido avanzará a la izquierda cuando se pulsen el botón de frenado.

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.46 Character_friction_factor

Factor de fricción que se aplicará al personaje principal cuando tenga una aceleración y no se esté pulsando ningún botón. De esta manera, el personaje no se parará de golpe sino que dejará de avanzar transcurrido un tiempo.

Valores aceptados	Valor por defecto	Opcional
Numérico	0.1	No

6.5.1.2.47 Character_max_positive_aceleration

Aceleración máxima que puede alcanzar el personaje principal cuando se mueve a la derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	5	No

6.5.1.2.48 Character_max_negative_aceleration

Aceleración máxima que puede alcanzar el personaje principal cuando se mueve a la izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	5	No

6.5.1.2.49 Character_gravity

Factor de gravedad que atrae al personaje principal hacia el suelo.

Valores aceptados	Valor por defecto	Opcional
Numérico	3	No

6.5.1.2.50 Character_inertia

Define si el personaje principal sentirá la atracción de la gravedad cuando esté parado encima de una rampa.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.51 Character_sprite_default_left_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje esté parado mirando hacia la izquierda.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.52 Character_sprite_default_left_width

Anchura del *sprite* cuando el personaje esté parado mirando hacia la izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.53 Character_sprite_default_left_height

Altura del *sprite* cuando el personaje esté parado mirando hacia la izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.54 Character_sprite_default_left_frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.55 Character_sprite_default_left_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.56 Character_sprite_default_right_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje esté parado mirando hacia la derecha.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.57 Character_sprite_default_right_width

Anchura del *sprite* cuando el personaje esté parado mirando hacia la derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.58 Character_sprite_default_right_height

Altura del *sprite* cuando el personaje esté parado mirando hacia la derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.59 Character_sprite_default_right_frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.60 Character_sprite_default_right_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.61 Character_sprite_left_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje está en movimiento hacia la izquierda.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.62 Character_sprite_left_width

Anchura del *sprite* cuando el personaje está en movimiento hacia la izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.63 Character_sprite_left_height

Altura del *sprite* cuando el personaje está en movimiento hacia la izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.64 Character_sprite_left_frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.65 Character_sprite_left_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.66 Character_sprite_right_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje está en movimiento hacia la derecha.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.67 Character_sprite _right _width

Anchura del *sprite* cuando el personaje está en movimiento hacia la derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.68 Character_sprite_right _height

Altura del *sprite* cuando el personaje está en movimiento hacia la derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.69 Character_sprite _right _frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.70 Character_sprite_right_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.71 Character_sprite_slope22up_left_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 22º ascendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.72 Character_ sprite_slope22up_left_width

Anchura del *sprite* cuando el personaje sube una rampa de 22º ascendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.73 Character_ sprite_slope22up_left_height

Altura del *sprite* cuando el personaje sube una rampa de 22º ascendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.74 Character_ sprite_slope22up_left_frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.75 Character_ sprite_slope22up_left_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.76 Character_sprite_slope22up_right_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 22º ascendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.77 Character_ sprite_slope22up_right_width

Anchura del *sprite* cuando el personaje sube una rampa de 22º ascendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.78 Character_ sprite_slope22up_right_height

Altura del *sprite* cuando el personaje sube una rampa de 22º ascendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.79 Character_ sprite_slope22up_right _frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.80 Character_ sprite_slope22up_right_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.81 Character_sprite_slope22down_left_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 22º descendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.82 Character_ sprite_slope22down_left_width

Anchura del *sprite* cuando el personaje sube una rampa de 22º descendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.83 Character_ sprite_slope22down_left_height

Altura del *sprite* cuando el personaje sube una rampa de 22º descendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.84 Character_ sprite_slope22down_left_frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.85 Character_ sprite_slope22down_left_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.86 Character_sprite_slope22down_right_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 22º descendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.87 Character_ sprite_slope22down_right_width

Anchura del *sprite* cuando el personaje sube una rampa de 22º descendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.88 Character_ sprite_slope22down_right_height

Altura del *sprite* cuando el personaje sube una rampa de 22º descendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.89 Character_ sprite_slope22down_right _frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.90 Character_ sprite_slope22down_right_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.91 Character_sprite_slope45up_left_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 45º ascendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.92 Character_ sprite_slope45up_left_width

Anchura del *sprite* cuando el personaje sube una rampa de 45º ascendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.93 Character_ sprite_slope45up_left_height

Altura del *sprite* cuando el personaje sube una rampa de 45º ascendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.94 Character_ sprite_slope45up_left_frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.95 Character_ sprite_slope45up_left_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.96 Character_sprite_slope45up_right_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 45º ascendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.97 Character_ sprite_slope45up_right_width

Anchura del *sprite* cuando el personaje sube una rampa de 45º ascendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.98 Character_ sprite_slope45up_right_height

Altura del *sprite* cuando el personaje sube una rampa de 45º ascendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.99 Character_ sprite_slope45up_right _frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.100 Character_ sprite_slope45up_right_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.101 Character_sprite_slope45down_left_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 45º descendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.102 Character_ sprite_slope45down_left_width

Anchura del *sprite* cuando el personaje sube una rampa de 45º descendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.103 Character_ sprite_slope45down_left_height

Altura del *sprite* cuando el personaje sube una rampa de 45º descendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.104 Character_ sprite_slope45down_left_frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.105 Character_ sprite_slope45down_left_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.106 Character_sprite_slope45down_right_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 45º descendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.107 Character_ sprite_slope45down_right_width

Anchura del *sprite* cuando el personaje sube una rampa de 45º descendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.108 Character_ sprite_slope45down_right_height

Altura del *sprite* cuando el personaje sube una rampa de 45º descendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.109 Character_ sprite_slope45down_right _frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.110 Character_ sprite_slope45down_right_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.111 Character_sprite_slope67up_left_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 67º ascendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.112 Character_ sprite_slope67up_left_width

Anchura del *sprite* cuando el personaje sube una rampa de 67º ascendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.113 Character_ sprite_slope67up_left_height

Altura del *sprite* cuando el personaje sube una rampa de 67º ascendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.114 Character_ sprite_slope67up_left_frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.115 Character_ sprite_slope67up_left_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.116 Character_sprite_slope67up_right_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 67º ascendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.117 Character_ sprite_slope67up_right_width

Anchura del *sprite* cuando el personaje sube una rampa de 67º ascendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.118 Character_ sprite_slope67up_right_height

Altura del *sprite* cuando el personaje sube una rampa de 67º ascendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.119 Character_ sprite_slope67up_right _frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.120 Character_ sprite_slope67up_right_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.121 Character_sprite_slope67down_left_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 67º descendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.122 Character_ sprite_slope67down_left_width

Anchura del *sprite* cuando el personaje sube una rampa de 67º descendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.123 Character_ sprite_slope67down_left_height

Altura del *sprite* cuando el personaje sube una rampa de 67º descendente y va en dirección izquierda.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.124 Character_ sprite_slope67down_left_frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.125 Character_ sprite_slope67down_left_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.2.126 Character_sprite_slope67down_right_source

Ruta del *sprite* (conjunto de imágenes) a mostrar cuando el personaje sube una rampa de 67º descendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.2.127 Character_ sprite_slope67down_right_width

Anchura del *sprite* cuando el personaje sube una rampa de 67º descendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Ancho del personaje*	No

6.5.1.2.128 Character_ sprite_slope67down_right_height

Altura del *sprite* cuando el personaje sube una rampa de 67º descendente y va en dirección derecha.

Valores aceptados	Valor por defecto	Opcional
Numérico	*Alto del personaje*	No

6.5.1.2.129 Character_ sprite_slope67down_right _frames

Número de fotogramas por segundo del *sprite.*

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.2.130 Character_ sprite_slope67down_right_has_loop

Indica si el *sprite* debe mostrarse de forma secuencial o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.3 Elemento trap

Este elemento engloba todas las propiedades que definen una trampa del nivel, es decir, un obstáculo que el personaje principal no puede atravesar y que hará que éste pierda una vida. Puede haber varias en un mismo nivel o no haber ninguna.

6.5.1.3.1 trap_name

Nombre de la trampa. No aparecerá en el juego.

Valores aceptados	Valor por defecto	Opcional
Alfanumérico	Ninguno	No

6.5.1.3.2 trap_x_position

Posición de la trampa en el eje horizontal.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.3.3 trap_y_position

Posición de la trampa en el eje vertical.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.3.4 trap_width

Anchura de la trampa.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.3.5 trap_height

Altura de la trampa.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.3.6 trap_sprite_default_source

Ruta del archivo del *sprite* que definirá la trampa. Estará formada por una única animación.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.3.7 trap_sprite_default_width

Anchura del *sprite* de la trampa.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.3.8 trap_sprite_default_height

Altura del *sprite* de la trampa.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.3.9 trap_sprite_default_frames

Número de fotogramas que conforman el *sprite* de la trampa.

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.3.10 trap_sprite_default_has_loop

Indica si el *sprite* que define la trampa debe ser ejecutado secuencialmente o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.4 Elemento checkpoint

Este elemento engloba todas las propiedades que definen un punto de control del nivel. Su comportamiento será hacer aparecer al personaje principal en el último punto colisionado. Puede haber varios en un mismo nivel o no haber ninguno.

6.5.1.4.1 checkpoint_name

Nombre del punto de control. No aparecerá en el juego.

Valores aceptados	Valor por defecto	Opcional
Alfanumérico	Ninguno	No

6.5.1.4.2 checkpoint_x_position

Posición del punto de control en el eje horizontal.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.4.3 checkpoint_y_position

Posición del punto de control en el eje vertical.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.4.4 checkpoint_width

Anchura del punto de control.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.4.5 checkpoint_height

Altura del punto de control.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.4.6 checkpoint _on_collected_source

Ruta del archivo de sonido que se reproducirá cuando el personaje active un punto de control.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	Sí

6.5.1.4.7 checkpoint_sprite_default_source

Ruta del archivo del *sprite* que definirá la animación del punto de control cuando esté desactivado (animación por defecto).

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.4.8 checkpoint_sprite_default _width

Anchura del *sprite* del punto de control desactivado.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.4.9 checkpoint_sprite_default_height

Altura del *sprite* del punto de control desactivado.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.4.10 checkpoint_sprite_default_frames

Número de fotogramas que conforman el *sprite* del punto de control desactivado.

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.4.11 checkpoint_sprite_default_has_loop

Indica si el *sprite* que define el punto de control desactivado debe ser ejecutado secuencialmente o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.4.12 checkpoint_sprite_on_collected_source

Ruta del archivo del *sprite* que se mostrará cuando el punto de control esté activo.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.4.13 checkpoint_sprite_on_collected_width

Anchura de la imagen que representa un punto de control activo.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.4.14 checkpoint_sprite_on_collected_height

Altura del *sprite* que representa un punto de control activo.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.4.15 checkpoint_sprite_on_collected_frames

Número de fotogramas que conforman el *sprite* de un punto de control activo.

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.4.16 checkpoint_sprite_on_collected_has_loop

Indica si el *sprite* que define el punto de control en modo activo debe ser ejecutado secuencialmente o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.5 Elemento target

Este elemento representa la meta a la que debe llegar el jugador para finalizar el nivel. Debe ser único y aparecer de forma obligatoria en el nivel.

6.5.1.5.1 Target_name

Nombre del objetivo. No aparecerá en el juego.

Valores aceptados	Valor por defecto	Opcional
Alfanumérico	Ninguno	No

6.5.1.5.2 Target_x_position

Posición del objetivo en el eje horizontal.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.5.3 Target_y_position

Posición del objetivo en el eje vertical.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.5.4 Target_width

Anchura del objetivo.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.5.5 Target_height

Altura del objetivo.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.5.6 target_image_source

Ruta del archivo de la imagen que definirá el objetivo.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.5.7 target_image_width

Anchura de la imagen del objetivo.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.5.8 target_image_height

Altura de la imagen del objetivo.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.5.9 target_image_frames

Número de fotogramas que conforman el *sprite* del objetivo.

Valores aceptados	Valor por defecto	Opcional
Numérico	1	No

6.5.1.5.10 target_image_has_loop

Indica si el *sprite* que define el objetivo debe ser ejecutado secuencialmente o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	NO	No

6.5.1.6 Elemento Tile

Este elemento representa los ladrillos que crean las estructuras de los niveles. Aparecerá repetidas veces en el mismo nivel.

6.5.1.6.1 Tile_name

Nombre del ladrillo. No aparecerá en el juego.

Valores aceptados	Valor por defecto	Opcional
Alfanumérico	Ninguno	No

6.5.1.6.2 Tile_x_position

Posición del ladrillo en el eje horizontal.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.6.3 Tile_y_position

Posición del ladrillo en el eje vertical.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.6.4 Tile_width

Anchura del ladrillo.

Valores aceptados	Valor por defecto	Opcional
Numérico	40	No

6.5.1.6.5 Tile_height

Altura del ladrillo.

Valores aceptados	Valor por defecto	Opcional
Numérico	32	No

6.5.1.6.6 Tile_image_source

Ruta del archivo del *sprite* que definirá el ladrillo.

Valores aceptados	Valor por defecto	Opcional
Ruta de archivo	Ninguno	No

6.5.1.6.7 Tile_image_width

Anchura del *sprite* del ladrillo.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.6.8 Tile_image_height

Altura del *sprite* del ladrillo.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.6.9 Tile_collision

Tipo de colisión del ladrillo.

Valores aceptados	Valor por defecto	Opcional
Alfanumérico	Impassable	No

6.5.1.6.10 Tile_image_frames

Número de fotogramas que conforman el *sprite* del ladrillo.

Valores aceptados	Valor por defecto	Opcional
Numérico	Ninguno	No

6.5.1.6.11 Tile_image_has_loop

Indica si el *sprite* que define el ladrillo debe ser ejecutado secuencialmente o en bucle.

Valores aceptados	Valor por defecto	Opcional
YES | NO	Ninguno	No

6.6 Diseño de la Interfaz

A continuación se mostrará el diseño de las interfaces definitivas de los diferentes subsistemas.

6.6.1 Interfaz del editor

Para navegar entre las diferentes pantallas se han incluido los botones de "siguiente y anterior" situadas en la parte inferior de la pantalla. También se puede acceder a una pantalla directamente haciendo clic en su nombre en la parte superior de la interfaz.

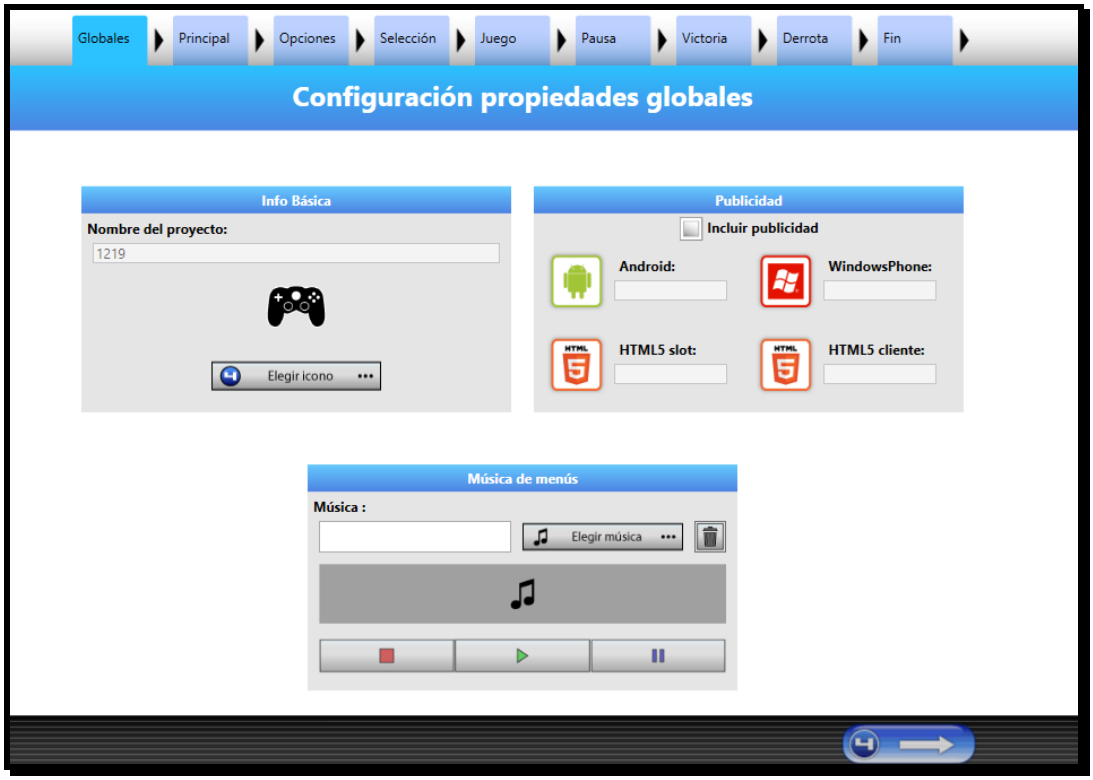


Ilustración 59. Pantalla definitiva del editor para configurar de propiedades globales

Ilustración 60. Pantalla definitiva del editor para configurar pantallas

En este caso se adjunta únicamente la pantalla de configurar la pantalla principal, pues la configuración de la pantalla de opciones, selección de nivel, juego, pausa, victoria, derrota y fin son análogas salvo en la pantalla que se ofrece por defecto.

Ilustración 61. Pantalla definitiva del editor para configurar el personaje principal

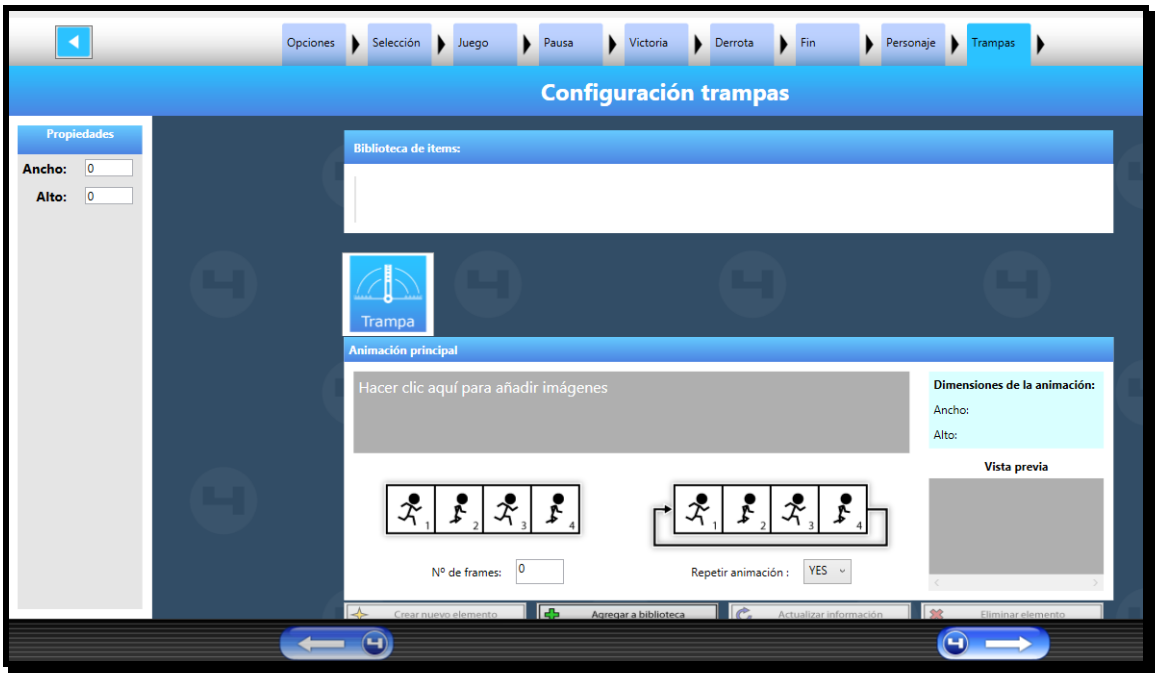


Ilustración 62. Pantalla definitiva del editor para configurar elementos del nivel

En este caso se adjunta únicamente la pantalla de configurar las trampas, pues la configuración de puntos de control, *tiles* y metas son prácticamente análogas salvo en las propiedades.

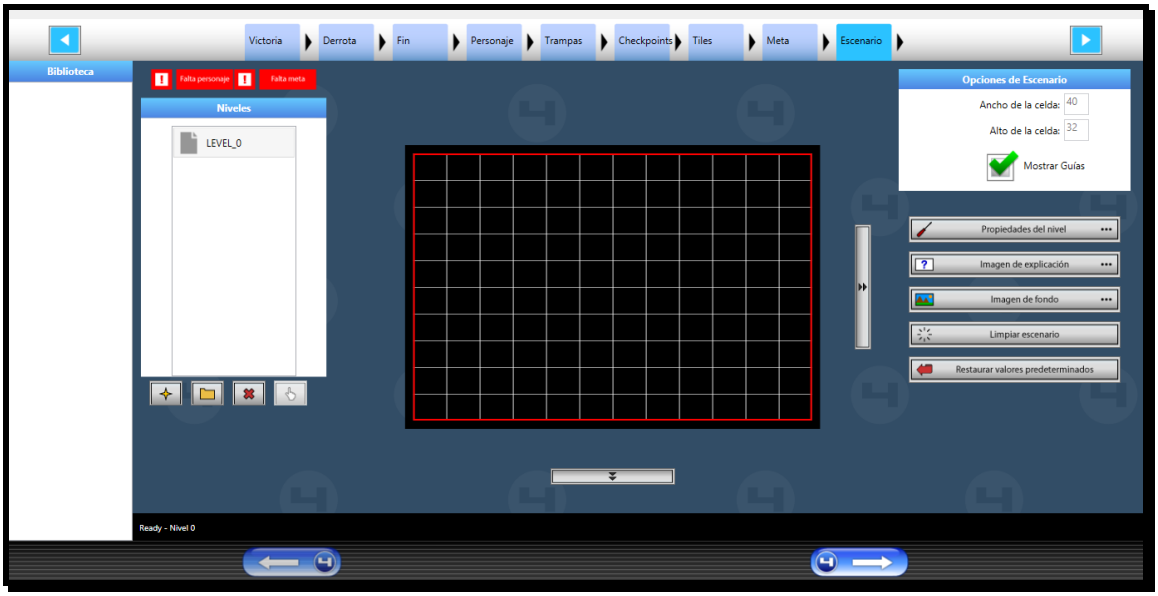


Ilustración 63. Pantalla definitiva de configuración de escenario

Ilustración 64. Pantalla definitiva del editor para exportar el videojuego

En este caso se ha habilitado únicamente el botón de exportación en Android. Se ha optado por dejar la posibilidad de exportación a otras plataformas para posteriores versiones del proyecto completo Gade4All que incluyan esta posibilidad.

6.6.2 Interfaz del videojuego

Como se comentó en el apartado de análisis, las pantallas son un aspecto personalizable por el usuario por lo que la estructura de las mismas no se atiene a un diseño concreto. Se incluye por tanto un diseño concreto generado a partir de una instancia básica del DSL con sus valores por defecto.

Ilustración 65. Pantalla principal del videojuego por defecto

Ilustración 66. Pantalla de opciones del videojuego por defecto

Ilustración 67. Pantalla de selección de nivel del videojuego por defecto

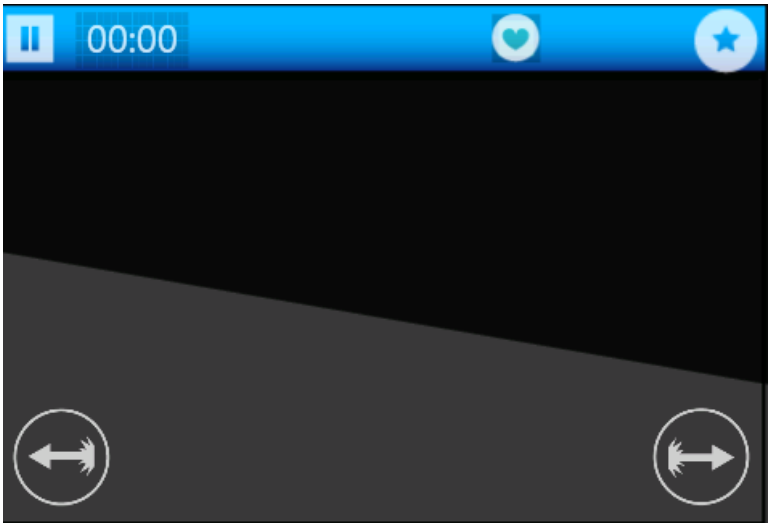


Ilustración 68. Pantalla de interacción del videojuego por defecto

Ilustración 69. Pantalla de pausa del videojuego por defecto

Ilustración 70. Pantalla de nivel finalizado con éxito del videojuego por defecto

Ilustración 71. Pantalla de "Game over" del videojuego por defecto

Ilustración 72. Pantalla de juego finalizado del videojuego por defecto

6.7 Especificación Técnica del Plan de Pruebas

6.7.1 Pruebas Unitarias

En el caso del editor, los casos de prueba se llevarán a cabo sobre la funcionalidad de cada pantalla, indicando las clases implicadas en la prueba unitaria.

Las pruebas unitarias se llevarán a cabo durante toda la fase de desarrollo para asegurar que no se implementa nuevo código sobre código erróneo. Para ello se probaran tanto con datos correctos como incorrectos para comprobar que la aplicación está blindada para obtener un videojuego funcional aunque el usuario intente meter datos erróneos.

Las pruebas unitarias en la fase de diseño son similares a las de la fase de análisis aunque se han añadido y eliminado algunas debido al conocimiento más profundo del software a implementar característico de esta fase. Asimismo, se ha decidido incluir algunas de esas pruebas de forma conjunta en la sección de pruebas de integración debido a la gran variedad de clases que involucran.

6.7.1.1 Configuración propiedades Globales

Configuración propiedades globales Clase **PageGlobalsPlatformVehicles.xaml**	
Prueba	**Resultado Esperado**
Insertar un nombre inválido (con tildes, espacios, caracteres especiales, etc.) para el videojuego.	El sistema informa de un error cuando el usuario intenta avanzar a la siguiente pantalla de edición.

Insertar un nombre vacío para el videojuego.	El sistema informa de un error cuando el usuario intenta avanzar a la siguiente pantalla de edición.
Insertar un nombre válido para el videojuego.	El editor permite continuar con el proceso de edición y cuando ésta finalice se obtiene un videojuego con el nombre asignado previamente.
Escoger un icono para la aplicación que no es formato .png.	El editor no permite seleccionar imágenes con formato distinto a .png. El explorador de carpetas para escoger el icono tiene un filtro, así que solamente mostrará imágenes con esta extensión.
Escoger un icono .png para la aplicación.	El editor muestra una pre visualización de la imagen escogida y se asignará al videojuego.
Marcar la opción de publicidad y no se añade ningún ID.	El sistema visa de que se necesita un ID para mostrar la publicidad.
Marcar la opción de publicidad e incluye un ID.	Se asigna el valor al identificador de la publicidad y se muestra dicha publicidad en el videojuego final.
Pulsar en el botón “siguiente”	Se crean las carpetas necesarias en la ruta escogida por el usuario para almacenar los diferentes elementos globales que ya se han definido.
Elegir una melodía para los menús del juego con formato .mp4	La aplicación no permite seleccionar archivos de sonido con formato distinto a .mp3 o .wav. El explorador de carpetas para escoger el sonido tiene un filtro, así que solamente mostrará archivos con dicha extensión.
Seleccionar un archivo de sonido en formato .mp3	La aplicación permite escucharlo y lo asigna ante un determinado evento o como música ambiente.
Pulsar el botón de reproducir sin haber escogido previamente un sonido.	El sonido no se reproduce.
Pulsar el botón de reproducir habiendo escogido previamente un sonido.	El sonido se reproduce sin problemas.
Pulsar sobre el botón de pausa una vez el sonido esté reproduciéndose.	El sonido se para.

No seleccionar ningún sonido para un determinado evento.	El editor permite no asignar sonidos a eventos.
Definir todos los sonidos necesarios y se continúa la edición.	Todos los sonidos seleccionados se incluirán en una carpeta sounds. La correspondencia entre sonidos y eventos se verá reflejada en el xml del nivel.

6.7.1.2 Configuración pantallas

Configuración pantallas Clases **PageMainScreen.xaml,PageOptionsScreenPlatformVehicles.xaml, PageSelectLevelScreenPlatformVehicles.xaml, PageGameScreenPlatformVehicles.xaml, PagePauseScreen.xaml, PageEndLevelScreenPlatformVehicles.xaml, PageLoseLevelScreenPlatformVehicles.xaml, PageEndGameScreen.xaml.**	
Prueba	**Resultado Esperado**
No definir la pantalla principal.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
No definir la pantalla de opciones.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
No definir la pantalla de victoria.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
No definir la pantalla de derrota.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
No definir la pantalla de fin.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
Disponer los elementos de la pantalla de juego en el medio de la misma.	El editor no dará ningún error aunque es lógico que de esta manera no se visualice correctamente el juego. Este aspecto de disposición de los elementos se deja a elección del diseñador.
Seleccionar un botón.	El botón se selecciona dentro de un recuadro
Escoger un fondo no .png	El editor no permite seleccionar imágenes con formato distinto a .png. El explorador de carpetas para escoger el icono tiene un filtro, así que solamente mostrará imágenes con esta extensión.

Cambiar la posición de un botón.	El botón se mueve y aparece en dicha posición en el juego final.
Ocultar un botón	El botón baja de opacidad y no aparece en el juego final

6.7.1.3 Configuración personaje principal

Configuración personaje principal Clase **CharacterSetPropertiesPlatformVehicles**	
Prueba	**Resultado Esperado**
Definir la misma imagen para todos los eventos.	El editor asignará la misma imagen para todos los eventos sin errores.
Seleccionar un personaje ya creado de la biblioteca.	El editor cargará dicho personaje y mostrará una visualización de las animaciones que lo componen.
Definir una animación para algún evento del personaje.	Se mostrará una vista previa en el editor que recoja los parámetros introducidos para definir la animación (imágenes, número de frames, si debe reproducirse en bucle, etc.)
Pulsar el botón "Añadir a biblioteca" cuando no hay ningún personaje creado.	El personaje incompleto no se agrega a la biblioteca.
Pulsar el botón "Añadir a biblioteca" cuando se ha finalizado la creación de un personaje.	El personaje se agrega correctamente a la biblioteca para poder ser usado posteriormente en otros desarrollos.
Pulsar el botón "Borrar" seleccionando un personaje de la biblioteca.	El personaje se borra de la biblioteca si no está vinculado a ningún escenario.
Seleccionar un personaje de la biblioteca	Se cargan todas las variables propias del personaje y se visualiza una muestra previa de sus animaciones.
Seleccionar un personaje de la biblioteca, cambiar alguna de sus propiedades y hacer clic en el botón de "Actualizar".	Si las propiedades son válidas, se actualiza dicha información en la base de datos de la biblioteca.

Insertar un valor negativo para aquellos factores que no pueden serlo (gravedad, factor de fricción, etc.)	El valor negativo no se asigna. Se preparará el editor para que no acepte números negativos para determinados elementos.
No modificar ninguno de los valores que afecten a la física del juego.	La física del videojuego será la que está definida por defecto en el editor.
Insertar un valor de gravedad igual a cero.	El editor no permitirá asignar valores iguales o menores que cero para la gravedad.
Insertar un valor de factor de aceleración igual a cero.	El editor permitirá asignar esos valores pero mostrará un mensaje indicando que los factores de aceleración que sean igual a cero impedirán que el personaje se mueva en una u otra dirección.
Insertar un valor con letras en uno de los campos.	El editor emitirá un mensaje de que el número introducido es incorrecto y no asignará dicho valor.
Insertar valores válidos en todos los textbox.	El editor asignará dichos valores a la instancia del DSL.
Intentar asignarle al personaje un tamaño menor que 35.	El editor sólo permitirá añadir valores para el tamaño del personaje principal comprendidos entre 35 y 70.
Intentar asignarle al personaje un tamaño mayor que 70.	El editor sólo permitirá añadir valores para el tamaño del personaje principal comprendidos entre 35 y 70.

6.7.1.4 Configuración elementos del nivel

Configuración elementos del nivel Clases **TrapSetPropertiesPlatformVehicles.xaml, CheckpointSetPropertiesPlatformVehicles.xaml, TileSetPropertiesPlatformVehicles.xaml, TargetSetPropertiesPlatformVehicles.**	
Prueba	**Resultado Esperado**
Elegir un recurso gráfico para un elemento que no sea .png	El editor no permite seleccionar imágenes con formato distinto a .png. El explorador de carpetas para escoger el icono tiene un filtro, así que solamente mostrará imágenes con esta extensión.

Definir la imagen, el número de frames y el tamaño de un elemento de nivel.	Se muestra la vista previa del elemento
Añadir un número de frames negativo	El editor muestra el mensaje de error correspondiente y no permite guardar el sprite.
Seleccionar un elemento de nivel de la biblioteca	Los parámetros del elemento ya creado se cargan y se muestran en el editor con su vista previa incluida.
Agregar un elemento nuevo a la biblioteca	Si todos sus parámetros son correctos se agrega ese nuevo punto de control a la biblioteca.

6.7.1.5 Configuración escenario

Configuración escenario Clase **SceneryPlatformVehicles**	
Prueba	**Resultado Esperado**
Definir el punto de meta en una posición anterior al punto de salida.	Se asignan ambos valores a la instancia del DSL. Una vez se defina el videojuego, el personaje principal podrá ir de derecha a izquierda para completar el nivel.
Se sitúa un Tile en la posición X,Y.	Si la posición es válida dentro de los parámetros del nivel, se situará el Tile correspondiente en el escenario.
Definir un escenario compuesto por Tiles de tal manera que es imposible que el personaje principal llegue a la meta.	El editor asignará igualmente los recursos a la instancia del DSL. Es competencia del usuario diseñador definir niveles que puedan ser finalizados.
Se pulsa el botón "Siguiente" sin haber definido la meta y el personaje principal	Es obligatorio que exista un personaje principal y una meta para que el nivel se guarde. Se mostrará el mensaje de error correspondiente.
No se escoge ninguna imagen para el fondo.	El fondo por defecto es negro
Se pulsa el botón de "Siguiente"	Los parámetros elegidos se agregan a la instancia del DSL y los recursos gráficos se almacenan en la carpeta Drawable.

Situar un punto de control en un punto posterior a la meta.	El punto de control se agrega sin problemas. Es responsabilidad del usuario diseñador definir puntos de control que realmente se vayan a utilizar.
Situar una trampa en un punto posterior a la meta.	La trampa se agrega sin problemas. Es responsabilidad del usuario diseñador definir trampas que no sean activas en el desarrollo del videojuego.
Situar una trampa en una posición que impide al personaje llegar a la meta.	La trampa se agrega sin problemas. Es responsabilidad del usuario diseñador definir un nivel factible.
No seleccionar número de vidas.	El selector ya tendrá un valor por defecto igual a tres.
Seleccionar un número de vidas inferior a uno.	Cuando se intente actualizar el número de vidas la aplicación dará un error y no lo permitirá.
Desmarcar el checkbox de tiempo y no definir un tiempo máximo para finalizar el nivel.	Si no se define un valor de tiempo mayor que cero, aunque la casilla de “sin límite de tiempo” esté desmarcada, el nivel se ejecutará sin límite de tiempo.
Desmarcar el checkbox de tiempo, definir un tiempo válido y escoger un número de vidas entre uno y cinco.	Se asignarán dichos parámetros al .xml del nivel y, posteriormente, el usuario jugador tendrá que llegar a la meta del nivel en el tiempo fijado y sin gastar las vidas definidas.
Pulsar el botón "Limpiar escenario"	Tras el mensaje de confirmación correspondiente, el escenario del nivel se borra.

6.7.2 Pruebas de Integración y del Sistema

Las siguientes pruebas se realizarán cuando se haya finalizado la implementación del sistema completo, corrigiendo los fallos a medida que se encuentren y repitiendo la comprobación cuando se crea que se haya resuelto la deficiencia.

6.7.2.1 Cargar/guardar videojuego

Paso 1. Guardar videojuego	
Prueba	**Resultado Esperado**
Abrir el editor, realizar algunos cambios y navegar entre pantallas. Dejar alguna pantalla con errores.	En la carpeta donde se almacena el juego se crea un archivo .dat por cada pantalla que se configura y se pulsa sobre "Siguiente". Si ya existe se actualiza.
Prueba	**Resultado Esperado**
En alguna pantalla pulsar sobre la opción "Guardar" del menú.	Actualiza los archivos .dat existentes o creo los nuevos necesarios guardando los parámetros del videojuego en desarrollo.
Paso 2. Cargar videojuego previamente guardado	
Prueba	**Resultado Esperado**
Cerrar la aplicación, volver a abrirla y cargar el juego guardado en el paso 1.	El videojuego se cargará correctamente recuperando todos los elementos que se habían salvado en su momento.
Prueba	**Resultado Esperado**
Acceder a la pantalla guardada con errores.	El editor mostrará los errores que impiden la edición del videojuego.
Prueba	**Resultado Esperado**
Pulsar el botón "anterior" en algún momento de la edición	En la pantalla anterior aparecerán los valores que introdujo el usuario al acceder a dicha pantalla.

6.7.2.2 Exportar videojuego

Exportar videojuego	
Prueba	**Resultado Esperado**
Crear un videojuego y pulsar "siguiente" en la última pantalla.	Se muestra la pantalla de selección de ruta de despliegue.

Prueba	**Resultado Esperado**
Se elige una ruta donde almacenar el videojuego, se selecciona la plataforma Android y se pulsa el botón de generar.	Tras el tiempo necesario, el proyecto Android con todos los elementos definidos aparecerá en la ruta seleccionada.
Prueba	**Resultado Esperado**
Se importa el proyecto generado en el Eclipse.	El proyecto se importa sin dar errores.
Prueba	**Resultado Esperado**
Se compila el proyecto.	El proyecto se compila sin errores y se genera el .adk, archivo a instalar posteriormente en el dispositivo del usuario jugador.
Prueba	**Resultado Esperado**
Se instala el fichero .adk generado en Eclipse en el dispositivo.	El videojuego se instala correctamente en el dispositivo y muestra en primera instancia la pantalla principal.

6.7.2.3 Jugar videojuego

Las pruebas del subsistema videojuego se realizarán sobre los siguientes dispositivos para garantizar el correcto funcionamiento en diversos dispositivos y en diferentes resoluciones:

Dispositivo	**Resolución**	**RAM**	**Sistema operativo**
Samsung Galaxy Mini	320x240	384 MB	Android 2.2 (Froyo).
Samsung Galaxy S II	800x480	1 GB	Android 4.1.2 (Jelly Bean).
Huawei U8650	480x320	256 MB	Android 2.3 (Gingerbread).
Emulador 1	320x240	512 MB	Android 4.2.2 (Jelly Bean).
Emulador 2	480x320	512 MB	Android 4.2.2 (Jelly Bean).
Emulador 3	800x480	512 MB	Android 4.2.2 (Jelly Bean).

A diferencia del resto de pruebas de integración, las pruebas del videojuego se realizarán a medida que se vaya implementando la funcionalidad del mismo, durante las primeras fases de implementación del proyecto general que consistirá en el desarrollo de un prototipo de juego.

Caso de Uso 5: Jugar videojuego	
Prueba	**Resultado Esperado**
Se ejecuta el videojuego en el emulador del Eclipse	El videojuego se instala correctamente y funciona. No se asegura que la velocidad de refresco iguale a la del dispositivo.
Prueba	**Resultado Esperado**
Se ejecuta el videojuego en diferentes dispositivos Android con diferentes resoluciones	La lógica del videojuego se comporta exactamente igual en todas las resoluciones.
Prueba	**Resultado Esperado**
Se selecciona un nivel	Tras el tiempo de espera requerido, el personaje principal apareceré en el nivel seleccionado
Prueba	**Resultado Esperado**
Se procede a seleccionar un nivel sin límite de tiempo	En el selector de niveles no aparecerá ningún indicador del tiempo necesario para superar el nivel.
Prueba	**Resultado Esperado**
Una vez superado un nivel se vuelve a la pantalla de selección de nivel.	En la pantalla de selección de nivel pueden elegirse sólo aquellos niveles que se hayan superado.
Prueba	**Resultado Esperado**
Se llega a la meta del nivel cumpliendo el objetivo predefinido	Se muestra un mensaje de fin de nivel con el tiempo invertido en completarlo y eventualmente se pasa al siguiente nivel si lo hubiera.
Prueba	**Resultado Esperado**
El personaje principal pierde todas las vidas	Se muestra una imagen tipo “Game Over” y se reinicia el nivel.
Prueba	**Resultado Esperado**
Se finaliza el último nivel	Se muestra una imagen de juego finalizado y se vuelve al menú principal del juego.

Prueba	**Resultado Esperado**
El personaje principal finaliza un nivel	Aparece la pantalla de fin de nivel con el tiempo que le llevó al usuario completarlo.
Prueba	**Resultado Esperado**
El personaje choca contra un ladrillo sólido.	El personaje principal debe de frenar aunque el botón e avance esté pulsado.
Prueba	**Resultado Esperado**
El personaje principal colisiona con una rampa ascendente	El personaje principal debe ascender por la rampa con la inclinación correspondiente.
Prueba	**Resultado Esperado**
El personaje principal colisiona con una rampa descendente	El personaje principal debe descender por la rampa con la inclinación correspondiente.
Prueba	**Resultado Esperado**
El personaje principal colisiona contra una trampa	El personaje principal pierde una vida.
Prueba	**Resultado Esperado**
El personaje principal cae por la parte inferior de la pantalla.	El personaje principal pierde una vida.
Prueba	**Resultado Esperado**
El personaje principal muere por alguna de las razones anteriores habiendo alcanzado algún *checkpoint* del nivel previamente.	El personaje principal pierde una vida y aparece en el último punto de control por el que haya pasado si sus vidas no han llegado a cero.
Prueba	**Resultado Esperado**
El personaje principal no supera el tiempo mínimo definido para completar el nivel	El marcador de tiempo debe de parpadear cuando el tiempo esté cerca de agotarse y una vez que lo haga el nivel se reinicia.

Prueba	**Resultado Esperado**
Al personaje principal sólo le queda una vida y la pierde.	El nivel se reinicia.
Prueba	**Resultado Esperado**
Se pausa el juego	Aparece la pantalla de pausa y la lógica del juego deja de actualizarse.
Prueba	**Resultado Esperado**
Se cambian los parámetros por defecto de los sonidos en la pantalla de opciones desactivando la música y los efectos.	Si los efectos y la música ambiental están desactivados, no se escuchará ningún tipo de sonido al comenzar cualquiera de los niveles.
Prueba	**Resultado Esperado**
Desde la pantalla de victoria probar que los botones de "Siguiente nivel" y "Repetir nivel" cumplen con la navegabilidad definida.	El botón de "Siguiente nivel" carga el siguiente nivel si no es el último. El botón de "Repetir nivel" carga el mismo nivel que se acaba de jugar.
Prueba	**Resultado Esperado**
Desde la pantalla de derrota probar que los botones de "Repetir nivel" y "Seleccionar nivel" cumplen con la navegabilidad definida.	El botón de "Repetir nivel" carga el mismo nivel que se acaba de jugar. El botón de "Seleccionar nivel" lleva a la pantalla de selección de niveles.

6.7.3 Pruebas de Usabilidad y Accesibilidad

Dado que uno de los principales objetivos del proyecto reside en implementar una plataforma que permita diseñar videojuegos de vehículos a personas sin conocimientos técnicos sobre programación de videojuegos, el aspecto de usabilidad de la aplicación es clave para que los usuarios se encuentren con una interfaz atractiva y fácil de utilizar para lograr su objetivo.

Por ello se han elegido distintos tipos de usuario que se adaptan a los eventuales perfiles de las personas que utilizarán la aplicación cuando ésta se encuentre en producción:

- **Tipo de usuario 1:** Estudiante de informática o usuario con estudios de ingeniero informático, con amplios conocimientos de programación y muy familiarizado con el manejo de ordenadores.

- **Tipo de usuario 2:** Usuario con estudios no relacionados con la informática y familiarizado con el manejo básico de ordenadores.

Las pruebas se han llevado a cabo en el despacho de la desarrolladora. Durante las mismas cada usuario contaba con el "Manual de Usuario" para consultar las dudas que surjan durante el proceso.

6.7.3.1.1 Cuestionario de evaluación

Para evaluar la usabilidad de la aplicación se ha llevado a cabo un cuestionario detallado a cada uno de los usuarios elegidos para las pruebas. Dicho cuestionario abarca los siguientes puntos:

- **1º**: **Preguntas de carácter general** acerca del propósito de la herramienta, los videojuegos y el mundo móvil.
- **2º**: **Actividades guiadas** sobre la aplicación.
- **3º**: **Batería de preguntas cortas** sobre algunos aspectos de la plataforma.
- **4º**: **Observaciones** para que el usuario aporte todo lo que considere oportuno de la aplicación.

6.7.3.1.2 Cuestionario para el responsable de las pruebas

También se ha redactado un cuestionario para que el responsable de las pruebas apunte alguna información útil a medida que el usuario realiza la prueba.

6.7.3.2 Actividades de las Pruebas de Usabilidad

6.7.3.2.1 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador? Puede marcar más de una.
1. Es parte de mi trabajo o profesión 2. Lo uso para ver películas o series y jugar a videojuegos.

3. Empleo aplicaciones estilo Office para el ámbito del estudio y trabajo.
4. Lo utilizo para comunicarme a través de redes sociales.
5. Únicamente leo el correo y navego ocasionalmente

¿Es un usuario habitual de videojuegos?

1. Sí, juego a menudo
2. No, pero juego de vez en cuando
3. No, nunca o casi nunca juego

¿En qué plataformas acostumbra a jugar? Puede marcar más de una.

1. Consolas
2. Ordenador
3. Móviles
4. Otro: …….

¿Qué tipo de actividades realiza con su teléfono móvil? Puede marcar más de una.

1. Para asuntos del trabajo
2. Sólo para llamar y/o mandar mensajes
3. Para conectarme a Internet y consultar información
4. Para mantenerme en contacto con mis amigos mediante redes sociales o aplicaciones de mensajes
5. Para jugar a juegos
6. Otro: …….

¿Qué sistema operativo tiene en su teléfono móvil?

1. Android
2. iOS
3. Windows Phone
4. BlackBerry OS
5. Otro: …….

¿Está familiarizado con el desarrollo de videojuegos?

1. Sí

2. No, pero es un campo interesante 3. No, no me interesa
¿Conoce la existencia de algún producto software similar al de esta prueba?
1. Sí. Indique cual: ……. 2. No
¿Qué grado de dificultad otorgaría al desarrollo de videojuegos?
1. Fácil 2. Medio 3. Difícil 4. Muy difícil
¿Qué grado de dificultad otorgaría al desarrollo de videojuegos utilizando la herramienta?
1. Fácil 2. Medio 3. Difícil 4. Muy difícil

6.7.3.2.2 Actividades guiadas

La actividad guiada consistirá en crear un videojuego desde cero con los parámetros que el usuario escoja libremente. Para definir los elementos de los niveles, las pruebas se acompañarán con una serie de carpetas con los recursos gráficos necesarios. Las condiciones de la prueba serán:

- Cambiar algunos parámetros por defecto que ofrece el editor.
- Definir un personaje principal, una trampa, un punto de control, una meta y *Tiles* de diferentes inclinaciones.
- Guardar y recuperar el videojuego guardado.
- Definir un nivel sencillo.
- Exportar el videojuego e instalarlo en el dispositivo.
- Jugar el primer nivel.

6.7.3.2.3 Preguntas cortas sobre la aplicación y observaciones

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe en qué fase del desarrollo se encuentra?				
¿Los mensajes de error le han servido para solucionar la situación conflictiva?				
¿Ha podido moverse con facilidad por las diferentes fases?				
¿La información de la pantalla es suficiente para entender qué se está haciendo?				
¿Cuán a menudo han surgido dudas acerca de cómo seguir avanzando en la aplicación?				
¿Le ha resultado intuitiva la manera de definir un personaje?				
¿Le ha resultado intuitiva la manera de definir un elemento del nivel?				
¿Le ha resultado intuitiva la manera de definir un nivel completo?				
¿Ha considerado el proceso tedioso y largo?				
¿Le ha resultado fácil jugar al videojuego?				
¿El manual de usuario ha resuelto sus dudas en el caso de haberlo utilizado?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea de manera correcta?				
¿El tiempo de respuesta de la aplicación es muy grande?				
¿Encontró algún error insalvable en la aplicación?				
¿Encontró algún componente se comportó de manera diferente a la esperada?				
¿Tuvo dudas acerca de la repercusión de una elección en el editor sobre el juego final?				
¿El videojuego funciona sin fallos?				
¿El videojuego funciona de manera poco realista?				
¿Se corresponden los elementos definidos en el editor con el videojuego resultante?				

¿El proceso para guardar un videojuego fue rápido?				
¿El proceso de carga de un videojuego fue rápido?				
¿El proceso de exportación fue rápido?				
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
¿El tipo de letra es adecuado?				
¿El tamaño de letra es el adecuado?				
¿El contraste entre fondo y letra es el adecuado?				
¿La disposición de los elementos en la pantalla en adecuada?				
¿Las imágenes e iconos utilizados son adecuados?				
¿Cree que la resolución de los elementos gráficos del juego es la adecuada?				
Diseño de la Interfaz		**Si**	**No**	**A veces**
A nivel general, ¿le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿El diseño de las pantallas y sus elementos internos es estructurado?				
¿Cree que la elección de orden de las pantallas es correcta?				
¿Ha tardado en encontrar algún elemento en una pantalla?				
¿Considera que hay alguna pantalla sobrecargada de información?				
¿Los botones tienen el tamaño suficiente y están bien ubicados?				
¿Le ha resultado satisfactorio el diseño del videojuego final?				
Observaciones				
Introduzca aquí cualquier comentario que le parezca pertinente.				

6.7.3.2.4 Cuestionario para el responsable de las pruebas

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	
Tiempo en realizar cada tarea	
Errores leves cometidos	
Errores graves cometidos	
Errores de la aplicación	
El usuario consulta el manual del usuario en determinado punto	
El usuario se pierde en determinado punto y no puede continuar por sí mismo	
El usuario carga correctamente un recurso visual	
Dificultad en leer algún texto	
El usuario encuentra tedioso el proceso involucrado en alguna de las pantallas	
El usuario desconoce para qué sirven los parámetros que está introduciendo	
Satisfacción del usuario mientras usa la herramienta	
Satisfacción del usuario con el videojuego creado	
El usuario sabe cómo jugar al videojuego final	

6.7.4 Pruebas de Rendimiento

Las pruebas de rendimiento se limitarán a comprobar que el refresco de la lógica del videojuego es el adecuado en cada uno de los dispositivos Android definidos en las pruebas de integración.

Los videojuegos deben ejecutar su lógica de negocio treinta veces por segundo (30 fps). En el momento en el que el número de ejecuciones baja se podría considerar que existe un problema de rendimiento.

Capítulo 7. Implementación del Sistema

7.1 Lenguajes de Programación

7.1.1 Java

Java es un lenguaje de programación de propósito general basado en clases y orientado a objetos que fue diseñado para tener tan pocas dependencias de implementación como fuera posible. Su sintaxis deriva de C y C++ con la diferencia de que no cuenta con muchos elementos a bajo nivel. Las aplicaciones de Java se compilan a bytecode (clase Java) y pueden ser ejecutadas en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la máquina, lo que facilita en gran medida su portabilidad.

En la actualidad es uno de los lenguajes de programación más populares, en concreto en aplicaciones cliente-servidor para web, con unos 10 millones de usuarios registrados.

Se ha elegido Java para programar la lógica del videojuego por tratarse del lenguaje más popular en el desarrollo de aplicaciones Android.

7.1.2 C#

C# o C Sharp es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft dentro de su plataforma .NET. Fue diseñado como un lenguaje de programación para lograr una infraestructura de lenguaje común.

Su sintaxis deriva de C y C++ aunque utiliza el modelo de objetos de la plataforma .NET de manera similar a como lo hace Java.

Se ha optado por usar C# a la hora de implementar la funcionalidad relacionada con el editor gráfico porque éste ya implementaba el resto de tipologías utilizando este lenguaje. Se utilizó C# en primera instancia por su similitud con Java y porque la plataforma .NET ofrece una manera fácil y directa de implementación de interfaces gráficas a través de WPF mediante el lenguaje XAML.

7.1.3 XML

XML se corresponde con las siglas en inglés para “lenguaje de marcas extensible”. Es por tanto un lenguaje de marcado desarrollado por el W3C (World Wide Web Consortium) que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

Es una tecnología sencilla que se ve complementada con otras que la enriquecen y la dotan de muchas posibilidades. Al tratarse de un estándar en el intercambio de información sus usos van desde bases de datos, editores de texto hasta servicios web u hojas de cálculo. En el presente proyecto se utilizará para definir las instancias del DSL. En definitiva, para transmitir la información entre el editor gráfico y la plantilla del videojuego.

7.1.4 XAML

XAML se corresponde con las siglas en inglés para "lenguaje extensible de formato para aplicaciones". Es por tanto un lenguaje de formato para la interfaz de usuario en la capa de presentación diseñado para soportar clases y métodos de la plataforma de desarrollo .NET relacionadas con la interacción del usuario.

Se trata de un lenguaje declarativo basado en XML pero optimizado para describir interfaces de usuarios visuales ricas desde el punto de vista gráfico. Forma parte de la WPF (*Microsoft Windows Presentation Foundation),* categoría de características de Microsoft .NET Framework 3.5 relacionadas con la presentación visual de aplicaciones basadas en Windows y de aplicaciones cliente orientadas a exploradores web.

Se ha utilizado en este proyecto para definir las interfaces del editor gráfico ya que viene embebido en el *framework* .NET.

7.2 Herramientas y Programas Usados para el Desarrollo

7.2.1 Eclipse y Android SDK

Eclipse es uno de los más famosos entornos de desarrollo software. Ofrece soporte en múltiples idiomas y está escrito de forma mayoritaria en Java. Se usa para desarrollar aplicaciones en Java aunque con la instalación de diversos *plug-ins* también se puede programar en una amplia variedad de lenguajes: C, C++, PHP, Python, Ruby, etc.

Su primera versión vio la luz en Junio de 2004 y la próxima versión de Eclipse (Kepler 4.3) se espera para Junio de 2013. Desde sus inicios Eclipse siempre ha sido software gratis y de código abierto.

Para desarrollar aplicaciones Android dentro del entorno Eclipse es necesario utiliza el *Android software development kit* (SDK) que utiliza el *plug-in* de *Android Development Tools* (ADT) para integrarse con el IDE. Este completo *plug-in* incluye una serie de herramientas de desarrollo como librerías software, depurador, emulador, documentación, códigos de ejemplo y tutoriales.

Para desarrollar este proyecto se ha descargado de la web de desarrolladores de Android (http://developer.android.com/sdk/index.html) un paquete que ya incluye la última versión de Android SDK dentro del Eclipse sin necesidad de instalaciones o configuraciones adicionales.

7.2.2 Visual Studio 2012

Microsoft Visual Studio es un IDE (entorno de desarrollo integrado) para sistemas operativos Windows. Soporta varios lenguajes de programación: Visual C++, Visual C#, Visual J# y Visual Basic .NET aunque ha desarrollado extensiones necesarias para aceptar muchos otros.

Visual Studio permite la creación de aplicaciones de escritorio, aplicaciones web, sitios y servicios web en cualquier entorno que soporte la plataforma .NET.

Se ha elegido este entorno para desarrollar el editor gráfico debido a que Visual Studio es el IDE estándar para el *framework* .NET y éste nos aporta una serie de ventajas que se han comentado en el apartado anterior de "Lenguajes de programación".

7.3 Creación del Sistema

En esta sección se incluyen todos aquellos aspectos con los que nos hemos encontrado a lo largo de la implementación.

7.3.1 Problemas Encontrados

7.3.1.1 Desconocimiento de la programación de videojuegos en Android

Aunque la alumna contaba con conocimientos de programación en Android y con nociones básicas de software dirigido por modelos, las primeras semanas de proyecto se centraron en adquirir las competencias básicas de programación de videojuegos en Android a través de tutoriales y ejemplos suministrados por los tutores. Por ello, en el primer mes de proyecto se avanzó a ritmo muy lento y con constantes dudas que dificultaron cumplir la planificación propuesta inicialmente.

7.3.1.2 Elección de metodología de implementación de videojuego

El desconocimiento que se comentaba en el apartado anterior ha influido en las primeras decisiones a tomar a la hora de implementar el videojuego. El tutor sugirió dos metodologías iniciales aunque recomendó encarecidamente la segunda:

- Utilizar un framework tipo AndEngine o jBox 2D que facilite en gran medida la implementación de la física del videojuego.
- Obviar cualquier framework e implementar la física del juego desde cero.

La alumna comenzó con la segunda opción pero en las primeras fases de implementación decidió indagar en el funcionamiento del framework jBox2D para saber si podría ser de ayuda para los primeros problemas del software que estaban apareciendo.

Finalmente se ha seguido el enfoque de todos los videojuegos de Gade4All de implementar la lógica desde cero ya que resulta más fácil que depender de una librería de terceros. Además, las físicas involucradas no iban a ser extremadamente complejas.

7.3.1.3 Estudio de código implementado por terceros

Para crear la nueva rama del editor era necesario realizar un estudio previo del editor Gade4All actual pues había clases que podrían reaprovecharse y el nuevo código a implementar debía estructurarse de acuerdo a la arquitectura ya existente. Dicho código no estaba bien documentado y los comentarios no ofrecían toda la información necesaria. Además, el hecho de no estar en el equipo de trabajo del proyecto de forma presencial fomentó que la fase de comprensión de funcionamiento del editor se alargara más de lo debido.

7.3.1.4 Implementación de la lógica para subir y bajar rampas de diferentes pendientes

La idea principal de implementar un videojuego orientado al control de vehículos radicaba en que el personaje principal pudiera subir y bajar diferentes tipos de elevaciones. A pesar de ser una idea bastante simple este aspecto se tardó en resolver más de tres semanas pues en la bibliografía se encontraron diferentes enfoques pero ninguno de ellos ofrecía resultados convincentes. Tras mucha investigación y pruebas de ensayo y error se utilizó la metodología que proponen en un blog llamado "Killerko's blog"[5] ligeramente modificada para poder aceptar cuestas de diferentes pendientes.

7.3.2 Descripción Detallada de las Clases

En primer lugar se van a detallar las clases más importantes del **subsistema editor** (código C#)

7.3.2.1 AlmacenPlatformVehicles

Nombre	**Tipo**	**Descripción**	**Hereda de…**
AlmacenPlatformVehicles	Serializable	Almacena los objetos serializables de todos los elementos de los niveles.	
Responsabilidades			
Número	**Descripción**		
1	Sirve como contenedor para almacenar los objetos serializables a través del proceso de edición. Las diferentes pantallas inyectan los datos en las listas que alberga para luego recuperarlos.		

[5] La URL concreta que sirvió de ayuda fue: http://killerkouk.blogspot.com.es/2011/03/how-to-implement-slopes.html

Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	Void	add_Objects	string newkey, AlmacenObjetos newValue
Público	AlmacenObjetos	get_Object	string newkey

Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Público		Dictionary<String, AlmacenObjetos>	dictionaryObjects
Público		List<AlmacenObjetos>	otherBackgrounds
Público		AlmacenPropiedades	gameProperties
Público		List<CharacterSerializationPlatformVehicles>	characterSerialization
Público		List<TrapSerializationPlatformVehicles>	trapSerialization
Público		List<CheckpointSerializationPlatformVehicles>	checkpointSerialization
Público		List<TargetSerializationPlatform	targetSerialization
Público		List<TileSerializationPlatform>	tileSerialization
Público		GraphicRepresentationSerializationPlatformVehicles	grSerialization
Observaciones			

7.3.2.2 AlmacenPropiedades

Nombre	**Tipo**	**Descripción**	**Hereda de…**
AlmacenPropiedades	Serializable	Almacena las propiedades simples globales del videojuego.	

Responsabilidades	
Número	**Descripción**
1	Almacena las propiedades simples globales del videojuego definidas en la primera pantalla. Común a todas las tipologías.

Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**

Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Público		string	nameGame
Público		string	iconImageSource
Público		string	namePackage
Público		string	comboBoxLanguage
Público		string	checkBoxAdvertising
Público		string	admobID
Público		string	adSenseID
Público		string	wphoneID

Público		string	html5SlotID
Público		string	html5ClientID
Público		string	screenOrientation
Público		string	comboBoxFps
Público		string	nameSound
Observaciones			

7.3.2.3 BaseWindow

Nombre	**Tipo**	**Descripción**	**Hereda de…**
BaseWindow	Parcial	Contiene las pantallas de edición de cada tipología de juego	Window
Responsabilidades			
Número	**Descripción**		
1	Dependiendo del parámetro que le llegue de *MainWindow* (tipología de juego elegida) carga unas pantallas u otras.		
2	Permite la navegación entre ellas haciendo de contenedor.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	void	addShortcutButtonToToolBar	String name, String page, int currentIndex
Público	void	recolocar	
Público	void	adaptButtonsColors	int currentIndex
Público	void	goToPreviousPage	
Público	void	goToNextPage	
Privado	void	openPage	String name, int currentIndex
Privado	void	MenuItemAbrir_Click	object sender, RoutedEventArgs e
Privado	void	MenuItemGuardar_Click	object sender, RoutedEventArgs e
Privado	void	MenuItemAcercaDe_Click	object sender, RoutedEventArgs e
Privado	void	MenuItemSalir_Click	object sender, RoutedEventArgs e
Privado	void	Window_Closing	object sender, System.ComponentModel. CancelEventArgs e
Privado	void	image_next_MouseDown	object sender, MouseButtonEventArgs e
Privado	void	image_back_MouseDown	object sender, MouseButtonEventArgs e
Privado	void	left_Click	object sender, RoutedEventArgs e

Privado	void	right_Click	object sender, RoutedEventArgs e
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
		bool	openNewFile
		string[]	window_names
		string[]	window_workflow
		string[]	window_names_touch_with_character
		string[]	window_workflow_touch_with_character
		string[]	window_names_platform
		string[]	window_workflow_platform
		string[]	window_names_puzzle_puzzlebubble
		string[]	window_workflow_puzzle_puzzlebubble
		string[]	window_names_trivial
		string[]	window_workflow_trivial
		string[]	window_names_strategy
		string[]	window_workflow_strategy
		string[]	window_names_platform_vehicles
		string[]	window_workflow_platform_vehicles
		Dictionary<String, String>	shortcutButtonNames
		int	index
		int	indexSuper
		int	indexMaximum
		List<ControlShortCut>	tabs
Observaciones			

7.3.2.4 CharacterSetPropertiesPlatformVehicles

Nombre	**Tipo**	**Descripción**	**Hereda de…**
CharacterSetPropertiesPlatformVehicles	Parcial	Gestiona la funcionalidad de la pantalla de creación de personajes de principales del juego	Page, IPageStep
Responsabilidades			
Número	**Descripción**		
1	Recibe y comprueba las propiedades de tamaño, colisiones, factores de física y sonidos del personaje principal.		
2	Gestiona las diferentes animaciones del personaje principal.		
3	Almacena toda la información anterior en la instancia del DSL.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**

Público	void	GenerateSpriteProperties	string ObjectName
Privado	void	LoadValuesIntoProperties	ControlSpriteProperty cspr
Privado	TabItem	GetItemToTabs	string name, ControlSpriteProperty cspr
Privado	void	LoadSpriteProperties	GameObject currentObject
Privado	void	InitializeControlProperties	
Privado	void	InitializeNpcProperties	
Privado	void	InitializeToolTips	
Público	void	RestoreControlProperties	
Público	bool	INextPage	
Público	bool	IPreviousPage	
Privado	void	AskForModifying	
Público	void	IPageLoaded	
Privado	void	Page_Loaded	object sender, RoutedEventArgs e
Privado	void	saveCurrentNPC	
Privado	void	AddGameObject	
Público	void	ISaveState	
Público	void	ILoadState	
Privado	void	Page_Unloaded	object sender, RoutedEventArgs e
Privado	void	listNpcs_SelectionChanged	object sender, SelectionChangedEventArgs e
Privado	void	LoadProperties	GameObject currentNPC
Público	int	getIndex	String key
Privado	void	butAddObjectToLibrary_Click	object sender, RoutedEventArgs e
Privado	void	AddGameObjectToLibrary	

Privado	void	butRefreshObject_Click	object sender, RoutedEventArgs e
Privado	void	butCreateNewObject_Click	object sender, RoutedEventArgs e
Privado	void	butDeleteObject_Click	object sender, RoutedEventArgs e
Privado	bool	IsOnScenery	
Privado	int	GetNumOfCharactersInCurrentLevel	(Dictionary<string, double[]> currentLevel
Privado	boolean	CheckVariables	

Atributos

Acceso	Modo	Tipo o Clase	Nombre
	Estático	bool	spritesDeUnFrame
	Estático	String	DSLKey
	Estático	String	DSLkey
		String	npcAutoID
		List<ControlSpriteProperty>	controlSpriteProperties
		Dictionary<String, List<ControlSpriteProperty>>	storeControlSpriteProperties
		List<GameProperty>	properties
		List<IControlProperty>	controlsProperties
		List<GameObject>	objects
		bool	autoSaveObject
		string[]	spriteNames
Privado		string	DSLInfo
		List<string>	tooltips

Observaciones

Las clases *CheckpointSetPropertiesPlatformVehicles.xaml, TrapSetPropertiesPlatformVehicles.xaml* son análogas a la presente pues también se basan en definir un conjunto de animaciones y propiedades para parametrizar un elemento del nivel. Para evitarse la repetición de tablas se han obviado en esta sección.

7.3.2.5 GamePropertiesPlatformVehicles

Nombre	Tipo	Descripción	Hereda de…
GamePropertiesPlatformVehicles	Parcial	Gestiona las propiedades de cada nivel propias de la tipología.	Window

Responsabilidades

Número	Descripción
1	Obtener las propiedades de FPS, vidas del nivel, tiempo para completarlo, sonido de nivel y sonido de finalizar nivel.
2	Comprobar que los parámetros anteriores son correctos.
3	Si son correctos, pasar los parámetros anteriores a la instancia del DSL en formato XML.

Métodos

Acceso | Modo	Tipo de Retorno	Nombre	Parámetros y tipos
Privado	void	butCancel_Click	object sender, RoutedEventArgs e
Privado	void	Button_Click	object sender, RoutedEventArgs e
Público, override	string	ToString	
Público, estático	string	EmptyStageGamePropertiesToString	string tboard
Privado	void	buttonAddMusic_Click	object sender, RoutedEventArgs e
Privado	void	textBoxTime_KeyUp	object sender, KeyEventArgs e
Privado	void	checkBoxTime_Check	object sender, RoutedEventArgs e
Privado	void	textBoxTime_LostFocus	object sender, RoutedEventArgs e
Público, estático	string	GetFormattedTime	long seconds
Privado	void	buttonDeleteMusic_MouseDown	object sender, MouseButtonEventArgs e

Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		string	sound
Privado		string	exitsound
Privado		string	fps
Privado		string	time
Privado		string	lives
Privado		StagePlatformVehicles	stage

Observaciones

7.3.2.6 LevelPlatformVehicles

Nombre	Tipo	Descripción	Hereda de…
LevelPlatformVehicles	Serializable	Funciona como contenedor de todas las propiedades de un nviel	

Responsabilidades	
Número	**Descripción**
1	Almacena todas las propiedades de un nivel incluyendo los elementos del mismo.
2	Serializa y deserializa los elementos del nivel.

Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Privado	void	GeneratePathsToImages	
Privado	void	GenerateLevelSerializated	
Público override	string	ToString	

Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
	Interno	string	stageDSL
	Interno	string	stageBackgroundsDSL
	Interno	string	stageGamePropertiesPlatformDSL
	Interno	string	gp_FPS
	Interno	string	gp_Lives
	Interno	string	gp_Time
	Interno	string	gp_SoundSource
	Interno	string	gp_ExitSource
	Interno	string	bp_pathScrollingBackground
	Interno	string	bp_scrollSpeedX
	Interno	string	bp_scrollSpeedY
	Interno	string	bp_pathLayerBack
	Interno	string	bp_parBackSpeedX
	Interno	string	bp_parBackSpeedY
	Interno	string	bp_pathLayerMedium
	Interno	string	bp_parMediumSpeedX
	Interno	string	bp_parMediumSpeedY
	Interno	string	bp_pathLayerFront
	Interno	string	bp_parFrontSpeedX
	Interno	string	bp_parFrontSpeedY
	Interno	string	dp_DescriptionImageSource
	Interno	string	dp_DescriptionWidth
	Interno	string	dp_DescriptionHeight
	Interno	string	dp_ThumbnailImageSource
	Interno	string	dp_ThumbnailWidth;
	Interno	string	dp_ThumbnailHeight;
	Interno	string	dp_DescriptionImageSource
	Interno	string	ve_TileWidth
	Interno	string	ve_TileHeight
	Interno	string	cm_CanvasWidth
	Interno	string	cm_CanvasHeight
	Interno	string	cm_ScrollWidth
	Interno	string	cm_ScrollHeight
Privado		Dictionary<string, double[]>	gameObjects
	Interno	Dictionary<string, double[]>	GameObjects
Privado		Point	redRectanglePosition
Privado	Non-serialized	Image[]	imagesFromBackground
Privado		string[]	imagesFromBackgroundPath
Privado		BackgroundPropertiesPlatform.Selection	backgroundSelected
Privado		int	backgroundSelectedLayerSign

Privado		List<string>	listLevelDSLFile
Observaciones			

7.3.2.7 PageGlobalPlatformVehicles

Nombre	**Tipo**	**Descripción**	**Hereda de…**
PageGlobalPlatformVehicles	Parcial	Gestiona la pantalla de definición de las propiedades globales	Page, IPageStep

Responsabilidades	
Número	**Descripción**
1	Recibe las propiedades globales del juego: nombre del videojuego, icono, publicidad si la tuviera y música para los menús.
2	Guarda las propiedades en el *AlmacenPropiedades*
3	Crea el sistema de carpetas para almacenar el estado de edición.

Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	bool	INextPage	
Público	void	ISaveState	
Público	void	ILoadState	
Público	bool	IPreviousPage	
Público	void	IPageLoaded	
Privado	void	addIcon_Click	object sender, RoutedEventArgs e
Privado	void	checkBoxAdvertising_Click	object sender, RoutedEventArgs e
Privado	void	addSound_Click	object sender, RoutedEventArgs e
Privado	void	deleteSound_Click	object sender, RoutedEventArgs e
Privado	void	Page_Loaded	object sender, RoutedEventArgs e
Privado	void	nameGame_TextChanged	object sender, TextChangedEventArgs e
Privado	void	imageStop_MouseDown	object sender, MouseButtonEventArgs e
Privado	void	Focus	object sender, RoutedEventArgs e

Privado	void	TextChanged	object sender, RoutedEventArgs e
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Público	Estático	string	g_name
Público	Interno	string	g_iconImage
Público	Interno	string	g_namePackage
Público	Interno	string	g_language
Público	Interno	string	g_fps
Público	Interno	string	g_screenOrientation
Público	Interno	string	g_advertising
Público	Interno	string	g_google_play_key
Público	Interno	string	g_adsense_key
Público	Interno	string	g_soundSource
Público	Interno	string	g_windows_phone_key
Público	Interno	string	g_html5_slot_key
Público	Interno	string	g_html5_client_key
Público	Interno	string	g_name
Público	Interno	string	g_iconImage
Público	Interno	string	g_namePackage
Público	Interno	string	g_language
Público	Interno	string	g_fps
Público	Interno	string	g_screenOrientation
Público	Interno	string	g_advertising
Público	Interno	string	g_google_play_key
Público	Interno	string	g_adsense_key
Público	Interno	string	g_soundSource
Público	Interno	string	g_windows_phone_key
Público	Interno	string	g_html5_slot_key

Público	Interno	string	g_html5_client_key
Público		string	nameData
Público		bool	gameLoaded
Público		bool	isCreated
Público		string	routeImages
Observaciones			

7.3.2.8 SceneryPlatformVehicles

Nombre	**Tipo**	**Descripción**	**Hereda de…**
SceneryPlatformVehicles	Parcial	Gestiona la pantalla de definición de los niveles del videojuego.	Page, IPageStep
Responsabilidades			
Número	**Descripción**		
1	Gestiona la situación de elementos en el escenario para definir el nivel		
2	Abre las ventanas de propiedades del nivel y recibe dichos parámetros una vez definidos.		
3	Crea los xml por nivel y guarda toda la información para dejar el editor preparado para la exportación.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Privado	int	GetNumOfCharactersInCurrentLevel	Dictionary<string, double[]> currentLevel
Privado	int	GetNumOfTargetsInCurrentLevel	Dictionary<string, double[]> currentLevel
Privado	void	mainCanvas_MouseLeftButtonDown	object sender, MouseButtonEventArgs e
Privado	void	mainCanvas_MouseMove	object sender, MouseEventArgs e
Privado	bool[,]	BuildTileScenery	Canvas mainCanvas
Privado	Point	GetAdecuatedPositionFromGrid	Point previousPos
Privado		HideOrShowTargetWarning	
Privado		HideOrShowCharacterWarning	

Privado	bool	CheckNumCharactersAndTargetPerLevel	
Público	bool	INextPage	
Público	void	ISaveState	
Público	void	ILoadState	
Público	bool	IPreviousPage	
Público	void	IPageLoaded	
Privado	void	butBackgroundPropertiesPlatform_Click	object sender, RoutedEventArgs e
Privado	void	DoScrollBackground	BackgroundProperties Platform bp
Privado	void	DoParallaxBackground	BackgroundProperties Platform bp
Privado	void	ClearBackground	
Privado, estático	Rectangle	BuildBackground	DrawingGroup dg, string name
Privado	void	ResizeCanvas	ImageBrush ib
Privado	void	butBackgroundMeasures_Click	object sender, RoutedEventArgs e
Privado	void	butRestoreAll_Click	object sender, RoutedEventArgs e
Privado	void	RestoreAllScenery	bool needsUpdate
Privado	void	butClearCanvas_Click	object sender, RoutedEventArgs e
Privado	void	ClearCanvas	
Privado	void	butNewLevel_Click	object sender, RoutedEventArgs e
Privado	void	CreateNewLevel	
Privado	void	LoadDefaultValues	
Privado	void	butDeleteLevel_Click	object sender, RoutedEventArgs e
Privado	bool	AskForModifying	
Privado	void	listLevels_SelectionChanged	object sender, SelectionChangedEventArgs e
Privado	bool	CanvasHasElements	
Privado	void	ReloadLevel	object sender, SelectionChangedEven

			tArgs e
Privado	Image[]	GetImagesFromLevel	
Privado	Point	GetElementPositionRelativeToCanvas	int index
Privado	Point	GetActualRedRectanglePosition	
Privado	void	SetActualRedRectanglePosition	Point redRectanglePosition
Privado	void	butDescriptionImage_Click	object sender, RoutedEventArgs e
Privado	void	butGameProperties_Click	object sender, RoutedEventArgs e
Privado	void	ReadecuateBackgrounds	object sender, int layerSign
Privado	void	ScrollImageOnRight	DrawingGroup newGroupScroll, ImageSource imageSource, double imageWidth, double imageHeight, string familyBackground
Privado	void	ScrollImageOnLeft	DrawingGroup newGroupScroll, ImageSource imageSource, double imageWidth, double imageHeight, string familyBackground
Privado	void	butAddCanvasOnLeft_Click	object sender, RoutedEventArgs e
Privado	void	butAddCanvasOnRight_Click	object sender, RoutedEventArgs e
Privado	void	LoadScreenParamethers	string itemID
Privado	void	LoadObjectsOnScreen	string levelIDSelected, SelectionChangedEventArgs e
Privado	void	textBoxTileWidth_LostFocus	object sender, RoutedEventArgs e
Privado	void	textBoxTileHeight_LostFocus	object sender, RoutedEventArgs e
Privado	void	textBoxTileHeight_KeyUp	object sender, KeyEventArgs e

Privado	void	textBoxTileWidth_KeyUp	object sender, KeyEventArgs e
Privado	void	RunProcessValues	object sender, EventArgs e
Privado	void	butTryLevel_Click	object sender, RoutedEventArgs e
Privado	void	ReadecuateGrid	double tileWidth, double tileHeight
Privado	void	checkGridLines_Checked	object sender, RoutedEventArgs e
Privado	void	checkGridLines_Unchecked	object sender, RoutedEventArgs e

Atributos

Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		Dictionary<string, double[]>	coordsCharacter
Privado		List<LibraryItem>	listLibraryItems
Privado		List<string>	listLevelsFiles
Privado		int	indexLevel
Público	Estático	int	levelIndexSelected
		int	pruebaNumRandom
Privado		string	stageDSL
Privado		string	stageBackgroundsDSL
Privado		string	stageGamePropertiesPlatformDSL
Privado		Image	scroll
Privado		int	scrollXSpeed
Privado		int	scrollYSpeed
Privado		Image	parBack
Privado		int	parBackXSpeed
Privado		int	parBackYSpeed
Privado		Image	parMedium
Privado		Image	parFront
Privado		int	layerSpeed
		Rectangle	rectangle

Observaciones

7.3.2.9 SerializerPlatformVehicles

Nombre	**Tipo**	**Descripción**	**Hereda de…**
SerializerPlatformVehicles	Pública	Serializa y deserializa las propiedades del juego.	
Responsabilidades			
Número	**Descripción**		
1	Convierte a binario todos los datos de *AlmacenPlatformVehicles.*		
2	Transforma el archivo binario a los atributos de *AlmacenPlatformVehicles.*		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	void	SerializeObject	string filename, AlmacenPlatformVehicles objectToSerialize
Público	AlmacenPlatformVehicles	DeSerializeObject	string filename
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Observaciones			

7.3.2.10 StorePlatformVehicles

Nombre	**Tipo**	**Descripción**	**Hereda de…**
StorePlatformVehicles	Serializable	Almacena algunas propiedades del juego.	
Responsabilidades			
Número	**Descripción**		
1	Almacena las propiedades de FPS, tiempo, vidas y sonido de fondo.		
2	Escribe en el disco duro el fichero XML con todos los datos de la instancia del DSL cubiertos.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público, estático	StorePlatformVehicles	Instance	
Público	void	getXmlScreens	
Público	void	SaveGameProperties	string gpFPS, string gpTime,string gpSoundSource, string gpLives
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Público		string	gp_FPS
Público		string	gp_Time
Público		string	gp_SoundBackground

Público		string	gp_Lives
Público		bool	isStoredGameProperties
Público		Hashtable	xmlScreens
Privado	Estático, sólo lectura	StorePlatformVehicles	instance
Observaciones			

A continuación se adjuntan las clases del **subsistema Analizador** (proyecto *analizadorPlatformVehicles*, programado en C#)

7.3.2.11 Analyzer

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Analyzer	Pública	Será la clase encargada de analizar el fichero instancia del DSL.	
Responsabilidades			
Número	**Descripción**		
1	Copia la plantilla a la carpeta de destino incluyendo los niveles definidos en formato XML.		
2	Procesa el XML del DSL insertando los valores en un hashMap.		
3	Sustituye cada uno de los parámetros en el DSL.java		
4	Guarda el fichero DSL. java en su ruta correspondiente dentro del proyecto Android final.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	bool	copyToDestiny	
Público	bool	createDestinyStructure	String dir
Público	bool	analyzeDSL	
Público	void	copyLevelFilesToDestiny	String dir
Público	void	copyResources	String dir
Público	void	processElement	XmlNode node, XmlNode father
Público	void	processChildList	XmlNode father
Público	bool	substitute	

Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Público		Dictionary<String, String>	dict
Público		Reader	r
Público		String	resI
Público		String	resS
Observaciones			

7.3.2.12 Reader

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Reader	Publica	Clase que hace algunas comprobaciones para que la clase Analyzer pueda trabajar correctamente.	
Responsabilidades			
Número	**Descripción**		
1	Comprueba que la ruta raíz está definida.		
2	Crea la ruta de destino si no existe previamente		
3	Comprueba que el fichero dsl.xml existe en la rutaF.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	bool	searchRoutes	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Privado		String	rutaO
Privado		String	rutaD
Privado		String	rutaF
Observaciones			

Por último se adjuntan las clases más importantes del **subsistema videojuego** (código Java). Se ha utilizado JavaDocs para extraer la información de manera más directa.

7.3.2.13 Button

Nombre	Tipo	Descripción	Hereda de…
Button	Pública	Elemento de interacción con el usuario	Control
Responsabilidades			
Número	**Descripción**		
1	Dibuja el botón en el canvas.		
2	Comprueba si el usuario ha pulsado dicho botón.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
public	void	Draw(Canvas canvas)	
public	void	Draw(Canvas canvas, int color)	
public	java.lang.String	getFilename()	
public	boolean	isPressed(Vector2 click)	
public	void	setVisible(boolean visible)	
Atributos			
Acceso y tipo	**Nombre**		
private int	boundX		
private int	boundY		
(package private) HandlerResolution	hr		
Observaciones			

7.3.2.14 Checkpoint

Nombre	Tipo	Descripción	Hereda de…
Checkpoint	Pública	Elemento del nivel que recuerda la posición del personaje principal cuando éste pierde una vida.	LevelElement
Responsabilidades			
Número	**Descripción**		
1	Inicializa las propiedades de los puntos de control		
2	Cambia la animación y reproduce el sonido correspondiente cuando se detecta una colisión del personaje con el punto de control.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	

public	void	LoadContent(int numFrames , int textureWidth, int textureHeight, java.lang.String image, boolean hasLoop, int collectedNumFrames, int collectedTextureWidth, java.lang.String imageCollected, boolean collectedHasLoop, java.lang.String item_sound_on_collected_source)
public	void	onCollected()

Atributos	
Acceso y tipo	Nombre
private int	checkPointCollectedSound
private boolean	hasAlreadySound
private Animation	onCollectedAnimation
Observaciones	

7.3.2.15 DSL

Nombre	**Tipo**	**Descripción**	**Hereda de…**
DSL	Pública	Clase que recibe los parámetros configurables en el editor	
Responsabilidades			
Número	**Descripción**		
1	Almacena las propiedades seleccionadas en el editor como instancia del DSL del juego.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
Atributos			
Acceso y tipo		**Nombre**	
static int		entityLayers	
static java.lang.String		gameID	
static boolean		has_advertising	

static int	heightEffect
static int	heightEffectOn
static int	heightIconHealth
static int	heightIconPause
static int	heightImageArrowLeft
static int	heightImageArrowRight
static int	heightImageButtonExit
static int	heightImageButtonStart
static int	heightImageGameButton
static int	heightMusic
static int	heightMusicOn
static int	heightOptions
static java.lang.String	imageArrowLeft
static java.lang.String	imageArrowRight
static java.lang.String	imageButtonAcelerate
static int	imageButtonAcelerateHeight
static int	imageButtonAceleratePosX
static int	imageButtonAceleratePosY
static int	imageButtonAcelerateWidth
static java.lang.String	imageButtonBrake
static int	imageButtonBrakeHeight
static int	imageButtonBrakePosX
static int	imageButtonBrakePosY
static int	imageButtonBrakeWidth
static java.lang.String	imageButtonExit
static java.lang.String	imageButtonExitLevel
static int	imageButtonExitLevelHeight
static int	imageButtonExitLevelPosX

static int	imageButtonExitLevelPosY
static int	imageButtonExitLevelWidth
static java.lang.String	imageButtonGoBack
static int	imageButtonGoBackHeight
static int	imageButtonGoBackPosX
static int	imageButtonGoBackPosY
static java.lang.String	imageButtonGoBackToMenu
static int	imageButtonGoBackToMenuHeight
static int	imageButtonGoBackToMenuPosX
static int	imageButtonGoBackToMenuPosY
static int	imageButtonGoBackToMenuWidth
static int	imageButtonGoBackWidth
static java.lang.String	imageButtonNextLevel
static int	imageButtonNextLevelHeight
static int	imageButtonNextLevelPosX
static int	imageButtonNextLevelPosY
static int	imageButtonNextLevelWidth
static java.lang.String	imageButtonRestart
static int	imageButtonRestartHeight
static int	imageButtonRestartPosX
static int	imageButtonRestartPosY
static int	imageButtonRestartWidth
static java.lang.String	imageButtonRetry
static int	imageButtonRetryHeight
static int	imageButtonRetryPosX
static int	imageButtonRetryPosY
static int	imageButtonRetryWidth
static java.lang.String	imageButtonSelectLevel

static int	imageButtonSelectLevelHeight
static int	imageButtonSelectLevelPosX
static int	imageButtonSelectLevelPosY
static int	imageButtonSelectLevelWidth
static java.lang.String	imageButtonStart
static java.lang.String	imageButtonTryAgain
static int	imageButtonTryAgainHeight
static int	imageButtonTryAgainPosX
static int	imageButtonTryAgainPosY
static int	imageButtonTryAgainWidth
static java.lang.String	imageCuadroPause
static java.lang.String	imageEffect
static java.lang.String	imageEffectOn
static java.lang.String	imageGameButton
static java.lang.String	imageGameFinished
static java.lang.String	imageGameOver
static java.lang.String	imageIconHealth
static java.lang.String	imageIconPause
static java.lang.String	imageLevelFinished
static java.lang.String	imageMenu
static int	imageMenuHeight
static int	imageMenuPosX
static int	imageMenuPosY
static int	imageMenuWidth
static java.lang.String	imageMusic
static java.lang.String	imageMusicBack
static java.lang.String	imageMusicOn
static java.lang.String	imageOptions

static java.lang.String	imageSelectLevel
static int	imageSelectLevelHeight
static int	imageSelectLevelPosX
static int	imageSelectLevelPosY
static int	imageSelectLevelWidth
static java.lang.String	imageTimeBox
static int	imageTimeBoxHeight
static int	imageTimeBoxPosX
static int	imageTimeBoxPosY
static int	imageTimeBoxWidth
static java.lang.String	level10Time
static java.lang.String	level1Time
static java.lang.String	level2Time
static java.lang.String	level3Time
static java.lang.String	level4Time
static java.lang.String	level5Time
static java.lang.String	level6Time
static java.lang.String	level7Time
static java.lang.String	level8Time
static java.lang.String	level9Time
static java.lang.String	menuMusic
static int	miniaturaHeight
static int	miniaturaPosX
static int	miniaturaPosY
static int	miniaturaWidth
static int	numberOfLevels
static int	posXButtonExitMenu
static int	posXButtonStartMenu

static int	posXEffect
static int	posXEffectOn
static int	posXGameButton
static int	posXIconHealth
static int	posXIconPause
static int	posXImageArrowLeft
static int	posXImageArrowRight
static int	posXMusic
static int	posXMusicOn
static int	posXOptions
static int	posXTimeBoard
static int	posXtimeLevel
static int	posYButtonExitMenu
static int	posYButtonStartMenu
static int	posYEffect
static int	posYEffectOn
static int	posYGameButton
static int	posYIconHealth
static int	posYIconPause
static int	posYImageArrowLeft
static int	posYImageArrowRight
static int	posYMusic
static int	posYMusicOn
static int	posYOptions
static int	posYTimeBoard
static int	posYtimeLevel
static java.lang.String	timeBoardColor
static java.lang.String	timeBoxTextColor

static java.lang.String	timeLevelColor
static int	timeLevelHeight
static int	timeLevelWidth
static long	warningTime
static int	widthEffect
static int	widthEffectOn
static int	widthIconHealth
static int	widthIconPause
static int	widthImageArrowLeft
static int	widthImageArrowRight
static int	widthImageButtonExit
static int	widthImageButtonStart
static int	widthImageGameButton
static int	widthMusic
static int	widthMusicOn
static int	widthOptions
static int	posYButtonExitMenu
static int	posYButtonStartMenu
static int	posYEffect
static int	posYEffectOn
static int	posYGameButton
static int	posYIconHealth
static int	posYIconPause
static int	posYImageArrowLeft
static int	posYImageArrowRight
static int	posYMusic
static int	posYMusicOn
static int	posYOptions

static int	posYTimeBoard
static int	posYtimeLevel
static java.lang.String	timeBoardColor
static java.lang.String	timeBoxTextColor
static java.lang.String	timeLevelColor
static int	timeLevelHeight
static int	timeLevelWidth
static long	warningTime
static int	widthEffect
static int	widthEffectOn
static int	widthIconHealth
static int	widthIconPause
static int	widthImageArrowLeft
static int	widthImageArrowRight
static int	widthImageButtonExit
static int	widthImageButtonStart
static int	widthImageGameButton
static int	widthMusic
static int	widthMusicOn
static int	widthOptions
Observaciones	

7.3.2.16 EndGameActivity

Nombre	**Tipo**	**Descripción**	**Hereda de…**
EndGameActivity	Pública	Actividad de la pantalla de fin de juego (todos los niveles superados)	Activity
Responsabilidades			
Número	**Descripción**		
1	Crea su *View* correspondiente y la *AdView* (publicidad).		
2	Gestiona la navegabilidad hacia otras pantallas.		

Métodos		
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**
public	void	goToMainMenu()
public	void	killActivity()
public	void	onBackPressed()
public	void	onCreate(Bundle savedInstanceState)
Atributos		
Acceso y tipo		Nombre
Observaciones		
Se han obviado de esta sección el resto de *Activities* por ser prácticamente análogas a la presente.		

7.3.2.17 EndGameView

Nombre	**Tipo**	**Descripción**	**Hereda de…**
EndGameView	Pública	Clase que gestiona la funcionalidad de la pantalla fin de juego	View
Responsabilidades			
Número	**Descripción**		
1	Muestra la imagen del DSL para todos los nvieles completados.		
2	Detecta cuando el usuario pulsa la tecla “Volver al menú”		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
private	void	initialize(Context context)	
protected	void	onDraw(Canvas canvas)	
public	boolean	onTouchEvent(MotionEvent event)	
Atributos			
Acceso y tipo		**Nombre**	
private Drawable		background	
(package private) Context		context	
private Button		exitButton	
private int		screenHeight	
private int		screenWidth	
Observaciones			
Se han obviado de esta sección el resto de *Views* por ser prácticamente análogas a la presente.			

7.3.2.18 GameLoop

Nombre	Tipo	Descripción	Hereda de…
GameLoop	Pública	Hilo que actualiza la lógica del juego.	Runnable (interfaz), Thread
Responsabilidades			
Número	**Descripción**		
1	Se encarga de inicializar la GameView y gestionar los *frames* del videojuego llamando a las funciones del GameView que necesitan ser actualizadas cada *frame*.		
2	Controla los *frames* saltados por el dispositivo para evitar "saltos" en el juego.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
Public	void	run()	
Public	void	setRunning(boolean running)	
Atributos			
Acceso y tipo	**Nombre**		
private boolean	corriendo		
private static int	FPS		
private static long	FRAME_TIEMPO		
private GameView	gameView		
private static int	MAXIMO_FRAMES_SALTARSE		
private SurfaceHolder	superficie		
Observaciones			

7.3.2.19 GameView

Nombre	Tipo	Descripción	Hereda de…
GameView	Pública	Clase que gestiona todos los elementos que intervienen en la lógica del videojuego	SurfaceView
Responsabilidades			
Número	**Descripción**		
1	Maneja el ciclo del juego en tanto que dibuja en pantalla los elementos y actualiza su comportamiento.		
2	Se encarga de inicializar todos los botones en la pantalla y carga el nivel correspondiente		
3	Comprueba y gestiona los estados del juego		
4	Procesa la entrada táctil del usuario.		

Métodos		
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**
private	void	checkRestartButton(long gameTime, MotionEvent me2)
protected	void	Draw(long frameTiempo, Canvas canvas)
private	void	DrawHud(Canvas canvas)
package private)	void	exitLevel()
public	int	getLevelIndex()
static	int	getNumberOfLevels()
private	Rect	getRectangle(Drawable d)
public	void	gl_carPhysics()
private	void	HandleInput()
public	void	init()
protected	void	LoadContent()
private	void	LoadNextLevel()
public	boolean	onTouchEvent(MotionEvent event)
public	void	playEffect(int sound)
private	void	ReloadCurrentLevel()
public	void	SavePreviousLevel()
public	void	setGameOver(boolean g)
public	void	setLevelFinished(boolean l)
public	void	setPaused(boolean p)
public	void	surfaceChanged(SurfaceHolder holder, int format, int width, int height)
public	void	surfaceCreated(SurfaceHolder holder)
public	void	surfaceDestroyed(SurfaceHolder holder)
public	void	Update(long gameTime)
Atributos		
Acceso y tipo		**Nombre**

(package private) boolean	_start
private Button	bAcelerate
private Button	bBrake
private Button	bRestart
Context	context
private Button	descriptionLevelButton
private boolean	descriptionScreenIsActivated
private Button	exitButton
private boolean	gameFinished
GameLoop	gameloop
private boolean	gameOver
private Drawable	gameOverMessage
private Button	goBackButton
(package private) float	KeyPressCheckDelay
private Level	level
private boolean	levelFinished
private Drawable	levelFinishedMessage
private int	levelIndex
private MotionEvent	me
private Button	nextLevelButton
private static int	numberOfLevels
private Button	pauseButton
private boolean	paused
private Drawable	pauseMessage
private Button	retryButton
private int	screenHeight
private boolean	screenTouched
private int	screenWidth

private Button	selectLevelButton
private Button	timeBox
(package private) float	TotalElapsedTime
private Button	tryAgainButton
private static long	warningTime
private boolean	wasContinuePressed
Observaciones	

7.3.2.20 Level

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Level	Pública	Almacena todas las propiedades del nivel: personaje, Tiles, puntos de control, trampas y la meta, además de gestionar su funcionalidad.	
Responsabilidades			
Número	**Descripción**		
1	Se encarga de inicializar los ladrillos del nivel, el fondo, el jugador principal, las trampas y los puntos de control.		
2	Actualiza la lógica de los elementos y los pinta en la pantalla.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
public	void	Draw(long gameTime, Canvas canvas)	
private	void	DrawBackgrounds(Canvas canvas)	
private	void	DrawTiles(Canvas canvas)	
public	Rect	GetBounds(int x, int y)	
public	Context	getContext()	
public	Data	getLevelLoaded()	
public	Player	getPlayer()	
public	int	getSoundAcelerate()	
public	int	getSoundBrake()	

public	int	getStageGameWidth()
public	Tile	getTile(int x, int y)
public	long	getTime()
public	long	getTimeToAchieve()
public	int	height()
public	boolean	isLoaded()
public	boolean	isPlayerAlive()
public	boolean	isReachedExit()
private	void	LoadBackgrounds()
private	Tile	LoadCheckPointTile(int checkPointWidth, int checkPointHeight, java.lang.String imageDefault, int checkPointNumFrames, int checkPointSpriteWidth, int checkPointSpriteHeight, int x, int y, boolean imageHasLoop, int checkPointCollectedWidth, java.lang.String imageCollectedDefault, int checkPointCollectedNumFrames, int checkPointCollectedSpriteWidth, boolean collectedHasLoop, java.lang.String checkPoint_sound_on_collected_source)
private	Tile	LoadExitTile(java.lang.String name, int x, int y)
private	Tile	LoadStartTile(int x, int y, int character_width, int character_height, int character_collision_area_left, int character_collision_area_top, int character_collision_area_right, int character_collision_area_down, java.lang.String character_sound_acelerate_source, java.lang.String character_sound_brake_source, java.lang.String character_sound_die_source, double character_acelerate_factor, double character_brake_factor, double character_friction_factor, double character_max_positive_aceleration, double character_max_negative_aceleration, double character_max_friction_slope,

		double character_gravity, boolean character_inertia, int character_health, java.lang.String character_sprite_default_left_source, int character_sprite_default_left_width, int character_sprite_default_left_height, int character_sprite_default_left_frames, boolean defaultLeftHasLoop, java.lang.String character_sprite_default_right_source, int character_sprite_default_right_width, int character_sprite_default_right_height, int character_sprite_default_right_frames, boolean defaultRightHasLoop, java.lang.String character_sprite_left_source, int character_sprite_left_width, int character_sprite_left_height, int character_sprite_left_frames, boolean movingLeftHasLoop, java.lang.String character_sprite_right_source, int character_sprite_right_width, int character_sprite_right_height, int character_sprite_right_frames, boolean movingRightHasLoop, java.lang.String character_sprite_slope22up_left_source, int character_sprite_slope22up_left_width, int character_sprite_slope22up_left_height, int character_sprite_slope22up_left_frames, boolean slope22UpLeftHasLoop, java.lang.String character_sprite_slope22up_right_source, int character_sprite_slope22up_right_width, int character_sprite_slope22up_right_height, int character_sprite_slope22up_right_frames, boolean slope22UpRightHasLoop, java.lang.String character_sprite_slope22down_left_source, int character_sprite_slope22down_left_width, int character_sprite_slope22down_left_height, int character_sprite_slope22down_left_frames, boolean slope22DownLeftHasLoop, java.lang.String character_sprite_slope22down_right_source, int character_sprite_slope22down_right_width, int character_sprite_slope22down_right_height, int character_sprite_slope22down_right_frames, boolean slope22DownRightHasLoop, java.lang.String character_sprite_slope45up_left_so

		urce, int character_sprite_slope45up_left_width, int character_sprite_slope45up_left_height, int character_sprite_slope45up_left_frames, boolean slope45UpLeftHasLoop, java.lang.String character_sprite_slope45up_right_source, int character_sprite_slope45up_right_width, int character_sprite_slope45up_right_height, int character_sprite_slope45up_right_frames, boolean slope45UpRightHasLoop, java.lang.String character_sprite_slope45down_left_source, int character_sprite_slope45down_left_width, int character_sprite_slope45down_left_height, int character_sprite_slope45down_left_frames, boolean slope45DownLeftHasLoop, java.lang.String character_sprite_slope45down_right_source, int character_sprite_slope45down_right_width, int character_sprite_slope45down_right_height, int character_sprite_slope45down_right_frames, boolean slope45DownRightHasLoop, java.lang.String character_sprite_slope67up_left_source, int character_sprite_slope67up_left_width, int character_sprite_slope67up_left_height, int character_sprite_slope67up_left_frames, boolean slope67UpLeftHasLoop, java.lang.String character_sprite_slope67up_right_source, int character_sprite_slope67up_right_width, int character_sprite_slope67up_right_height, int character_sprite_slope67up_right_frames, boolean slope67UpRightHasLoop, java.lang.String character_sprite_slope67down_left_source, int character_sprite_slope67down_left_width, int character_sprite_slope67down_left_height, int character_sprite_slope67down_left_frames, boolean slope67DownLeftHasLoop, java.lang.String character_sprite_slope67down_right_source, int character_sprite_slope67down_right_width, int character_sprite_slope67down_right_height, int character_sprite_slope67down_right_frames, boolean slope67DownRightHasLoop)
private	Tile	LoadTile(java.lang.String name,

		java.lang.String collision)
public	void	LoadTiles()
private	Tile	LoadTrapTile(int trapWidth, int trapHeight, java.lang.String imageDefault, int trapNumFrames, int trapSpriteWidth, int trapSpriteHeight, int x, int y, boolean imageHasLoop)
private	void	OnCheckPointCollected(CheckPoint cPoint, Player co llectedBy)
private	void	OnExitReached()
public	boolean	playerHasLives()
public	int	ScreenHeight()
public	int	ScreenWidth()
private	void	ScrollCamera(Canvas canvas)
public	void	StartNewLife()
public	void	Update(long gameTime)
private	void	UpdateCheckPoints()
private	void	UpdateTraps()
public	int	width()
Atributos		
Acceso y tipo		**Nombre**
private float		cameraPosition
float		cameraPositionY
(package private) LoaderLevel		charger
Context		context
private java.util.List<CheckPoint>		cPoints
private int		distanceToStartScroll
private int		EntityLayer
private Vector2		exit
(package private) int		exitReachedSound
private static Vector2		InvalidPosition

private boolean	isLoaded
private Layer[]	layers
(package private) Data	levelLoaded
private LifeBoard	lifeBoard
private Player	player
(package private) boolean	reachedExit
int	screenHeight
int	screenWidth
private int	stageGameWidth
private Vector2	start
private Tile[][]	tiles
private long	time
private long	timeToAchieve
private java.util.List<Trap>	traps
Observaciones	

7.3.2.21 LevelElement

Nombre	**Tipo**	**Descripción**	**Hereda de…**
LevelElement	Pública, abstracta	Clase abstracta que contiene los elementos comunes de los objetos del nivel.	
Responsabilidades			
Número	**Descripción**		
1	Establece las propiedades y métodos comunes a los elementos del nivel		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
Public	void	Draw(long gameTime, Canvas canvas)	
Public	Circle	getBoundingCircle()	
Public	Level	getLevel()	
Public	int	getNumFrames()	
Public	Vector2	getPosition()	

Public	int	getTextureHeight()
Public	float	getX()
Public	float	getY()
Public	boolean	HasLoop()
Public abstract	void	onCollected()
Public	void	setNumFrames(int n)
Public	void	setTextureHeight(int textureHeight)
Public	void	setTextureWidth(int textureWidth)
Atributos		
Acceso y tipo		Nombre
protected boolean		hasLoop
protected int		height
protected Animation		idleAnimation
(package private) Level		level
protected int		numFrames
protected Vector2		position
protected AnimationPlayer		sprite
protected int		textureHeight
protected int		textureWidth
protected int		width
Observaciones		

7.3.2.22 LoaderLevel

Nombre	Tipo	Descripción	Hereda de…
LoaderLevel	Pública	Lee los ficheros XML del nivel y carga la información en un contenedor de tipo *Data*	
Responsabilidades			
Número	**Descripción**		
1	Carga el XML y parsea cada propiedad en su objeto dentro de Data.		
Métodos			

Acceso | Modo	Tipo de Retorno	Nombre, parámetros y tipos.
public	Data	Load()
private	Data	LoadXML(java.lang.String filename)
Atributos		
Acceso y tipo		**Nombre**
private Context		context
private java.lang.String		filenameXML
Observaciones		

7.3.2.23 PlatformerDroidActivity

Nombre	Tipo	Descripción	Hereda de…
PlatformerDroidActivity	Pública	Actividad que lanza la *View* de la lógica del juego.	Activity
Responsabilidades			
Número	**Descripción**		
1	Lanza la *GameView* y la *AdView* para ejecutar el videojuego		
2	Gestiona la pulsación del botón para ir a la *EndGameActivity*		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
public	void	goToEndGameActivity()	
public	void	killLoadingActivity()	
private	void	loadCurrentLevel()	
public	void	onBackPressed()	
public	void	onCreate(Bundle savedInstanceState)	
public	boolean	onTouchEvent(MotionEvent event)	
Atributos			
Acceso y tipo		**Nombre**	
(package private) com.google.ads.AdView		adView	
(package private) GameView		gameView	
Observaciones			

7.3.2.24 Player

Nombre	Tipo	Descripción	Hereda de…
Player	Pública	Personaje principal y único de cada nivel. Esta clase gestiona la física del personaje con el entorno.	
Responsabilidades			
Número	**Descripción**		
1	Inicializa el personaje principal con todas sus propiedades.		
2	Gestiona la física, colisiones con *Tiles,* etc.		
3	Dibuja las diferentes animaciones del personaje ante determinados eventos.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
public	void	acelerate(boolean forward)	
private	void	changeAnimation(Tile.TileCollision collision, double dx)	
private	void	correctPosition()	
public	void	decelerate()	
public	void	Draw(long gameTime, Canvas canvas)	
public	void	flipLeft()	
public	void	flipRight()	
public	double	getBottomY()	
public	Rect	getBoundingRectangle()	
public	double	getGravity()	
public	double	getLeftX()	
public	int	getLives()	
public	double	getRightX()	
public	Vector2	getSavedPosition()	
public	int	getSoundAcelerate()	
public	int	getSoundBrake()	
public	int	getSoundDie()	

public	Tile	getTile(double xPos, double yPos)
public	int	getWidth()
public	double	getX()
public	double	getY()
public	void	gravity()
public	void	handleCollisions()
public	boolean	isAlive()
private	boolean	isBetweenSlopes()
public	boolean	isCollisionSlopeDown(Tile.TileCollision col)
private	boolean	isCollisionSlopeUp(Tile.TileCollision col)
public	boolean	isEnteringSlope(double dx)
public	boolean	isLeavingSlope(double dx)
public	boolean	isOnImpassableSlope()
public	boolean	isOnSlopeDown()
public	boolean	isOnSlopeUp()
private	boolean	isTopLeftImpassable()
private	boolean	isTopRightImpassable()
public	void	LoadContent(Vector2 position, java.lang.String imageIdleLeft, int imageIdleLeftWidth, int imageIdleLeftHeight, int imageIdleLeftNumFrames, boolean imageIdleLeftHasLoop, java.lang.String imageIdleRight, int imageIdleRightWidth, int imageIdleRightHeight, int imageIdleRightNumFrames, boolean imageIdleRightHasLoop, java.lang.String imageRunLeft, int imageRunLeftWidth, int imageRunLeftHeight, int imageRunLeftNumFrames, boolean imageRunLeftHasLoop, java.lang.String imageRunRight, int imageRunRightWidth, int imageRunRightHeight, int imageRunRightNumFrames, boolean imageRunRightHasLoop,

		java.lang.String imageSlope22UpLeft, int imageSlope22UpLeftWidth, int imageSlope22UpLeftHeight, int imageSlope22UpLeftNumFrames, boolean imageSlope22UpLeftHasLoop, java.lang.String imageSlope22UpRight, int imageSlope22UpRightWidth, int imageSlope22UpRightHeight, int imageSlope22UpRightNumFrames, boolean imageSlope22UpRightHasLoop, java.lang.String imageSlope22DownLeft, int imageSlope22DownLeftWidth, int imageSlope22DownLeftHeight, int imageSlope22DownLeftNumFrames, boolean imageSlope22DownLeftHasLoop, java.lang.String imageSlope22DownRight, int imageSlope22DownRightWidth, int imageSlope22DownRightHeight, int imageSlope22DownRightNumFrames, boolean imageSlope22DownRightHasLoop, java.lang.String imageSlope45UpLeft, int imageSlope45UpLeftWidth, int imageSlope45UpLeftHeight, int imageSlope45UpLeftNumFrames, boolean imageSlope45UpLeftHasLoop, java.lang.String imageSlope45UpRight, int imageSlope45UpRightWidth, int imageSlope45UpRightHeight, int imageSlope45UpRightNumFrames, boolean imageSlope45UpRightHasLoop, java.lang.String imageSlope45DownLeft, int imageSlope45DownLeftWidth, int imageSlope45DownLeftHeight, int imageSlope45DownLeftNumFrames, boolean imageSlope45DownLeftHasLoop, java.lang.String imageSlope45DownRight, int imageSlope45DownRightWidth, int imageSlope45DownRightHeight, int imageSlope45DownRightNumFrames, boolean imageSlope45DownRightHasLoop, java.lang.String imageSlope67UpLeft, int imageSlope67UpLeftWidth, int imageSlope67UpLeftHeight, int imageSlope67UpLeftNumFrames, boolean imageSlope67UpLeftHasLoop,

		java.lang.String imageSlope67UpRight, int imageSlope67UpRightWidth, int imageSlope67UpRightHeight, int imageSlope67UpRightNumFrames, boolean imageSlope67UpRightHasLoop, java.lang.String imageSlope67DownLeft, int imageSlope67DownLeftWidth, int imageSlope67DownLeftHeight, int imageSlope67DownLeftNumFrames, boolean imageSlope67DownLeftHasLoop, java.lang.String imageSlope67DownRight, int imageSlope67DownRightWidth, int imageSlope67DownRightHeight, int imageSlope67DownRightNumFrames, boolean imageSlope67DownRightHasLoop, java.lang.String soundOnAcelerate, java.lang.String soundOnBrake, java.lang.String soundOnDie)
public	void	OnKilled()
private	boolean	outOfBoundsLeft()
private	boolean	outOfBoundsRight()
private	void	reset(Vector2 start)
public	void	setSavedPosition(Vector2 savedPosition)
public	void	setX(double x)
public	void	setY(double y)
public	void	StartNewLife()
public	double	Update()
public	void	updatePoints()
Atributos		
Acceso y tipo		**Nombre**
private double		aceleracionY
private double		accelerateFactor
private double		accelerationX
private Tile		actualTile
private Tile.TileCollision		bottomCollision

double	bottomX
double	bottomY
private double	brakeFactor
private int	collisionAreaBottom
private int	collisionAreaLeft
private int	collisionAreaRight
private int	collisionAreaTop
private double	frictionFactor
private boolean	goesRight
private double	gravity
private int	height
private Animation	idleAnimationLeft
private Animation	idleAnimationRight
private boolean	inertia
private boolean	isAlive
private boolean	isCorrected
private boolean	isOnAir
private Tile.TileCollision	leftCollision
private double	leftX
private double	leftY
private Level	level
private int	lives
private double	maxFrictionOnSlope
private double	maxNegAceleration
private double	maxPosAceleration
private double	previousAceleration
private Tile.TileCollision	rightCollision
private double	rightX

private double	rightY
private Animation	runAnimationLeft
private Animation	runAnimationRight
private Vector2	savedPosition
private Animation	slopeDown22AnimationLeft
private Animation	slopeDown22AnimationRight
private Animation	slopeDown45AnimationLeft
private Animation	slopeDown45AnimationRight
private Animation	slopeDown67AnimationLeft
private Animation	slopeDown67AnimationRight
private Animation	slopeUp22AnimationLeft
private Animation	slopeUp22AnimationRight
private Animation	slopeUp45AnimationLeft
private Animation	slopeUp45AnimationRight
private Animation	slopeUp67AnimationLeft
private Animation	slopeUp67AnimationRight
private int	soundAcelerate
private int	soundBrake
private int	soundDie
private AnimationPlayer	sprite
private Tile.TileCollision	topLeftCollision
private double	topLeftY
private Tile.TileCollision	topRightCollision
private double	topRightY
private int	width
private double	x
private double	y
Observaciones	

7.3.2.25 SoundEngine

Nombre	Tipo	Descripción	Hereda de…
SoundEngine	Pública	Clase que gestiona la música y efectos sonoros del juego	
Responsabilidades			
Número	**Descripción**		
1	Almacena en un array todos los sonidos del juego.		
2	Se encarga de reproducirlos, pausarlos y pararlos cuando el sonido está activado.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
public	java.lang.Float	getEffectsVolume()	
public	java.lang.Float	getSoundsVolume()	
public	boolean	isMute()	
public	void	mute()	
public	void	pauseSound()	
public	void	playEffect(Context app, int resId)	
public	void	playSound(Context ctxt, int resId, boolean loop)	
public	void	preloadEffect(Context app, int resId)	
public	void	preloadSound(Context ctxt, int resId)	
public static	void	purgeSharedEngine()	
public	void	realesAllEffects()	
public	void	realesAllSounds()	
public	void	realesSound(int resId)	
public	void	resumeSound()	
public	void	setEffectsVolume(java.lang.Float volume)	
public	void	setSoundVolume(java.lang.Float volume)	
public static	SoundEngine	sharedEngine()	
public	void	stopEffect(Context app, int resId)	

public	void	unmute()
Atributos		
Acceso y tipo		**Nombre**
(package private) static SoundEngine		_sharedEngine
(package private) IntMap<java.lang.Integer>		effectsMap
(package private) java.lang.Float		effectsVolume
(package private) int		lastSndId
(package private) boolean		mute
(package private) java.lang.Float		prevEffectsVolume
(package private) java.lang.Float		prevSoundsVolume
(package private) IntMap<MediaPlayer>		soundsMap
(package private) java.lang.Float		soundsVolume
(package private) SoundPool		sp
(package private) IntMap<java.lang.Integer>		streamsMap
Observaciones		

7.3.2.26 Tile

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Tile	Pública	Clase que alberga las propiedades de los *Tiles*	
Responsabilidades			
Número	**Descripción**		
1	Inicializar las propiedades de un *Tile.*		
2	Devolver el factor de fricción dependiendo de la colisión.		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
public	Tile.TileCollision	getCollision()	
public	double	getFrictionFactor()	
Atributos			
Acceso y tipo		**Nombre**	
Tile.TileCollision		Collision	

static double	frictionFactor22
static double	frictionFactor45
static double	frictionFactor67
static int	Height
static Vector2	Size
Drawable	Texture
static int	Width
Observaciones	

7.3.2.27 Trap

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Trap	Pública	Elemento del nivel que resta una vida al personaje si éste colisiona con él.	LevelElement
Responsabilidades			
Número	**Descripción**		
1	Inicializa la trampa con sus propiedades		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre, parámetros y tipos.**	
public	void	LoadContent(int numFrames, int textureWidth, int textureHeight, java.lang.String name, boolean hasLoop)	
public	void	onCollected()	
Atributos			
Acceso y tipo		Nombre	
Observaciones			
Los atributos de *Trap* son los heredados de *LevelElement*.			

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

8.1.1.1 Configuración propiedades Globales

Configuración propiedades globales Clase **PageGlobalsPlatformVehicles.xaml**	
Caso de prueba 1.1	**Resultado Esperado**
Insertar un nombre inválido (con tildes, espacios, caracteres especiales, etc.) para el videojuego.	El sistema informa de un error cuando el usuario intenta avanzar a la siguiente pantalla de edición.
	Resultado Obtenido
	Aparece el error "El nombre del juego sólo puede contener letras, números y '.'" No se permite avanzar a la siguiente pantalla de edición.
Caso de prueba 1.2	**Resultado Esperado**
Insertar un nombre vacío para el videojuego.	El sistema informa de un error cuando el usuario intenta avanzar a la siguiente pantalla de edición.
	Resultado Obtenido
	Al realizar la prueba el sistema no dejaba continuar con la edición. Para mejorar la ayuda al usuario se ha incluido el mensaje de error "El nombre del juego no puede ser vacío".
Caso de prueba 1.3	**Resultado Esperado**
Insertar un nombre válido para el videojuego.	El editor permite continuar con el proceso de edición y cuando ésta finalice se obtiene un videojuego con el nombre asignado previamente.
	Resultado Obtenido
	El editor permite continuar con el proceso de edición.

Caso de prueba 1.4	**Resultado Esperado**
Escoger un icono para la aplicación que no es formato .png.	El editor no permite seleccionar imágenes con formato distinto a .png. El explorador de carpetas para escoger el icono tiene un filtro, así que solamente mostrará imágenes con esta extensión.
	Resultado Obtenido
	El editor no permite seleccionar imágenes con formato distinto a .png.
Caso de prueba 1.5	**Resultado Esperado**
Escoger un icono .png para la aplicación.	El editor muestra una pre visualización de la imagen escogida y se asignará al videojuego.
	Resultado Obtenido
	El editor muestra una pre visualización de la imagen escogida y se asignará al videojuego.
Caso de prueba 1.6	**Resultado Esperado**
Marcar la opción de publicidad y no se añade ningún ID.	El sistema avisa de que se necesita un ID para mostrar la publicidad.
	Resultado Obtenido
	El sistema muestra el mensaje de error “Debe rellenar al menos un identificador de la publicidad ya que ha seleccionado que el juego tenga publicidad”.
Caso de prueba 1.7	**Resultado Esperado**
Marcar la opción de publicidad e incluir un ID.	Se asigna el valor al identificador de la publicidad y se muestra dicha publicidad en el videojuego final.
	Resultado Obtenido
	Se asigna el valor al identificador de la publicidad y se muestra dicha publicidad en el videojuego final.

Caso de prueba 1.8	**Resultado Esperado**
Pulsar en el botón “siguiente”	Se crean las carpetas necesarias en la ruta escogida por el usuario para almacenar los diferentes elementos globales que ya se han definido.
	Resultado Obtenido
	Se crean las carpetas necesarias en la ruta escogida por el usuario para almacenar los diferentes elementos globales que ya se han definido.
Caso de prueba 1.9	**Resultado Esperado**
Elegir una melodía para los menús del juego con formato .mp4	La aplicación no permite seleccionar archivos de sonido con formato distinto a .mp3 o .wav. El explorador de carpetas para escoger el sonido tiene un filtro, así que solamente mostrará archivos con dicha extensión.
	Resultado Obtenido
	La aplicación no permite seleccionar archivos de sonido con formato distinto a .mp3 o .wav.
Caso de prueba 1.10	**Resultado Esperado**
Seleccionar un archivo de sonido en formato .mp3	La aplicación permite escucharlo y lo asigna ante como música para los menús.
	Resultado Obtenido
	La aplicación permite escucharlo y lo asigna como música para los menús.
Caso de prueba 1.11	**Resultado Esperado**
Pulsar el botón de reproducir sin haber escogido previamente un sonido.	El sonido no se reproduce.
	Resultado Obtenido
	El sonido no se reproduce y se muestra el mensaje “Introduzca un sonido primero” para informar al usuario debidamente.

Caso de prueba 1.12	**Resultado Esperado**
Pulsar el botón de reproducir habiendo escogido previamente un sonido.	El sonido se reproduce sin problemas.
	Resultado Obtenido
	El sonido se reproduce sin problemas.
Caso de prueba 1.13	**Resultado Esperado**
Pulsar sobre el botón de pausa una vez el sonido esté reproduciéndose.	El sonido se para.
	Resultado Obtenido
	El sonido se para.
Caso de prueba 1.14	**Resultado Esperado**
No seleccionar ningún sonido para un determinado evento.	El editor permite no asignar sonidos a eventos.
	Resultado Obtenido
	El editor permite no asignar sonidos a eventos.
Caso de prueba 1.15	**Resultado Esperado**
Definir todos los sonidos necesarios y se continúa la edición.	Todos los sonidos seleccionados se incluirán en una carpeta sounds. La correspondencia entre sonidos y eventos se verá reflejada en el xml del nivel.
	Resultado Obtenido
	Todos los sonidos seleccionados se incluirán en una carpeta sounds. La correspondencia entre sonidos y eventos se verá reflejada en el xml del nivel.

8.1.1.2 Configuración pantallas

Configuración pantallas Clases **PageMainScreen.xaml,PageOptionsScreenPlatformVehicles.xaml, PageSelectLevelScreenPlatformVehicles.xaml, PageGameScreenPlatformVehicles.xaml, PagePauseScreen.xaml, PageEndLevelScreenPlatformVehicles.xaml, PageLoseLevelScreenPlatformVehicles.xaml, PageEndGameScreen.xaml.**	
Caso de prueba 2.1	**Resultado Esperado**
No definir la pantalla principal.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
	Resultado Obtenido
	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
Caso de prueba 2.2	**Resultado Esperado**
No definir la pantalla de opciones.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
	Resultado Obtenido
	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
Caso de prueba 2.3	**Resultado Esperado**
No definir la pantalla de victoria.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
	Resultado Obtenido
	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
Caso de prueba 2.4	**Resultado Esperado**
No definir la pantalla de derrota.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
	Resultado Obtenido
	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.

Caso de prueba 2.5	**Resultado Esperado**
No definir la pantalla de fin.	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
	Resultado Obtenido
	Al tratarse de una pantalla obligatoria, tomará los valores por defecto y se creará igualmente.
Caso de prueba 2.6	**Resultado Esperado**
Disponer los elementos de la pantalla de juego en el medio de la misma.	El editor no dará ningún error aunque es lógico que de esta manera no se visualice correctamente el juego. Este aspecto de disposición de los elementos se deja a elección del diseñador.
	Resultado Obtenido
	El editor no dará ningún error y aceptará la configuración.
Caso de prueba 2.7	**Resultado Esperado**
Seleccionar un botón.	El botón se selecciona dentro de un recuadro.
	Resultado Obtenido
	El botón se selecciona dentro de un recuadro.
Caso de prueba 2.8	**Resultado Esperado**
Escoger un fondo no .png	El editor no permite seleccionar imágenes con formato distinto a .png. El explorador de carpetas para escoger el icono tiene un filtro, así que solamente mostrará imágenes con esta extensión.
	Resultado Obtenido
	El editor no permite seleccionar imágenes con formato distinto a .png.

Caso de prueba 2.9	**Resultado Esperado**
Cambiar la posición de un botón.	El botón se mueve y aparece en dicha posición en el juego final.
	Resultado Obtenido
	El botón se mueve y aparece en dicha posición en el juego final.
Caso de prueba 2.10	**Resultado Esperado**
Ocultar un botón	El botón baja de opacidad y no aparece en el juego final.
	Resultado Obtenido
	El botón baja de opacidad y no aparece en el juego final.

8.1.1.3 Configuración personaje principal

Configuración personaje principal Clase **CharacterSetPropertiesPlatformVehicles**	
Caso de prueba 3.1	**Resultado Esperado**
Definir la misma imagen para todos los eventos.	El editor asignará la misma imagen para todos los eventos sin errores.
	Resultado Obtenido
	El editor asignará la misma imagen para todos los eventos sin errores.
Caso de prueba 3.2	**Resultado Esperado**
Seleccionar un personaje ya creado de la biblioteca.	El editor cargará dicho personaje y sus animaciones.
	Resultado Obtenido
	El editor cargará dicho personaje y sus animaciones.

Caso de prueba 3.3	**Resultado Esperado**
Definir una animación para algún evento del personaje.	Se mostrará una vista previa en el editor que recoja los parámetros introducidos para definir la animación (imágenes, número de frames, si debe reproducirse en bucle, etc.)
	Resultado Obtenido
	Se mostrará una vista previa en el editor que recoja los parámetros introducidos para definir la animación.
Caso de prueba 3.4	**Resultado Esperado**
Pulsar el botón "Añadir a biblioteca" cuando no hay ningún personaje creado.	Se lanza el mensaje de error correspondiente y no se guarda nada.
	Resultado Obtenido
	Se lanza el mensaje de error "Debe especificarse la animación por defecto" y no se guarda nada.
Caso de prueba 3.5	**Resultado Esperado**
Pulsar el botón "Añadir a biblioteca" cuando se ha finalizado la creación de un personaje.	El personaje se agrega correctamente a la biblioteca para poder ser usado posteriormente.
	Resultado Obtenido
	El personaje se agrega correctamente a la biblioteca para poder ser usado posteriormente.
Caso de prueba 3.6	**Resultado Esperado**
Pulsar el botón "Borrar" seleccionando un personaje de la biblioteca.	El personaje se borra de la biblioteca si no está vinculado a ningún escenario.
	Resultado Obtenido
	El personaje se borra de la biblioteca si no está vinculado a ningún escenario.

Caso de prueba 3.7	**Resultado Esperado**
Seleccionar un personaje de la biblioteca, cambiar alguna de sus propiedades y hacer clic en el botón de "Actualizar".	Si las propiedades son válidas, se actualiza dicha información en la base de datos de la biblioteca.
	Resultado Obtenido
	Si las propiedades son válidas, se actualiza dicha información en la base de datos de la biblioteca.
Caso de prueba 3.8	**Resultado Esperado**
Insertar un valor negativo para aquellos factores que no pueden serlo (gravedad, factor de fricción, etc.)	El valor negativo no se asigna. Se preparará el editor para que no acepte números negativos para determinados elementos.
	Resultado Obtenido
	El valor negativo no se asigna. Los factores que influyen en la física del juego se escogen a través de listas desplegables que sólo ofrecen valores válidos.
Caso de prueba 3.9	**Resultado Esperado**
No modificar ninguno de los valores que afecten a la física del juego.	La física del videojuego será la que está definida por defecto en el editor.
	Resultado Obtenido
	La física del videojuego será la que está definida por defecto en el editor.
Caso de prueba 3.10	**Resultado Esperado**
Insertar un valor de gravedad igual a cero.	El editor no permitirá asignar valores iguales o menores que cero para la gravedad.
	Resultado Obtenido
	El editor no permitirá asignar valores iguales o menores que cero para la gravedad. El selector tiene 1 como valor mínimo.

Caso de prueba 3.11	**Resultado Esperado**
Insertar un valor de factor de aceleración igual a cero.	El editor permitirá asignar esos valores pero mostrará un mensaje indicando que los factores de aceleración que sean igual a cero impedirán que el personaje se mueva en una u otra dirección.
	Resultado Obtenido
	El editor permitirá asignar esos valores pero mostrará un mensaje indicando que los factores de aceleración que sean igual a cero impedirán que el personaje se mueva en una u otra dirección.
Caso de prueba 3.12	**Resultado Esperado**
Insertar un valor con letras en uno de los campos.	El editor emitirá un mensaje de que el número introducido es incorrecto y no asignará dicho valor.
	Resultado Obtenido
	El editor emitirá un mensaje de que el número introducido es incorrecto y no asignará dicho valor.
Caso de prueba 3.13	**Resultado Esperado**
Insertar valores válidos en todos los campos.	El editor asignará dichos valores a la instancia del DSL.
	Resultado Obtenido
	El editor asignará dichos valores a la instancia del DSL.
Caso de prueba 3.14	**Resultado Esperado**
Intentar asignarle al personaje un tamaño menor que 35.	El editor sólo permitirá añadir valores para el tamaño del personaje principal comprendidos entre 35 y 70.
	Resultado Obtenido
	El editor lanza un mensaje de error y no deja guardar un personaje con esas dimensiones.

Caso de prueba 3.15	**Resultado Esperado**
Intentar asignarle al personaje un tamaño mayor que 70.	El editor sólo permitirá añadir valores para el tamaño del personaje principal comprendidos entre 35 y 70.
	Resultado Obtenido
	El editor lanza un mensaje de error y no deja guardar un personaje con esas dimensiones.

8.1.1.4 Configuración elementos del nivel

Configuración elementos del nivel Clases **TrapSetPropertiesPlatformVehicles.xaml, CheckpointSetPropertiesPlatformVehicles.xaml, TileSetPropertiesPlatformVehicles.xaml, TargetSetPropertiesPlatformVehicles.**	
Caso de prueba 4.1	**Resultado Esperado**
Elegir un recurso gráfico para un elemento que no sea .png.	El editor no permite seleccionar imágenes con formato distinto a .png. El explorador de carpetas para escoger el icono tiene un filtro, así que solamente mostrará imágenes con esta extensión.
	Resultado Obtenido
	El editor no permite seleccionar imágenes con formato distinto a .png.
Caso de prueba 4.2	**Resultado Esperado**
Definir la imagen, el número de frames y el tamaño de un elemento de nivel.	Se muestra la vista previa del elemento.
	Resultado Obtenido
	Se muestra la vista previa del elemento.

Caso de prueba 4.3	**Resultado Esperado**
Añadir un número de frames negativo.	El editor muestra el mensaje de error correspondiente y no permite guardar el sprite.
	Resultado Obtenido
	El editor muestra el mensaje de error "El número de frames no puede ser igual a cero o exceder el ancho de la imagen" y no permite guardar el sprite.
Caso de prueba 4.4	**Resultado Esperado**
Seleccionar un elemento de nivel de la biblioteca.	Los parámetros del elemento ya creado se cargan y se muestran en el editor con su vista previa incluida.
	Resultado Obtenido
	Los parámetros del elemento ya creado se cargan y se muestran en el editor con su vista previa incluida.
Caso de prueba 4.5	**Resultado Esperado**
Agregar un elemento nuevo a la biblioteca.	Si todos sus parámetros son correctos se agrega ese nuevo elemento a la biblioteca.
	Resultado Obtenido
	Si todos sus parámetros son correctos se agrega ese nuevo punto de control a la biblioteca

8.1.1.5 Configuración escenario

Configuración escenario Clase **SceneryPlatformVehicles**	
Caso de prueba 5.1	**Resultado Esperado**
Definir el punto de meta en una posición anterior al punto de salida.	Se asignan ambos valores a la instancia del DSL. Una vez se defina el videojuego, el personaje principal podrá ir de derecha a izquierda para completar el nivel.
	Resultado Obtenido
	Se asignan ambos valores a la instancia del DSL.
Caso de prueba 5.2	**Resultado Esperado**
Se sitúa un Tile en la posición X,Y.	Si la posición es válida dentro de los parámetros del nivel, se situará el Tile correspondiente en el escenario.
	Resultado Obtenido
	Si la posición es válida dentro de los parámetros del nivel, se situará el Tile correspondiente en el escenario.
Caso de prueba 5.3	**Resultado Esperado**
Definir un escenario compuesto por Tiles de tal manera que es imposible que el personaje principal llegue a la meta.	El editor asignará igualmente los recursos a la instancia del DSL. Es competencia del usuario diseñador definir niveles que puedan ser finalizados.
	Resultado Obtenido
	El editor asignará igualmente los recursos a la instancia del DSL. Es competencia del usuario diseñador definir niveles que puedan ser finalizados.

Caso de prueba 5.4	**Resultado Esperado**
Se pulsa el botón "Siguiente" sin haber definido la meta y el personaje principal	Es obligatorio que exista un personaje principal y una meta para que el nivel se guarde. Se mostrará el mensaje de error correspondiente.
	Resultado Obtenido
	Es obligatorio que exista un personaje principal y una meta para que el nivel se guarde. Aparecerán unos recuadros rojos mientras no haya meta ni personaje principal y se mostrará otro error al pulsar siguiente.
Caso de prueba 5.5	**Resultado Esperado**
No se escoge ninguna imagen para el fondo.	El fondo por defecto es negro.
	Resultado Obtenido
	El fondo por defecto es negro.
Caso de prueba 5.6	**Resultado Esperado**
Se pulsa el botón de "Siguiente"	Los parámetros elegidos se agregan a la instancia del DSL y los recursos gráficos se almacenan en la carpeta Drawable.
	Resultado Obtenido
	Los parámetros elegidos se agregan a la instancia del DSL y los recursos gráficos se almacenan en la carpeta Drawable.
Caso de prueba 5.7	**Resultado Esperado**
Situar un punto de control en un punto posterior a la meta.	El punto de control se agrega sin problemas. Es responsabilidad del usuario diseñador definir puntos de control que realmente se vayan a utilizar.
	Resultado Obtenido
	El punto de control se agrega sin problemas. Es responsabilidad del usuario diseñador definir puntos de control que realmente se vayan a utilizar.

Caso de prueba 5.8	**Resultado Esperado**
Situar una trampa en un punto posterior a la meta.	La trampa se agrega sin problemas. Es responsabilidad del usuario diseñador definir trampas que no sean activas en el desarrollo del videojuego.
	Resultado Obtenido
	La trampa se agrega sin problemas. Es responsabilidad del usuario diseñador definir trampas que no sean activas en el desarrollo del videojuego.
Caso de prueba 5.9	**Resultado Esperado**
Situar una trampa en una posición que impide al personaje llegar a la meta.	La trampa se agrega sin problemas. Es responsabilidad del usuario diseñador definir un nivel factible.
	Resultado Obtenido
	La trampa se agrega sin problemas. Es responsabilidad del usuario diseñador definir un nivel factible.
Caso de prueba 5.10	**Resultado Esperado**
No seleccionar número de vidas.	El selector ya tendrá un valor por defecto igual a tres.
	Resultado Obtenido
	El selector ya tendrá un valor por defecto igual a tres.
Caso de prueba 5.11	**Resultado Esperado**
Seleccionar un número de vidas inferior a uno.	Cuando se intente actualizar el número de vidas la aplicación dará un error y no lo permitirá.
	Resultado Obtenido
	Cuando se intente actualizar el número de vidas la aplicación dará el error “Número erróneo en el campo vidas. Inserte un número entre 1 y 5” y no asignará dicho valor.

Caso de prueba 5.12	**Resultado Esperado**
Desmarcar el *checkbox* de tiempo y no definir un tiempo máximo para finalizar el nivel.	Si no se define un valor de tiempo mayor que cero, aunque la casilla de "sin límite de tiempo" esté desmarcada, el nivel se ejecutará sin límite de tiempo.
	Resultado Obtenido
	Si no se define un valor de tiempo mayor que cero, aunque la casilla de "sin límite de tiempo" esté desmarcada, el nivel se ejecutará sin límite de tiempo.
Caso de prueba 5.13	**Resultado Esperado**
Desmarcar el *checkbox* de tiempo, definir un tiempo válido y escoger un número de vidas entre uno y cinco.	Se asignarán dichos parámetros al .xml del nivel y, posteriormente, el usuario jugador tendrá que llegar a la meta del nivel en el tiempo fijado y sin gastar las vidas definidas.
	Resultado Obtenido
	Se asignarán dichos parámetros al .xml del nivel.
Caso de prueba 5.14	**Resultado Esperado**
Pulsar el botón "Limpiar escenario"	Tras el mensaje de confirmación correspondiente, el escenario del nivel se borra.
	Resultado Obtenido
	Tras el mensaje de confirmación correspondiente (Se perderán todos los datos relativos a este nivel, ¿desea continuar?) , el escenario del nivel se borra.

8.2 Pruebas de Integración y del Sistema

8.2.1.1 Cargar/guardar videojuego

Paso 1. Guardar videojuego	
Caso de prueba 6.1	**Resultado Esperado**
Abrir el editor, realizar algunos cambios y navegar entre pantallas. Dejar alguna pantalla con errores.	En la carpeta donde se almacena el juego se crea un archivo .dat por cada pantalla que se configura y se pulsa sobre "Siguiente". Si ya existe se actualiza.
	Resultado Obtenido
	En la carpeta donde se almacena el juego se crea un archivo .dat por cada pantalla que se configura y se pulsa sobre "Siguiente".
Caso de prueba 6.2	**Resultado Esperado**
En alguna pantalla pulsar sobre la opción "Guardar" del menú.	Actualiza los archivos .dat existentes o crea los nuevos necesarios guardando los parámetros del videojuego en desarrollo.
	Resultado Obtenido
	Actualiza los archivos .dat existentes o crea los nuevos necesarios guardando los parámetros del videojuego en desarrollo.
Paso 2. Cargar videojuego previamente guardado	
Caso de prueba 6.3	**Resultado Esperado**
Cerrar la aplicación, volver a abrirla y cargar el juego guardado en el paso 1.	El videojuego se cargará correctamente recuperando todos los elementos que se habían salvado en su momento.
	Resultado Obtenido
	El videojuego se cargará correctamente recuperando todos los elementos que se habían salvado en su momento.

Caso de prueba 6.4	**Resultado Esperado**
Acceder a la pantalla guardada con errores.	El editor mostrará los errores que impiden la edición del videojuego.
	Resultado Obtenido
	El editor mostrará los errores que impiden la edición del videojuego.
Caso de prueba 6.5	**Resultado Esperado**
Pulsar el botón "anterior" en algún momento de la edición	En la pantalla anterior aparecerán los valores que introdujo el usuario al acceder a dicha pantalla.
	Resultado Obtenido
	En la pantalla anterior aparecerán los valores que introdujo el usuario al acceder a dicha pantalla.

8.2.1.2 Exportar videojuego

Exportar videojuego	
Caso de prueba 7.1	**Resultado Esperado**
Crear un videojuego y pulsar "siguiente" en la última pantalla.	Se muestra la pantalla de selección de ruta de despliegue.
	Resultado Obtenido
	Se muestra la pantalla de selección de ruta de despliegue.
Caso de prueba 7.2	**Resultado Esperado**
Se elige una ruta donde almacenar el videojuego, se selecciona la plataforma Android y se pulsa el botón de generar.	Tras el tiempo necesario, el proyecto Android con todos los elementos definidos aparecerá en la ruta seleccionada.
	Resultado Obtenido
	Tras el tiempo necesario, el proyecto Android con todos los elementos definidos aparecerá en la ruta seleccionada.

Caso de prueba 7.3	**Resultado Esperado**
Se importa el proyecto generado en el Eclipse.	El proyecto se importa sin dar errores.
	Resultado Obtenido
	El proyecto se importa sin dar errores. En primera instancia se obtenía un proyecto con *Warnings,* la mayoría de ellos por causa de *imports* inutilizados. Para no confundir al usuario no experto se ha decidido eliminar todos los *warnings* de la plantilla y por tanto, del videojuego final.
Caso de prueba 7.4	**Resultado Esperado**
Se compila el proyecto.	El proyecto se compila sin errores, se abre el emulador y se genera el .adk, archivo a instalar posteriormente en el dispositivo del usuario jugador.
	Resultado Obtenido
	El proyecto se compila sin errores, se abre el emulador y se genera el .adk, archivo a instalar posteriormente en el dispositivo del usuario jugador.
Caso de prueba 7.5	**Resultado Esperado**
Se instala el fichero .adk generado en Eclipse en el dispositivo.	El videojuego se instala correctamente en el dispositivo y muestra en primera instancia la pantalla principal.
	Resultado Obtenido
	El videojuego se instala correctamente en el dispositivo y muestra en primera instancia la pantalla principal. El dispositivo debe tener la opción “Permitir instalación de aplicaciones que no sean de Market” activada.

8.2.1.3 Jugar videojuego

Jugar videojuego	
Caso de prueba 8.1	**Resultado Esperado**
Se ejecuta el videojuego en el emulador del Eclipse	El videojuego se instala correctamente y funciona.
	Resultado Obtenido
	El videojuego se instala correctamente y funciona.
Caso de prueba 8.2	**Resultado Esperado**
Se ejecuta el videojuego en diferentes dispositivos Android con diferentes resoluciones	La lógica del videojuego se comporta exactamente igual en todas las resoluciones.
	Resultado Obtenido
	La lógica del videojuego se comporta exactamente igual en todas las resoluciones.
Caso de prueba 8.3	**Resultado Esperado**
Se selecciona un nivel	Tras el tiempo de espera requerido, el personaje principal aparecerá en el nivel seleccionado
	Resultado Obtenido
	Tras el tiempo de espera requerido, el personaje principal aparecerá en el nivel seleccionado
Caso de prueba 8.4	**Resultado Esperado**
Se procede a seleccionar un nivel sin límite de tiempo	En el selector de niveles no aparecerá ningún indicador del tiempo necesario para superar el nivel.
	Resultado Obtenido
	En el selector de niveles no aparecerá ningún indicador del tiempo necesario para superar el nivel.

Caso de prueba 8.5	**Resultado Esperado**
Una vez superado un nivel se vuelve a la pantalla de selección de nivel.	En la pantalla de selección de nivel pueden elegirse sólo aquellos niveles que se hayan superado.
	Resultado Obtenido
	En la pantalla de selección de nivel pueden elegirse sólo aquellos niveles que se hayan superado.
Caso de prueba 8.6	**Resultado Esperado**
Se llega a la meta del nivel cumpliendo el objetivo predefinido.	Se muestra un mensaje de fin de nivel con el tiempo invertido en completarlo.
	Resultado Obtenido
	Se muestra un mensaje de fin de nivel con el tiempo invertido en completarlo.
Caso de prueba 8.7	**Resultado Esperado**
El personaje principal pierde todas las vidas.	Se muestra una imagen tipo “Game Over” y se reinicia el nivel.
	Resultado Obtenido
	Se muestra una imagen tipo “Game Over” y se ofrece la posibilidad de “Reiniciar nivel” o “Volver a la pantalla de selección de nivel”.
Caso de prueba 8.8	**Resultado Esperado**
Se finaliza el último nivel.	Se muestra una imagen de juego finalizado y se vuelve al menú principal del juego.
	Resultado Obtenido
	Se muestra una imagen de juego finalizado y se vuelve al menú principal del juego.

Caso de prueba 8.9	**Resultado Esperado**
El personaje choca contra un ladrillo sólido.	El personaje principal debe de frenar aunque el botón de avance esté pulsado.
	Resultado Obtenido
	El personaje principal frena aunque el botón de avance esté pulsado.
Caso de prueba 8.10	**Resultado Esperado**
El personaje principal colisiona con una rampa ascendente	El personaje principal debe ascender por la rampa con la inclinación correspondiente.
	Resultado Obtenido
	El personaje principal debe ascender por la rampa con la inclinación correspondiente.
Caso de prueba 8.11	**Resultado Esperado**
El personaje principal colisiona con una rampa descendente	El personaje principal debe descender por la rampa con la inclinación correspondiente.
	Resultado Obtenido
	El personaje principal debe descender por la rampa con la inclinación correspondiente.
Caso de prueba 8.12	**Resultado Esperado**
El personaje principal colisiona contra una trampa.	El personaje principal pierde una vida.
	Resultado Obtenido
	El personaje principal pierde una vida y se reproduce (si se definió en el editor) el sonido de pérdida.

Caso de prueba 8.13	**Resultado Esperado**
El personaje principal cae por la parte inferior de la pantalla.	El personaje principal pierde una vida.
	Resultado Obtenido
	El personaje principal pierde una vida y se reproduce (si se definió en el editor) el sonido de pérdida. Se han tenido que efectuar algunos cambios en la plantilla durante la fase de pruebas para evitar una excepción *outOfBounds.*
Caso de prueba 8.14	**Resultado Esperado**
El personaje principal muere por alguna de las razones anteriores habiendo alcanzado algún *checkpoint* del nivel previamente.	El personaje principal pierde una vida y aparece en el último punto de control por el que haya pasado si sus vidas no han llegado a cero.
	Resultado Obtenido
	El personaje principal pierde una vida y aparece en el último punto de control por el que haya pasado si sus vidas no han llegado a cero.
Caso de prueba 8.15	**Resultado Esperado**
El personaje principal no supera el tiempo mínimo definido para completar el nivel	El marcador de tiempo debe de parpadear cuando el tiempo esté cerca de agotarse y una vez que lo haga el nivel se reinicia.
	Resultado Obtenido
	El marcador de tiempo debe de parpadear cuando el tiempo esté cerca de agotarse y una vez que lo haga el nivel se reinicia.
Caso de prueba 8.16	**Resultado Esperado**
Al personaje principal sólo le queda una vida y la pierde.	El nivel se reinicia.
	Resultado Obtenido
	El nivel se reinicia.

Caso de prueba 8.17	**Resultado Esperado**
Se pausa el juego	Aparece la pantalla de pausa y la lógica del juego deja de actualizarse.
	Resultado Obtenido
	Aparece la pantalla de pausa y la lógica del juego deja de actualizarse. Se ha detectado durante la fase de *testing* que si la pantalla de pausa tenía zonas transparentes, los botones de la interfaz (reiniciar nivel y moverse a izquierda y derecha) seguían recibiendo la pulsación. Se cambió la función *onTouchEvent* de la clase *GameView* para que esto no ocurriera.
Caso de prueba 8.18	**Resultado Esperado**
Se cambian los parámetros por defecto de los sonidos en la pantalla de opciones desactivando la música y los efectos.	Si los efectos y la música ambiental están desactivados, no se escuchará ningún tipo de sonido al comenzar cualquiera de los niveles.
	Resultado Obtenido
	Si los efectos y la música ambiental están desactivados, no se escuchará ningún tipo de sonido al comenzar cualquiera de los niveles.
Caso de prueba 8.19	**Resultado Esperado**
Desde la pantalla de derrota probar que los botones de “Repetir nivel” y “Seleccionar nivel” cumplen con la navegabilidad definida.	El botón de “Repetir nivel” carga el mismo nivel que se acaba de jugar. El botón de “Seleccionar nivel” lleva a la pantalla de selección de niveles.
	Resultado Obtenido
	El botón de “Repetir nivel” carga el mismo nivel que se acaba de jugar. El botón de “Seleccionar nivel” no funcionaba en el momento de realización del *test.* Se ha hecho el cambio oportuno en la clase *PlatformerDroidActivity* para obtener los resultados esperados.

Caso de prueba 8.20	**Resultado Esperado**
Desde la pantalla de victoria probar que los botones de "Siguiente nivel" y "Repetir nivel" cumplen con la navegabilidad definida.	El botón de "Siguiente nivel" carga el siguiente nivel si no es el último. El botón de "Repetir nivel" carga el mismo nivel que se acaba de jugar.
	Resultado Obtenido
	El botón de "Siguiente nivel" carga el siguiente nivel si no es el último. El botón de "Repetir nivel" carga el mismo nivel que se acaba de jugar

8.3 Pruebas de Usabilidad

A continuación se adjuntan los resultados de las pruebas de usabilidad realizadas a usuarios. Para obtener una mejor visión general de los mismos se han dividido en tablas de porcentaje dependiendo del perfil de usuario que realizó la prueba. Se recuerda que los tipos de usuarios definidos son los siguientes:

- **Tipo de usuario 1:** Estudiante de informática o usuario con estudios de ingeniero informático, con amplios conocimientos de programación y muy familiarizado con el manejo de ordenadores.
- **Tipo de usuario 2:** Usuario con estudios no relacionados con la informática y familiarizado con el manejo básico de ordenadores.

Las pruebas se han efectuado sobre un total de seis personas teniendo en cuenta que cada perfil de usuario definido en la sección de diseño contaba con tres representantes.

En primer lugar los usuarios cubrieron el cuestionario de preguntas de carácter general y a continuación realizaron las tareas definidas en la sección de diseño. Finalmente contestaron a las preguntas cortas de la aplicación teniendo en cuenta el desarrollo de las pruebas.

8.3.1 Resultados de las preguntas de carácter general

8.3.1.1 Tipo de usuario 1

Nº pregunta	Opción 1	Opción 2	Opción 3	Opción 4	Opción 5
1	100%	0%	0%	0%	--
2	100%	66%	100%	100%	0%
3	66%	0%	33%	--	--
4	33%	100%	66%	--	--
5	33%	0%	100%	100%	33%
6	100%	0%	0%	0%	--
7	66%	33%	0%	--	--
8	0%	100%	--	--	--
9	0%	0%	66%	33%	--
10	33%	66%	0%	0%	--

8.3.1.2 Tipo de usuario 2

Nº pregunta	Opción 1	Opción 2	Opción 3	Opción 4	Opción 5
1	100%	0%	0%	0%	--
2	33%	66%	100%	100%	0%
3	0%	33%	66%	--	--
4	66%	0%	33%	--	--
5	33%	0%	100%	100%	0%
6	66%	0%	0%	33%	--
7	0%	33%	66%	--	--

8	0%	100%	--	--	--
9	0%	0%	0%	100%	--
10	0%	100%	0%	0%	--

8.3.2 Resultados de las preguntas cortas sobre la aplicación

8.3.2.1 Perfil de usuario 1

Facilidad de uso				
Nº pregunta	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
1	66%	33%	0%	0%
2	33%	66%	0%	0%
3	100%	0%	0%	0%
4	0%	66%	33%	0%
5	0%	0%	66%	33%
6	33%	66%	0%	0%
7	66%	33%	0%	0%
8	66%	33%	0%	0%
9	0%	0%	100%	0%
10	66%	33%	0%	0%
11	33%	66%	0%	0%

Funcionalidad				
Nº pregunta	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
1	66%	33%	0%	0%
2	0%	0%	100%	0%
3	0%	0%	0%	100%
4	0%	0%	100%	0%
5	0%	0%	100%	0%
6	100%	0%	0%	0%
7	0%	0%	66%	33%
8	100%	0%	0%	0%
9	100%	0%	0%	0%
10	100%	0%	0%	0%
11	100%	0%	0%	0%

Calidad del interfaz				
Nº pregunta	**Muy adecuado**	**Adecuado**	**Poco adecuado**	**Nada adecuado**
1	100%	0%	0%	0%
2	100%	0%	0%	0%
3	100%	0%	0%	0%
4	66%	33%	0%	0%
5	100%	0%	0%	0%
6	66%	33%	0%	0%

Diseño de la interfaz			
Nº pregunta	**Sí**	**No**	**A veces**
1	66%	0%	33%
2	100%	0%	0%
3	100%	0%	0%
4	100%	0%	0%
5	0%	0%	100%
6	0%	100%	0%
7	100%	0%	0%
8	100%	0%	0%

8.3.2.2 Perfil de usuario 2

Facilidad de uso				
Nº pregunta	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
1	33%	66%	0%	0%
2	33%	33%	33%	0%
3	100%	0%	0%	0%
4	0%	100%	0%	0%
5	0%	66%	33%	0%
6	0%	33%	66%	0%
7	33%	66%	0%	0%
8	33%	33%	33%	0%
9	0%	0%	66%	33%
10	66%	33%	0%	0%

11	100%	0%	0%	0%

Funcionalidad				
Nº pregunta	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
1	33%	66%	0%	0%
2	0%	0%	66%	0%
3	0%	0%	0%	100%
4	0%	0%	33%	66%
5	0%	0%	100%	0%
6	100%	0%	0%	0%
7	0%	0%	33%	66%
8	100%	0%	0%	0%
9	100%	0%	0%	0%
10	100%	0%	0%	0%
11	100%	0%	0%	0%

Calidad del interfaz				
Nº pregunta	**Muy adecuado**	**Adecuado**	**Poco adecuado**	**Nada adecuado**
1	66%	33%	0%	0%
2	66%	33%	0%	0%
3	100%	0%	0%	0%
4	33%	66%	0%	0%
5	66%	33%	0%	0%
6	33%	66%	0%	0%

Diseño de la interfaz			
Nº pregunta	**Sí**	**No**	**A veces**
1	100%	0%	0%
2	100%	0%	0%
3	100%	0%	0%
4	100%	0%	0%
5	0%	33%	66%
6	0%	100%	0%
7	100%	0%	0%
8	100%	0%	0%

8.3.3 Análisis de los resultados, cambios efectuados y conclusiones

Todos los usuarios que efectuaron las pruebas pudieron completar las tareas definidas en la parte de diseño. No obstante, todos necesitaron acudir al manual de usuario para poder completar las partes más complejas del proceso que por unanimidad fueron: creación del personaje, creación de los *Tiles,* creación del nivel e importación y ejecución en Eclipse. En general la diferencia de perfiles o el hecho de si eran usuarios habituales de videojuegos no influyó en el proceso de edición del videojuego, únicamente en la tarea final de ejecutar el proyecto desde el Eclipse pues algunos usuarios ya habían trabajado con este entorno de desarrollo y no necesitaron ayuda en este paso.

El primer cambio efectuado en la interfaz del editor para mejorar la usabilidad fue cambiar el nombre de la tipología. En un principio se pensó en bautizarla como "Plataformas orientado a vehículos" pero los usuarios prefirieron el nombre de "Rampas" para entender la diferencia con el resto de tipologías y por ser éste más directo y sencillo.

Varios usuarios manifestaron que el proceso de seleccionar una imagen distinta a la de por defecto a la hora de definir los botones era poco intuitiva e implicaba hacer clic en la imagen y acto seguido mover el ratón para acceder al sistema de ficheros. Se añadió por tanto un segundo sistema mediante el cual se accede al sistema de ficheros simplemente haciendo clic derecho sobre la imagen del botón. Esta información también fue añadida a la parte correspondiente del manual de usuario para futuras consultas.

El proceso de creación de *Tiles* fue el más difícil de ejecutar por parte de los usuarios, sobre todo aquellos *Tiles* de 22º y 67º ya que deben ser definidos en dos partes. Por ello se ha decidido incluir una explicación más profunda y con más capturas de pantalla y ejemplos en el manual de usuario.

El editor estaba preparado para seleccionar una imagen por defecto de explicación en el primer nivel en el caso de que el usuario no definiera ninguna. Este hecho fue comentado por varios usuarios mientras probaban el videojuego puesto que en el mismo aparecía un elemento que ellos no habían definido. Para evitar la confusión se decidió no incluir ninguna imagen de explicación por defecto.

A nivel general el 83% de los usuarios aseguraron que el editor era fácil de utilizar y sólo el 17% restante consideró que esta característica se cumplía sólo en algunas ocasiones. Los resultados de la parte de calidad del interfaz de las encuestas muestran resultados muy positivos en cuanto a que todos los usuarios percibieron el diseño del editor como atractivo y usable. Otro factor que permite asegurar que se han cumplido los objetivos del proyecto es que todos los usuarios consideraron que el desarrollo de videojuegos se hace significativamente más fácil gracias a esta herramienta.

8.4 Pruebas de Rendimiento

Las pruebas de rendimiento del videojuego se han limitado a comprobar que la velocidad de refresco en los dispositivos especificados en la definición de pruebas de la sección de diseño del sistema ha sido la correcta (no inferior a 30 fps) y no se han detectado cuellos de botella ni partes del juego en los que el rendimiento se viera afectado.

Sólo se ha detectado en algunas ocasiones una velocidad de refresco baja en el emulador de Eclipse, lo cual era un efecto esperado debido a que el emulador tiene que convertir las instrucciones de la CPU de la arquitectura ARM en la que se basa Android a la de los equipos de desarrollo de tipo x86.

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

9.1.1 Instalación del editor

Para poder instalar el editor Gade4All es necesario tener instalado **.NET Framework 4.0** o superior. Una vez se haya instalado se deberá acceder a la carpeta "Ejecutables" del CD entregado. En dicha carpeta se encontrará un fichero llamado "Setup.msi". Si se hace clic en el mismo comenzará la instalación.

Ilustración 73. Icono para comenzar la instalación

La primera pantalla es meramente informativa, pulsando "Siguiente" se pasa a la pantalla donde se elige la ruta para instalar el editor. Lo más recomendable es hacerlo en "Archivos de programa" (*Program Files).*

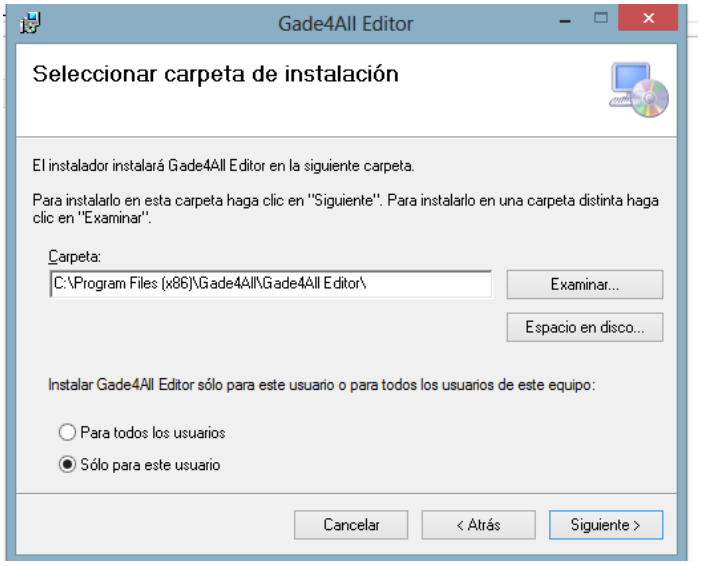


Ilustración 74. Pantalla de instalación de selección de carpeta

Una vez seleccionada la ruta se pulsa "Siguiente" y de nuevo, "Siguiente" para confirmar las opciones. Tras un tiempo determinado el programa dará un aviso de que ha terminado con la instalación (es posible que se requieran aceptar permisos de administrador durante el proceso).

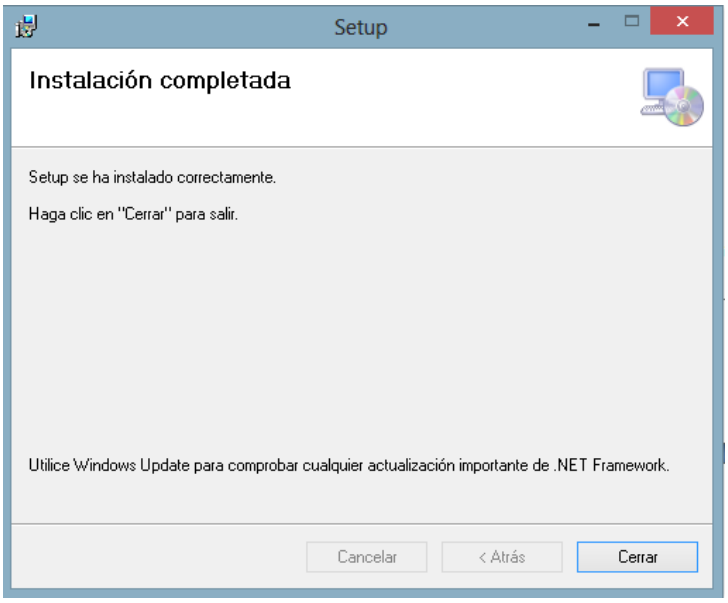


Ilustración 75. Pantalla de instalación completada

En este momento ya se podrá acceder al editor instalado en la ruta seleccionada en el proceso de instalación o en el escritorio.

9.1.2 Instalación de Eclipse

Para simplificar el proceso de contar con un Eclipse totalmente funcional en lo que se refiere a la funcionalidad relacionada con las Android Tools se recomienda descargar el fichero desde: http://developer.android.com/sdk/index.html que no requiere de ningún tipo de instalación adicional.

9.2 Manual de Ejecución

9.2.1 Ejecución del editor

Una vez finalizada la instalación del editor Gade4All se generará automáticamente un acceso directo en el escritorio. Basta hacer clic en él para abrir el editor.

Ilustración 76. Icono en el escritorio para abrir el editor Gade4All

El editor también puede ser ejecutado abriendo la carpeta donde se decidió instalar y haciendo clic en el archivo *Gade4all.exe*

9.2.2 Ejecución de Eclipse

Para ejecutar el Eclipse descargado en la sección anterior basta con acceder a la ruta donde se descargó y en la carpeta Eclipse hacer clic en el archivo *eclipse.exe*

9.3 Manual de Usuario

A continuación se detalla el proceso de creación de un videojuego basado en el control de vehículos a través de las diferentes pantallas del editor Gade4All tratando de dar explicaciones sencillas y concretas sobre los diferentes procesos de edición y su consiguiente resultado en el videojuego final.

9.3.1 Pantalla principal

En la primera pantalla que se visualiza al abrir el editor se pueden ver las diferentes tipologías de juegos a elegir. Para acceder a la tipología del presente proyecto el usuario deberá hacer clic en el icono de "Rampas"

Ilustración 77. Pantalla de selección de tipología del editor

Es importante destacar que si se desea recuperar el estado de edición de un juego previamente guardado sólo podrá hacerse desde esta pantalla, seleccionando la opción de "Cargar juego" desde la pestaña de "Archivo" situada en la parte superior izquierda de la pantalla.

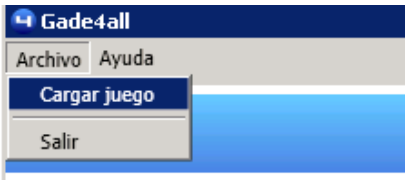


Ilustración 78. Captura de la opción de cargar el estado de edición de un juego desde el editor

Una vez seleccionada la tipología de "Rampas" aparecerá la siguiente ventana, en la cual se elegirá la ruta donde almacenar las carpetas que guarden la edición y los recursos audiovisuales que se vayan definiendo en el proceso.

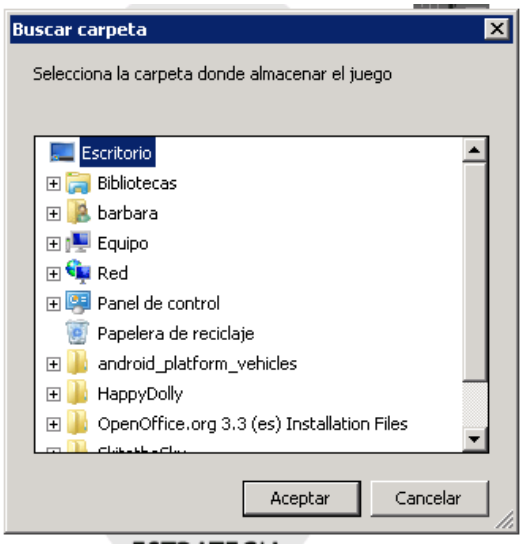


Ilustración 79. Captura de la ventana emergente donde almacenar el estado de edición

9.3.2 Configuración de propiedades globales

En esta pantalla se definen las propiedades globales del videojuego. Contiene las siguientes partes:

Ilustración 80. Captura de la pantalla de configuración de propiedades globales

- **Info básica:** se deberá definir de manera obligatoria el nombre del videojuego. Si no se define un icono (aquella imagen que aparecerá representando al juego en el menú del dispositivo Android) haciendo clic en "Elegir icono", se dejará el icono por defecto que se ve en la captura de pantalla.

- **Publicidad:** si se activa el *checkbox* y se introduce un ID de publicidad válido en la plataforma de Android en videojuego final mostrará en cada una de sus pantallas una zona de publicidad en la zona inferior de la pantalla.

- **Música de menús**: es opcional añadir música que se reproduzca en los menús del videojuego. Puede añadirse accediendo al sistema de ficheros mediante el icono de "Elegir música" y reproducirla con los botones de la parte inferior.

A partir de esta pantalla aparecerán en la parte superior los siguientes botones que permitirán moverse entre las pantallas de edición tanto para avanzar como para retroceder sobre las que ya se han editado.

Ilustración 81. Captura de los botones de navegación del editor

Asimismo, en el momento en el que se avance o se retroceda hacia otra pantalla, se guardará automáticamente el estado de la pantalla en edición. Si se desea cerrar la aplicación y guardar el estado de la última pantalla también se puede hacer en Archivo > Guardar.

Ilustración 82. Cómo guardar el estado de edición de manera manual

En la parte superior aparecerán los nombres de las pantallas de edición. Haciendo clic en una pestaña se puede acceder directamente a dicha pantalla

Ilustración 83. Captura de la barra de navegación superior del editor

9.3.3 Pantalla principal

La pantalla principal es la primera que aparece cuando se ejecuta el videojuego y se configura en esta pantalla del editor.

Sus partes personalizables son: el fondo, y los botones de "Jugar" (en el juego llevará a la pantalla de selección de nivel), "Opciones" (en el juego llevará a la pantalla de opciones) y "Salir" (saldrá del juego).

Ilustración 84. Captura de la pantalla de configuración de la pantalla principal

Para cambiar alguno de los botones por defecto o el fondo basta con hacer clic en uno de ellos (aparecerá un recuadro verde a su alrededor como en la captura de pantalla superior) y a continuación pulsar el botón "Elegir" situado en la parte izquierda de la pantalla. Se abrirá una ventana que permitirá la elección de un recurso gráfico para dicho elemento. Otra forma de cambiar la imagen de manera más directa pasa por hacer clic derecho encima del fondo o del botón. Sólo se permitirá seleccionar imágenes cuyo formato sea .png.

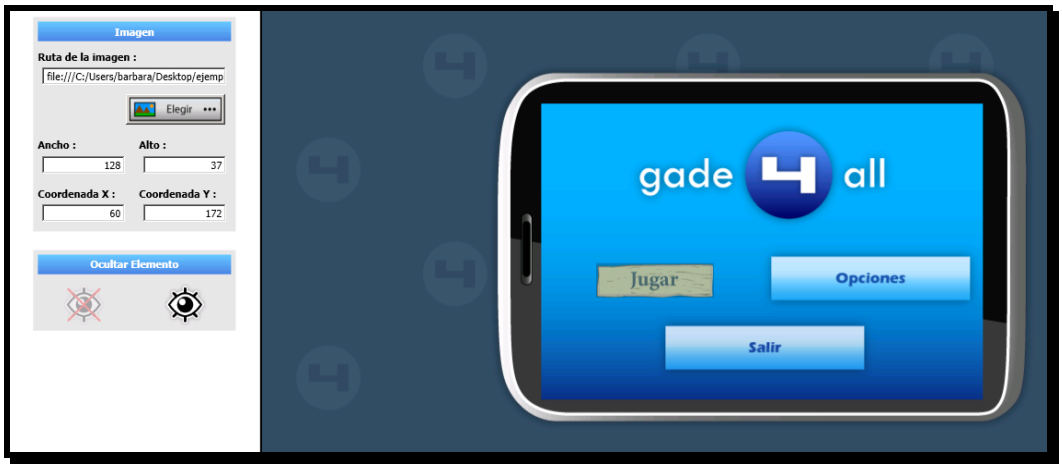


Ilustración 85. Ejemplo de selección de un botón personalizado "Jugar"

Como se puede comprobar, el ancho y alto se han actualizado automáticamente de acuerdo al tamaño de la imagen seleccionada. Estos valores, además de la posición X e Y pueden cambiarse manualmente de manera que los botones cambiarán para que el usuario pueda percibir los cambios. Las coordenadas X e Y de los botones también pueden modificarse mediante selección y arrastre sobre el escenario o con las teclas de arriba, abajo, izquierda y derecha del teclado para una mayor precisión.

Si se desea que alguno de los botones no sea visible cuando el videojuego final se ejecute se debe seleccionar el elemento haciendo clic en él y a continuación pulsar en el ojo con una cruz en la sección de "Ocultar elemento". El elemento se hará más transparente para indicar que en el videojuego final no aparecerá.

Ilustración 86. Ejemplo de ocultar un elemento, en este caso el botón "Salir"

Pulsando en el ojo de la derecha, el botón sería visible de nuevo.

9.3.4 Pantalla de opciones

Ilustración 87. Captura de la pantalla de opciones

Como puede comprobarse, la edición de esta pantalla es completamente análoga a la pantalla anterior.

Cabe destacar que en el videojuego final aparecerán únicamente los botones de "Efectos activados" y "Música activada" si no se oculta ningún elemento. Si se pulsa cualquiera de esos botones aparecerá su botón homólogo de "desactivado". Es decir, aunque en esta pantalla se definan los cuatro botones, en el videojuego sólo se mostrarán dos simultáneamente.

9.3.5 Pantalla de selección de nivel

Esta pantalla es a la que se accede cuando se pulsa el botón "Jugar" en la pantalla principal del juego. En ella puede elegirse el nivel a jugar siempre y cuando se haya superado con anterioridad en el dispositivo.

Ilustración 88. Captura de la pantalla de selección de nivel

La forma de editar los botones y recursos gráficos de esta pantalla es análoga a las pantallas anteriores aunque en este caso tenemos una sección más en la que es posible editar el color de texto del marcador de tiempo. Para cambiar el color del tiempo se debe seleccionar su recuadro correspondiente y a continuación hacer clic en la lista desplegable de colores y seleccionar el más adecuado. Por ejemplo, si se selecciona el color rojo (Red) la pantalla por defecto quedaría como sigue:

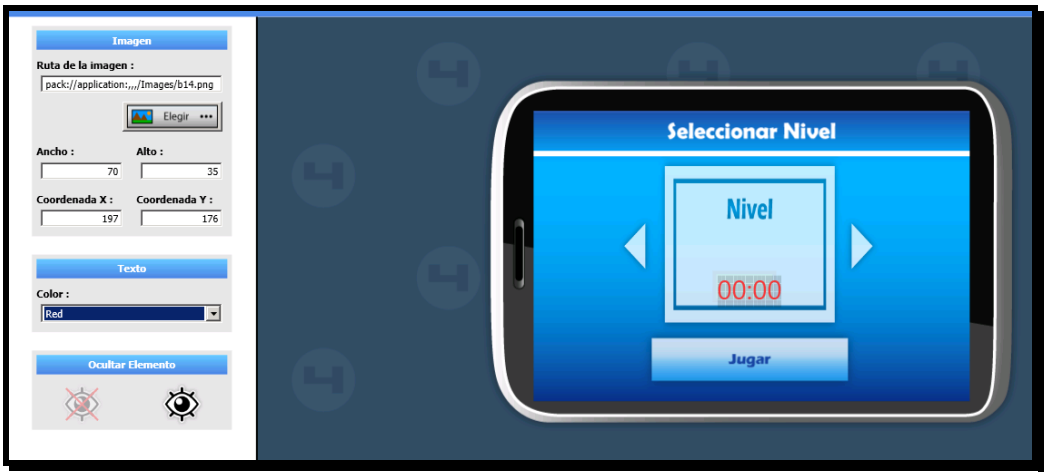


Ilustración 89. Ejemplo de cambio de color del marcador de tiempo

Es importante saber que en el caso de que el nivel no tenga límite de tiempo para ser completado (aspecto que se definirá en pantallas posteriores), el marcador de tiempo no aparecerá para ese nivel.

Además del color del marcador de tiempo, en esta pantalla también pueden modificarse el botón de "Jugar", el fondo y los botones de derecha e izquierda para avanzar entre niveles.

La imagen representativa del nivel también puede definirse pero no aparecerá en ningún momento en el juego ya que la imagen miniatura de cada nivel se definirá en pantallas posteriores. No obstante, es útil situarla en la pre visualización en este punto pues de esta manera se guarda el tamaño predefinido que van a tener todas las imágenes miniatura y se conoce la posición del marcador de tiempo para que encajen bien en la misma.

Ilustración 90. Ejemplo de pantalla de selección de nivel con imagen miniatura

9.3.6 Pantalla de juego

En esta pantalla se definen todos los elementos de interacción que se mantienen fijos durante el transcurso normal del videojuego.

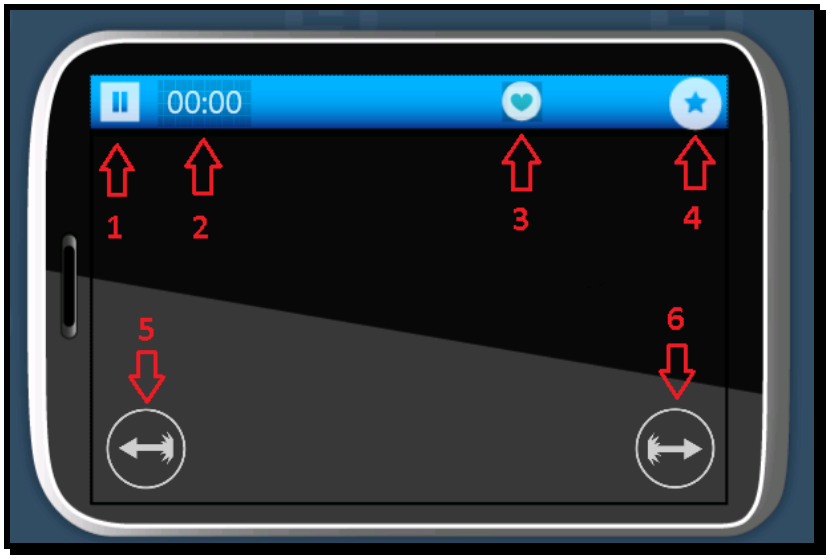


Ilustración 91. Captura de la pantalla del juego con indicaciones

(1) El primer elemento se trata del botón de pausa. Si se pulsa durante el transcurso del juego, la lógica del mismo deja de actualizarse y se muestra la pantalla de pausa que se definirá más adelante.

(2) Marcador de tiempo. Muestra el tiempo transcurrido desde el comienzo del nivel y parpadea cuando queda poco tiempo para el tiempo límite del nivel si este valor está definido. Al igual que en la pantalla anterior puede modificarse su color.

(3) Icono de vidas. Aunque el usuario sólo deba definir una única imagen, en el juego se repetirá dicha imagen tantas veces como vidas sea hayan definido en el nivel. A medida que el personaje pierda vidas en el desarrollo del juego, el número de imágenes repetidas disminuirá. Hay que tener en cuenta el número de vidas del nivel para dejar el suficiente espacio entre el icono de vidas que se repetirá y el resto de iconos de esta pantalla.

(4) Botón de reinicio. En algunas ocasiones el personaje principal puede quedarse "atrapado" en un punto del nivel y no ser capaz de llegar a la meta. Pulsando este botón, el personaje pierde una vida y aparece en el último punto de control visitado o en el punto de salida.

(5) Botón de retroceso. Pulsando este botón el personaje va hacia la izquierda.

(6) Botón de avance. Pulsando este botón el personaje va hacia la derecha.

9.3.7 Pantalla de pausa

Es el menú que aparece cuando se pulsa el botón de pausa en la pantalla de juego. Está compuesta por los botones de "Reanudar" y "Salir" y el fondo y se edita de manera análoga a las anteriores.

9.3.8 Pantalla de nivel superado

Es la pantalla que se muestra cuando se llega a la meta del nivel. Está compuesta por los botones de "Repetir nivel" y "Siguiente nivel", el fondo y el marcador donde se muestra el tiempo que ha llevado completar el nivel. Se edita de manera análoga a las anteriores.

9.3.9 Pantalla de perder nivel

Es la pantalla que aparece cuando el personaje pierde todas las vidas o se acaba el tiempo predefinido para superar el nivel. Está compuesta por el fondo y los botones de "Repetir nivel" y "Seleccionar nivel" que lleva a dicha pantalla. Se edita de manera análoga a las anteriores.

9.3.10 Pantalla de fin de juego

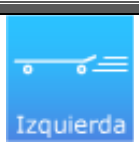
Es la pantalla que se muestra cuando se han superado todos los niveles. Está compuesta por el fondo y el botón de "Volver a menú" que una vez pulsado muestra de nuevo la pantalla principal. Se edita de manera análoga a las anteriores.

9.3.11 Configuración del personaje principal

En esta pantalla se configurarán los personajes principales, es decir, aquellos elementos que el usuario jugador podrá controlar.

En la parte de edición (parte central de la pantalla) pueden observarse las diferentes animaciones del personaje que deben ser configuradas. Su función es la siguiente:

Icono	Nombre	Explicación
	Animación por defecto izquierda	Es la animación que se muestra cuando el personaje está parado y el último botón pulsado fue el de dirección izquierda.
	Animación por defecto derecha	Es la animación que se muestra cuando el personaje está parado y el último botón pulsado fue el de dirección derecha.

	Animación izquierda	Es la animación que se muestra cuando el personaje se está moviendo hacia la izquierda.
	Animación derecha	Es la animación que se muestra cuando el personaje se está moviendo hacia la derecha.
	Animación subir rampas de 22º hacia la izquierda	Es la animación que se muestra cuando el personaje se mueve a la izquierda y se encuentra con una rampa ascendente de 22º de pendiente.
	Animación subir rampas de 22º hacia la derecha	Es la animación que se muestra cuando el personaje se mueve a la derecha y se encuentra con una rampa ascendente de 22º de pendiente.
	Animación bajar rampas de 22º hacia la izquierda	Es la animación que se muestra cuando el personaje se mueve a la izquierda y se encuentra con una rampa descendente de 22º de pendiente.
	Animación bajar rampas de 22º hacia la derecha	Es la animación que se muestra cuando el personaje se mueve a la derecha y se encuentra con una rampa descendente de 22º de pendiente.
	Animación subir rampas de 45º hacia la izquierda	Es la animación que se muestra cuando el personaje se mueve a la izquierda y se encuentra con una rampa ascendente de 45º de pendiente.
	Animación subir rampas de 45º hacia la derecha	Es la animación que se muestra cuando el personaje se mueve a la derecha y se encuentra con una rampa ascendente de 45º de pendiente.
	Animación bajar rampas de 45º hacia la izquierda	Es la animación que se muestra cuando el personaje se mueve a la izquierda y se encuentra con una rampa descendente de 45º de pendiente.
	Animación bajar rampas de 45º hacia la derecha	Es la animación que se muestra cuando el personaje se mueve a la derecha y se encuentra con una rampa descendente de 45º de pendiente.

	Animación subir rampas de 67º hacia la izquierda	Es la animación que se muestra cuando el personaje se mueve a la izquierda y se encuentra con una rampa ascendente de 67º de pendiente.
	Animación subir rampas de 67º hacia la derecha	Es la animación que se muestra cuando el personaje se mueve a la derecha y se encuentra con una rampa ascendente de 67º de pendiente.
	Animación bajar rampas de 67º hacia la izquierda	Es la animación que se muestra cuando el personaje se mueve a la izquierda y se encuentra con una rampa descendente de 67º de pendiente.
	Animación bajar rampas de 67º hacia la derecha	Es la animación que se muestra cuando el personaje se mueve a la derecha y se encuentra con una rampa descendente de 67º de pendiente.

Por ejemplo, si se quisiera establecer la animación de subir rampas de 22º hacia la izquierda, en primer lugar se seleccionaría el icono correspondiente y a continuación se haría clic en la zona gris ("Hacer clic aquí para añadir imágenes") para acceder al sistema de ficheros y elegir la imagen correspondiente. Cabe destacar que si no se define algunas de las animaciones (no es obligatorio hacerlo), a la animación no definida se le asignará por defecto la primera, es decir, la "animación por defecto izquierda" que sí es obligatoria.

Ilustración 92. Ejemplo de selección de animación para el personaje principal

En esta zona también pueden elegirse el número de *frames* y si se desea repetir la animación. Esta configuración tiene sentido cuando queremos que el personaje sea una animación compuesta por varias imágenes (en el ejemplo anterior no tendría sentido pues el personaje está formado por una única imagen)

La metodología para generar un personaje animado sería situar en una misma imagen todos los "fotogramas" que componen la animación.

Ilustración 93. Ejemplo de imagen representativa de un personaje animado

En el ejemplo el número de *frames* sería 4. Si se selecciona "Repetir animación" la imagen se repetirá una y otra vez a lo largo del desarrollo del videojuego. En caso contrario sólo se ejecutaría una vez y el personaje se quedaría con la última imagen de la animación. Este proceso debería repetirse para cada uno de los tipos de animación vistos anteriormente. En la zona de "Vista previa" puede observarse el resultado final que se verá en el juego.

Ilustración 94. Ejemplo de animación definiendo el número de frames

En el cuadro de propiedades aparecen una serie de características propias del personaje que conviene analizar para conocer sus repercusiones en el juego.

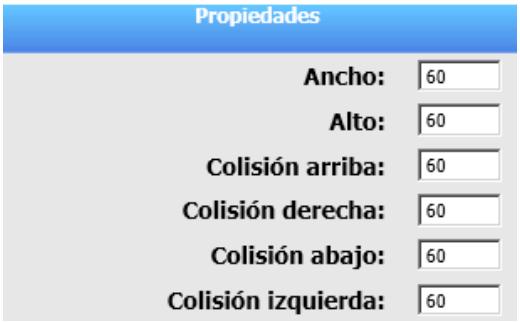


Ilustración 95. Captura de las propiedades de tamaño y colisión del personaje

En primer lugar aparecen las propiedades de "Ancho" y "Alto" del personaje. Estos valores se cargan cuando se selecciona una imagen en alguna de las animaciones comentadas anteriormente pero pueden ser modificadas a posteriori teniendo en cuenta la resolución de la imagen original para que ésta no aparezca deformada.

El editor no permite definir personajes cuyo ancho o alto sea menor de 35 píxeles o mayor de 70. Esta restricción se ha impuesto para impedir que se defina un personaje muy grande o muy pequeño que no interactúe de manera correcta con la lógica del juego.

Las siguientes propiedades se utilizan para definir el área de colisión del personaje y así calcular cuando colisiona con los diferentes elementos del nivel. En un caso perfecto el área de colisión del personaje debería coincidir con las dimensiones del modelo/imagen, pero esto no siempre es así, porque puede que la imagen tenga transparencias y no interesa que si algo toca una esquina transparente de la imagen el juego lo considere una colisión. Estas cuatro mediciones consiguen "generar un rectángulo" a partir del punto medio del personaje.

Ilustración 96. Imagen explicativa de la definición de la zona de colisión

Las siguientes propiedades definen la física que gobernará en el juego final.

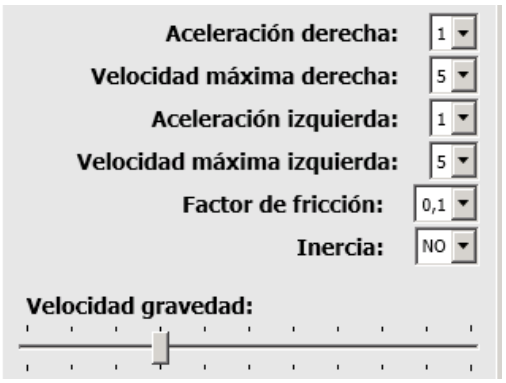


Ilustración 97. Captura de los valores por defecto para la física del personaje

- **Aceleración derecha:** establece el grado de avance hacia la derecha del personaje. En el ejemplo, cada vez que el usuario pulse el botón de avance a la derecha el personaje aumentará en una unidad su aceleración en ese sentido.
- **Velocidad máxima derecha:** establece el máximo valor que puede tomar la aceleración hacia la derecha. En el ejemplo, el personaje acelerará paulatinamente hasta una velocidad de cinco unidades.
- **Aceleración izquierda:** análogo a la aceleración derecha pero en el otro sentido.

- **Velocidad máxima izquierda:** análogo a la velocidad máxima izquierda pero en el otro sentido.

- **Factor de fricción:** si este valor se pusiera a cero el personaje no pararía una vez se inicie su movimiento. Por tanto es el valor que "frena" al personaje principal cuando éste se mueve y cuánto mayor sea este valor, menos fluido será el movimiento del personaje.

- **Inercia**: activando este valor el personaje principal descendería si se sitúa encima de una rampa simulando lo que sucede en la realidad para vehículos con ruedas.

- **Gravedad:** es el factor que determina la velocidad hacia abajo del personaje cuando éste se encuentra en el aire. Cuanto mayor sea este valor (indicador más a la derecha) más rápidamente descenderá el personaje.

  Las últimas propiedades que pueden personalizarse están relacionadas con los sonidos que se reproducirán ante determinados eventos del personaje. Para acceder al sistema de ficheros y elegir un archivo de sonido (en formato .mp3) se debe hacer clic en el botón +. Si se quiere borrar un sonido una vez se haya seleccionado, se pulsaría sobre el botón de la papelera. La definición de sonidos no es obligatoria.

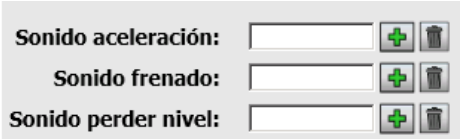


Ilustración 98. Captura de los sonidos del personaje

- **Sonido aceleración:** este sonido se reproducirá cada vez que el usuario pulse el botón de avance hacia la derecha. Se recomienda, por tanto, escoger archivos de corta duración.

- **Sonido frenado:** este sonido se reproducirá cada vez que el usuario pulse el botón de avance hacia la izquierda. Se recomienda, por tanto, escoger archivos de corta duración.

- **Sonido perder nivel:** es el sonido que se reproducirá cuando el personaje colisione contra una trampa o se le acabe el tiempo predefinido para acabar el nivel.

  Por último existen cuatro botones en la parte inferior de la pantalla que se encargan de gestionar el almacenamiento de los personajes.

Ilustración 99. Botones que gestionan la persistencia de los datos de los personajes

- **Crear nuevo elemento:** aunque en cada nivel sólo pueda aparecer un personaje principal, es posible definir varios personajes para variarlos entre niveles. Haciendo clic en este botón se crea un nuevo elemento vacío.

- **Agregar a biblioteca:** este botón sólo estará activo mientras se esté modificando un personaje que no se ha guardado previamente en la biblioteca. Haciendo clic en dicho botón se guardarán los datos del personaje en el ordenador. Para asegurarse de que al usuario no se le olvida este paso, el editor preguntará de manera automática al pasar a otra pantalla si desea guardar el estado del personaje en edición si detecta que no se ha agregado a la biblioteca previamente.

- **Actualizar información:** si se ha guardado un personaje a la biblioteca es posible que se deseen cambiar algunos de sus parámetros y que estos cambios sean permanentes, efecto que se consigue haciendo clic en este botón. No se podrán cambiar las propiedades de un personaje principal que ya esté en el escenario de un nivel (se verá posteriormente cómo desvincular un personaje del escenario para poder efectuar cambios sobre el mismo)

- **Eliminar elemento:** borra de manera permanente el personaje y todas sus propiedades. Al igual que en el caso anterior, no se podrán eliminar personajes que ya estén en el escenario de un nivel.

9.3.12 Configuración trampas

Las tramas son todos aquellos elementos del nivel con los cuales si el personaje principal colisiona, éste perderá una vida. Como se puede comprobar, la configuración de trampas es análoga a la configuración de personajes con la salvedad de que las trampas tienen muchas menos propiedades (sólo ancho y alto) y una única animación que será la que se muestre en el videojuego.

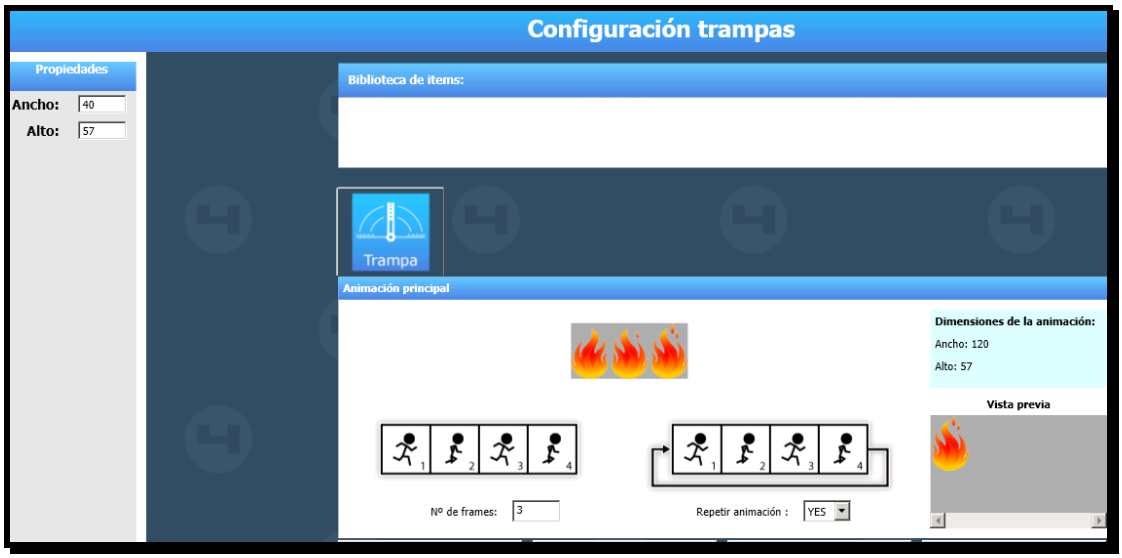


Ilustración 100. Ejemplo de definición de un elemento trampa

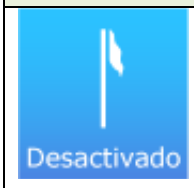
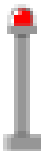
9.3.13 Configuración puntos de control

Los puntos de control o *checkpoints* son aquellos elementos que "recuerdan" la posición del personaje en el caso de que éste pierda una vida. Es decir, si un personaje activa un punto de control en la posición X del nivel y luego pierde una vida, reaparecerá en dicha posición.

Su configuración es muy similar a la de las trampas con la diferencia de que además del ancho y el alto también puede definirse el sonido que se reproducirá cuando el personaje principal active dicho punto de control.

Ilustración 101. Ejemplo de configuración de un punto de control

Asimismo, se deben definir dos animaciones para cada punto de control:

Icono	Explicación	Ejemplo
	Es la animación que se muestra por defecto en el punto de control antes de que el personaje lo haya alcanzado.	
	Es la animación que se muestra una vez que el personaje haya colisionado con el punto de control para indicar que está activo.	

9.3.14 Configuración Tiles

Los *Tiles* o elementos de escenario son aquellos elementos que componen la estructura del nivel. Son los "rectángulos" que establecerán los caminos por los que podrá discurrir el personaje principal.

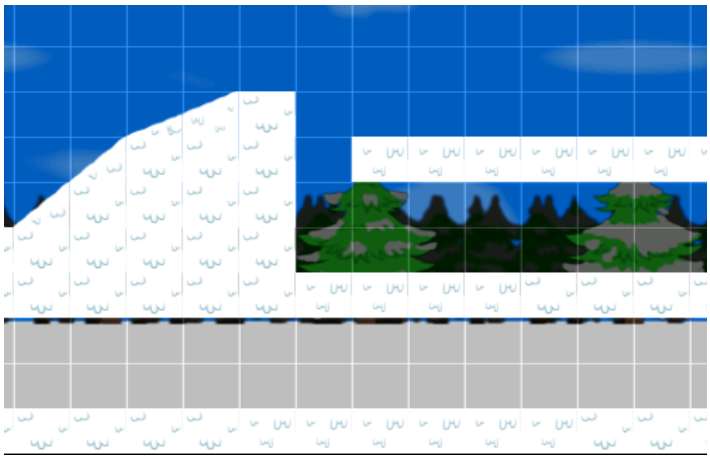


Ilustración 102. Ejemplo de un escenario compuesto a base de Tiles

La definición de estos "ladrillos" del nivel sigue el mismo esquema que los elementos vistos anteriormente. Conviene destacar, no obstante, que los *Tiles* en el videojuego final tendrán unas dimensiones fijas de 40x32 por lo que se recomienda en este punto crear los elementos con este tamaño para evitar que la imagen final se deforme respecto a la original.

El punto crítico de la definición de los elementos de escenario es entender qué tipo de imagen corresponde con qué tipo de *Tile.* Si la definición no se hace de manera correcta el videojuego final distará mucho de presentar un aspecto realista. Existen, por tanto, varios tipos de *Tile*:

Nombre	Explicación	Ejemplo
Sólido	Es el *Tile* más básico y está compuesto por un rectángulo. Suelen componer el suelo de los niveles o paredes. El personaje no puede atravesarlos.	
No sólido (pasable)	Sirven de adorno para los niveles. El personaje principal puede atravesarlos completamente.	
Rampa 45º arriba	Cuando el personaje colisiona con este tipo de *Tiles* asciende (si va hacia la derecha) o desciende (si va hacia la izquierda) por la diagonal del rectángulo definido por la imagen.	
Rampa 45º abajo	Cuando el personaje colisiona con este tipo de *Tiles* desciende (si va hacia la derecha) o asciende (si va hacia la izquierda) por la diagonal del rectángulo definido por la imagen.	

Rampa 22º arriba 1	Para definir una rampa de 22º ascendente es necesario hacerlo en dos *Tiles* separados. El primero debe ser del presente tipo y una vez el personaje colisione con él, éste ascenderá hasta la mitad de la altura del *Tile.*	
Rampa 22º arriba 2	Para definir una rampa de 22º ascendente es necesario hacerlo en dos *Tiles* separados. El segundo debe ser del presente tipo y una vez el personaje colisione con él, éste ascenderá desde hasta la mitad de la altura del *Tile* hasta el punto más alto.	
Rampa 22º abajo 1	Para definir una rampa de 22º descendente es necesario hacerlo en dos *Tiles* separados. El primero debe ser del presente tipo y una vez el personaje colisione con él, éste descenderá desde el punto superior hasta la mitad de la altura del *Tile.*	
Rampa 22º abajo 2	Para definir una rampa de 22º descendente es necesario hacerlo en dos *Tiles* separados. El segundo debe ser del presente tipo y una vez el personaje colisione con él, éste descenderá desde el punto medio hasta el punto más bajo del *Tile.*	
Rampa 67º arriba 1	Para definir una rampa de 67º ascendente es necesario hacerlo en dos *Tiles* separados. El primero debe ser del presente tipo y una vez el personaje colisione con él, éste ascenderá hasta el punto más alto del *Tile* (eje Y) pero en su punto medio (eje X)	

Rampa 67º arriba 2	Para definir una rampa de 67º ascendente es necesario hacerlo en dos *Tiles* separados. El segundo debe ser del presente tipo y una vez el personaje colisione con él, éste ascenderá hasta el punto más alto del *Tile* (eje Y) pero desde su punto medio (eje X)	
Rampa 67º abajo 1	Para definir una rampa de 67º descendente es necesario hacerlo en dos *Tiles* separados. El primero debe ser del presente tipo y una vez el personaje colisione con él, éste descenderá hasta el punto más bajo del *Tile* (eje Y) pero en su punto medio (eje X)	
Rampa 67º abajo 2	Para definir una rampa de 67º descendente es necesario hacerlo en dos *Tiles* separados. El segundo debe ser del presente tipo y una vez el personaje colisione con él, éste descenderá hasta el punto más bajo del *Tile* (eje Y) pero desde su punto medio (eje X)	

Las rampas de 22º y 67º son los que más dificultad entrañan. Para facilitar el entendimiento de su funcionamiento a continuación se adjunta un ejemplo de cómo encajarían estos tipos de *Tile* en un escenario final.

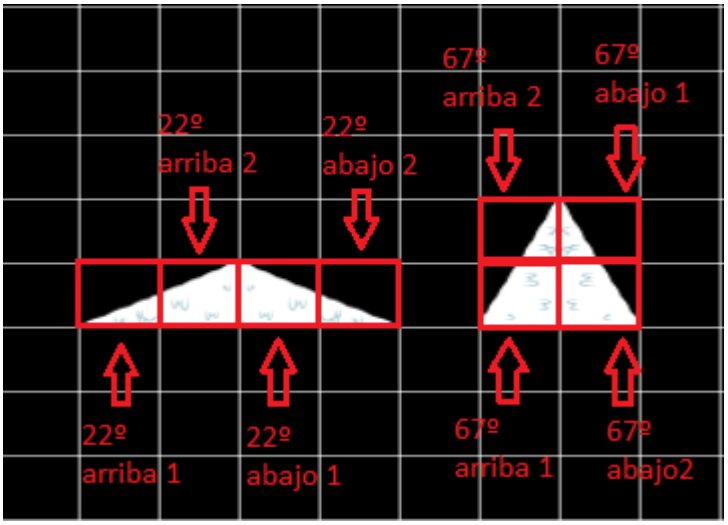


Ilustración 103. Forma de situar Tiles de 22º y 67º en el escenario

Como puede comprobarse en el ejemplo, es necesario "juntar" dos *Tiles* de 22º para subir o bajar un *Tile* sólido completo pero debe hacerse manteniendo el orden (Arriba 1, arriba 2 y abajo 1, abajo2). Las rampas de 67º se basan en el principio análogo de utilizar dos para conseguir un desplazamiento de "esquina a esquina" cuando el personaje colisione con ellos.

Las disposiciones propuestas en la imagen de arriba no son únicas, existiendo una enorme variedad que supondrá el inmenso número de diferentes escenarios que se podrán crear para cada nivel.

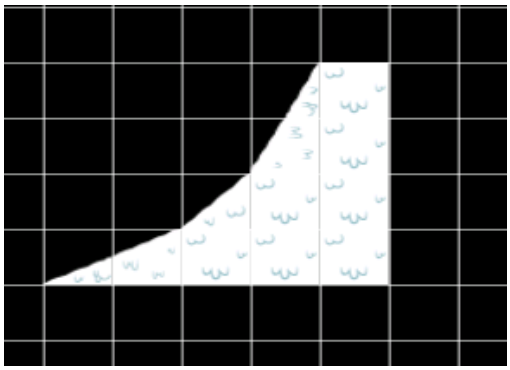


Ilustración 104. Ejemplo de rampa creada a partir de dos rampas de 22º, una de 45º y dos de 67º además de varios Tiles sólidos

9.3.15 Configuración meta

La meta es elemento obligatorio de todo nivel y representa el punto final al que tiene que llegar el personaje principal para finalizar dicho nivel.

Su configuración sigue el esquema de las anteriores con la diferencia de que no es posible definir una animación para la meta, solamente una imagen estática.

Ilustración 105. Ejemplo de configuración del elemento meta

9.3.16 Escenario

Esta es la pantalla donde se situarán en el escenario todos los elementos definidos en las pantallas anteriores. Es la fase donde realmente se van a definir los niveles que compondrán el videojuego final.

La parte más intuitiva supone situar elementos en el escenario. En la parte izquierda del editor aparecerán todos los elementos definidos hasta el momento. Para situarlos en el escenario basta con hacer clic en el elemento en cuestión y a continuación hacer clic en un punto del escenario y el elemento se situará sin problemas. Si nos hemos equivocado a la hora de situar un personaje, un *Tile,* una trampa, etc., la acción puede deshacerse haciendo clic derecho sobre el elemento en el escenario y éste desaparecerá.

Es obligatorio que el nivel tenga un personaje principal y una meta, y no podrá tener más de uno de estos elementos.

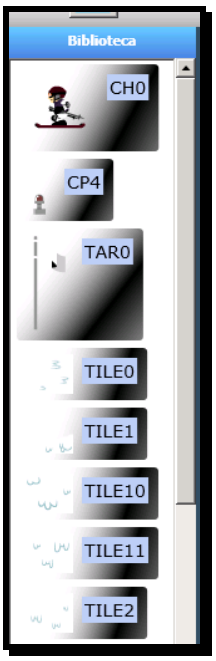


Ilustración 106. Ejemplo de elementos definidos en pantallas anteriores

En la parte de la derecha se encuentran otros botones, cuya funcionalidad se especificará a continuación.

- **Propiedades del nivel.** La ventana que aparecerá si se hace clic en este botón permitirá definir la condición de fin de nivel, incluyendo el tiempo del cual dispondrá el jugador para ir desde el punto inicial a la meta y el número de vidas que tendrá para ello. Si se marca el *checkbox* "Sin límite de tiempo" el nivel del juego final podrá finalizarse en cualquier lapso de tiempo, siempre y cuando el personaje llegue a la meta. El número de vidas del personaje podrá ser entre una y cinco (el número escogido en este punto serán las veces que se repita el icono de vidas definido en la pantalla de juego).

La propiedad de FPS es el número de *frames* por segundo que se mostrarán en el videojuego final. Conviene dejar este valor por defecto pues no todos los dispositivos pueden refrescar la lógica del juego más rápidamente.

En la parte de sonidos también puede definirse la música de fondo del nivel y la música que se reproducirá cuando el jugador supere dicho nivel.

Para guardar los cambios se debe hacer clic en el botón "Aceptar", para cancelarlos en el botón "Cancelar".

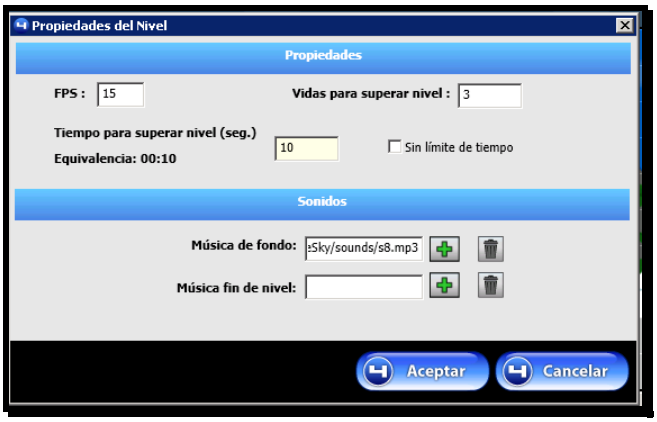


Ilustración 107. Pantalla de propiedades del nivel

- **Imagen de explicación.** Haciendo clic en este botón se abre una ventana donde podemos definir dos imágenes.

  La imagen de descripción del nivel es aquella que se mostrará justo antes de ejecutar la lógica del juego de dicho nivel. Es útil, por ejemplo, para explicar cómo funciona la lógica del juego o dejar claros los objetivos del nivel de cara a usuarios que se encuentran con el juego por primera vez.

  La imagen miniatura del nivel es la que se mostrará en la pantalla de selección de nivel. Tal y como se comentó en la subsección correspondiente a dicha pantalla en la presente guía, conviene definir todas las miniaturas del mismo tamaño que se estipuló en dicha pantalla. Es útil que la imagen miniatura contenga el número del nivel para mejorar la usabilidad del juego.

Ilustración 108. Pantalla de propiedades de descripción del nivel

- **Propiedades de los fondos.** Haciendo clic en dicho botón se puede definir el fondo del nivel. Existen dos maneras de hacerlo:

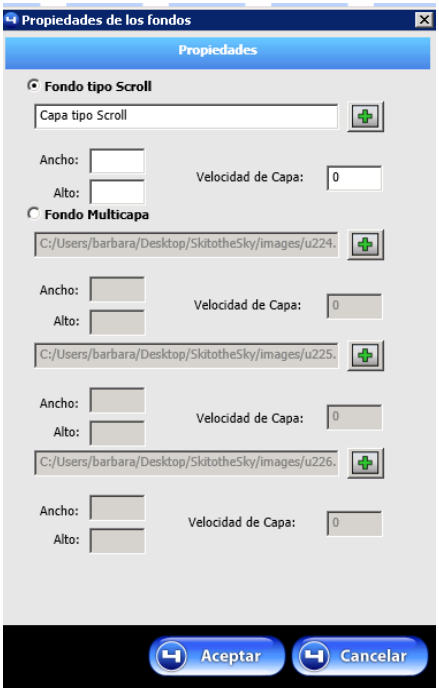


Ilustración 109. Pantalla de propiedades de los fondos

La primera es utilizando "Fondo tipo Scroll", es decir, una única imagen. Se recomienda que las dimensiones mínimas de esa imagen sean 320x480 para evitar zonas vacías. La velocidad de capa representa lo rápido que se moverá el fondo respecto al personaje. Si el fondo se supone que es un objeto cercano la velocidad será mayor; si es un objeto lejano (por ejemplo el cielo) la velocidad será menor.

El fondo multicapa lo que persigue es tener tres fondos diferentes cada una con la velocidad que le corresponde. El efecto que se trata de conseguir es el mismo que cuando una persona va en el coche ve la carretera moverse rápidamente y sin embargo las montañas alejadas se mueven mucho más despacio.

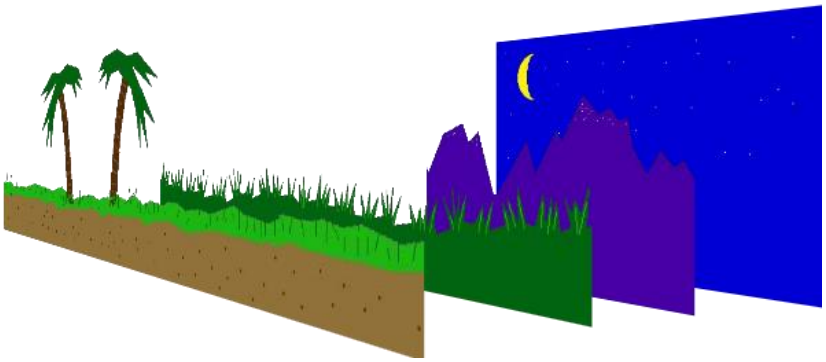


Ilustración 110. Concepto de fondo multicapa

Para conseguir este efecto de la manera más realista posible se recomienda utilizar valores comprendidos entre 1 y 12 en el apartado de velocidades. Además, hay que tener en cuenta que el fondo frontal y el medio deberán tener una parte transparente para poder ver el fondo trasero a través de ellos.

- **Limpiar escenario.** Borra todos los elementos situados en el escenario.
- **Restaurar valores predeterminados.** Hace que todas las propiedades del nivel vuelvan a ser las que ofrece el editor por defecto.

Todos estos parámetros se definen para un único nivel. Para gestionar los niveles del juego el usuario debe dirigirse a la zona de "Niveles" de esta pantalla.

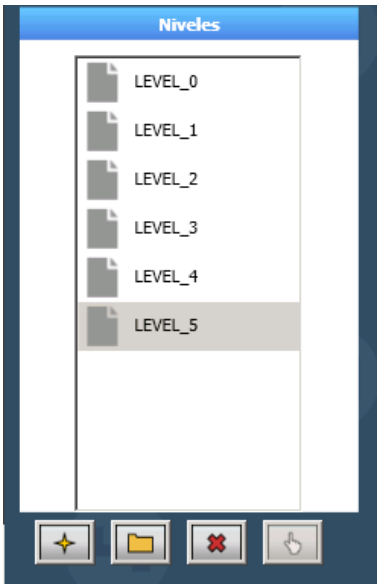


Ilustración 111. Ejemplo de gestión de niveles de un juego

En primer lugar aparece una lista de los niveles ya creados. La funcionalidad de los botones de la parte inferior es la siguiente:

- El primer botón empezando por la derecha crea un nuevo nivel vacío.
- El segundo botón guarda los cambios efectuados en el nivel que se encuentre en ese momento en edición. La acción de guardar las propiedades de un nivel también se efectúa automáticamente cuando se accede a la pantalla anterior o siguiente.
- El tercer botón elimina el nivel, el escenario que contiene y todas las propiedades. El nivel 0 no puede ser borrado porque es obligatorio que el juego contenga al menos un nivel.
- El cuarto botón es el botón de selección. En esta tipología está inactivo.

Una vez se hayan definido todos los niveles del videojuego (hasta un máximo de diez niveles) con sus escenarios y sus propiedades correspondientes se continúa a la última pantalla.

9.3.17 Plataformas de despliegue

Ya se han acabado de definir todas y cada una de las partes parametrizables del juego basado en el control de vehículos. En este punto debe elegirse la plataforma de exportación.

En el momento de redacción de este manual de usuario sólo se disponía de la plataforma Android por lo que el usuario sólo podrá exportar su juego a un proyecto de esta plataforma. Para ello se debe hacer clic en el botón con los tres puntos ("…") y a continuación elegir la ruta donde se desea que se sitúe el proyecto Android final.

Ilustración 112. Captura de la pantalla de elección de la plataforma de despliegue

Pulsando en "Generar juego" y esperando el tiempo pertinente aparecerá un mensaje de éxito si todo ha funcionado correctamente, lo que indicará que el juego se ha exportado correctamente.

9.3.18 Exportación a Eclipse y ejecución

Eclipse es un editor muy completo que usan los programadores para crear sus programas. Como es lógico, el usuario no deberá realizar ninguna tarea de programación, simplemente importará el proyecto ya hecho y Eclipse se ocupará de compilarlo y ejecutarlo.

En primer lugar se debe abrir el programa Eclipse siguiendo el manual de ejecución. Una vez abierto, se debe hacer clic derecho en el área de la pantalla indicada como "Package Explorer". Si el Eclipse no se ha usado con anterioridad, esta área estará vacía. En el menú que se despliegue al hacer clic derecho se seleccionará la opción de "Import". En la ventana de importación se seleccionará "Android" y a continuación "Existing Android Code into Workspace"

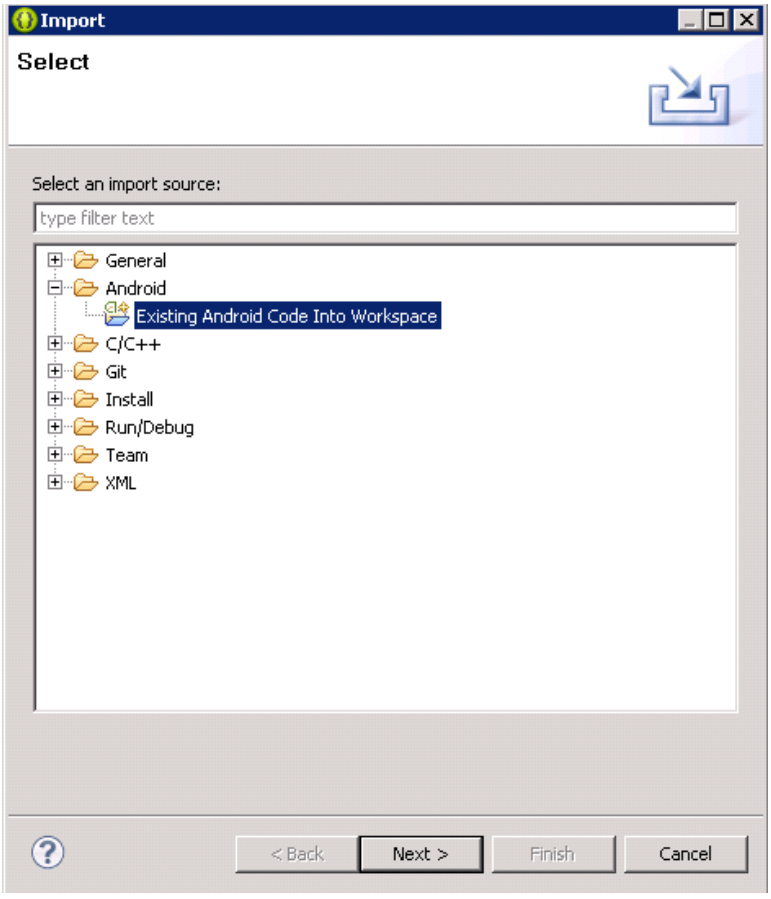


Ilustración 113. Captura del diálogo de importación inicial

Pulsando en "Next" se llegará a otra pantalla donde habrá que hacer clic en el botón "Browse". Se abrirá un diálogo para buscar el proyecto con el juego. El fichero con el juego se encontrará en la ruta que se haya definido en la última pantalla del editor con el nombre "android_platform_vehicles".

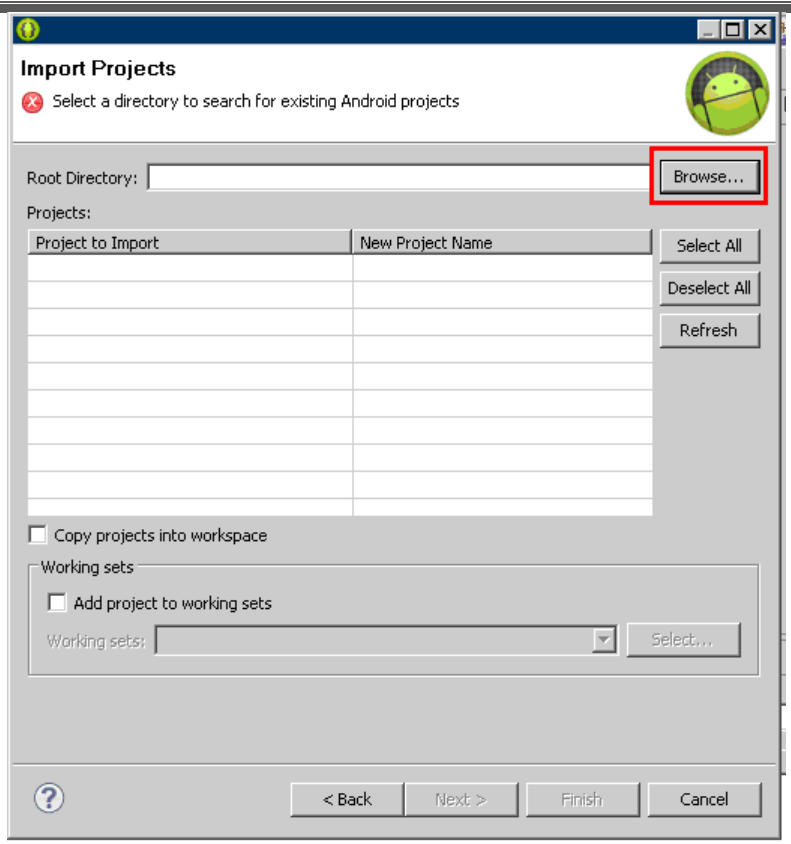


Ilustración 114. Captura de la pantalla de selección de archivo de importación

Una vez seleccionado el proyecto se debe hacer clic en “Finish”. Si este botón no está activo es que el proyecto seleccionado no se corresponde con un proyecto Android.

Si todo ha ido correctamente aparecerá un proyecto llamado “platformVehicles” en el explorador de paquetes. Para ejecutar el juego basta con hacer clic derecho en el nombre del proyecto > “Run as” > “Android application”.

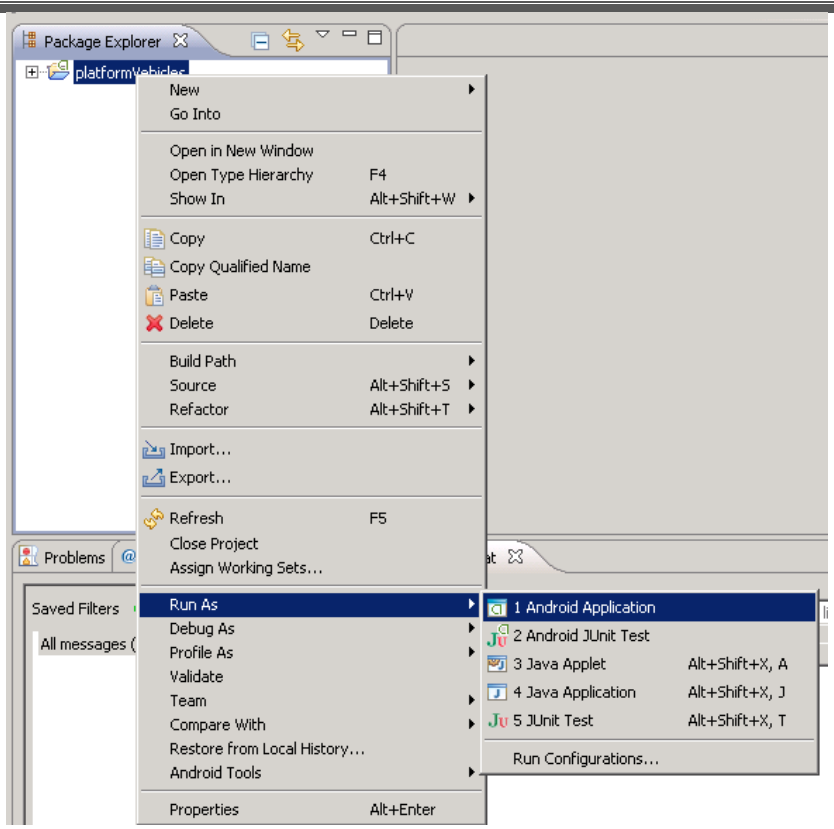


Ilustración 115. Cómo ejecutar el videojuego

Si no se ha conectado ningún dispositivo Android al ordenador, automáticamente se abrirá el emulador con el juego instalado. Puede demorar unos minutos la aparición del juego en el emulador. Si el dispositivo está conectado, el usuario puede escoger la opción de donde prefiere ejecutar el juego.

Si el usuario quiere instalar el juego y no tiene su dispositivo conectado tiene que tener en cuenta que el único fichero que necesita tener en su móvil para instalarlo es el *platformVehicles.apk* situado en la carpeta */bin* del proyecto final.

9.4 Manual del Programador

9.4.1 Subsistema editor

El editor es un subsistema muy amplio e involucra una gran cantidad de clases pues actualmente cuenta con seis tipologías de videojuegos diferentes cada una con su propia funcionalidad.

En el presente manual del programador sólo se hará referencia a los eventuales cambios o ampliaciones que se podrán efectuar de relacionados con la tipología de “Rampas” o de carácter general.

9.4.1.1 Nuevas pantallas

La definición de las pantallas y el orden de las mismas se realiza en la clase *BaseWindow.xaml.cs* situada en la carpeta *Tasks*. Para su declaración se utilizan dos arrays de Strings tal y como se ve a continuación para el caso de la tipología de rampas (cada tipología tendrá sus dos arrays correspondientes)

```
string[] window_names_platform_vehicles = {"Globales","Principal", "Opciones",
"Selección", "Juego", "Pausa","Victoria", "Derrota", "Fin",
"Personaje","Trampas","Checkpoints","Tiles", "Meta","Escenario", "Plataforma"};

string[] window_workflow_platform_vehicles = {
"PlatformVehicles/PageGlobalsPlatformVehicles.xaml", "PageMainScreen.xaml",
"PlatformVehicles/PageOptionsScreenPlatformVehicles.xaml",
"PlatformVehicles/PageSelectLevelScreenPlatformVehicles.xaml",
"PlatformVehicles/PageGameScreenPlatformVehicles.xaml", "PagePauseScreen.xaml",
"PlatformVehicles/PageEndLevelScreenPlatformVehicles.xaml",
"PlatformVehicles/PageLoseLevelScreenPlatformVehicles.xaml",
"PageEndGameScreen.xaml",
"PlatformVehicles/CharacterSetPropertiesPlatformVehicles.xaml",
"PlatformVehicles/TrapSetPropertiesPlatformVehicles.xaml","PlatformVehicles/Checkpoi
ntSetPropertiesPlatformVehicles.xaml",
"PlatformVehicles/TileSetPropertiesPlatformVehicles.xaml","PlatformVehicles/TargetSe
tPropertiesPlatformVehicles.xaml",
"PlatformVehicles/SceneryPlatformVehicles.xaml","PageSelectGamePlataform.xaml" };
```

El primer array (*window_names_platform_vehicles*) contiene los nombres de las pantallas. Esos nombres aparecerán en la parte superior del editor para poder navegar entre las mismas.

El segundo array (*windows_workflow_platform_vehicles*) indica las rutas donde se encuentran las clases que definen cada una de las pantallas.

Si se quiere añadir una nueva pantalla se deberá incluir su nombre en el array y la clase .xaml de su interfaz (tendrá una clase .xaml.cs que implemente la funcionalidad) teniendo en cuenta que las pantallas se mostrarán en el orden en el que se inicialicen en el array.

9.4.1.2 Definir el aspecto gráfico

La mayor parte de las clases del editor se dividen en dos partes:

- Archivo .xaml que representa la interfaz de una pantalla.
- Archivo .xaml.cs con el mismo nombre que implementa la funcionalidad de la pantalla.

Por tanto, si se quiere modificar el aspecto gráfico de una de las pantallas basta acceder al fichero .xaml y realizar los cambios pertinentes, ya sea desde el interfaz gráfico o directamente cambiando el código .xaml.

9.4.1.3 Modificar parámetros del videojuego final

Es posible que en el futuro se desee añadir más funcionalidad a los juegos de esta tipología a través del editor. Actualmente, el editor cuenta con diversas maneras de guardar los parámetros para después inyectarlos en el juego pero todos siguen un patrón común: hacerlo mediante .xml.

En todas las clases que definen las pantallas del juego existe una función llamada *reimplement_writeXML()* que coge todos los parámetros ya definidos y los guarda en un XML. Si se quisiera, por ejemplo, incluir un nuevo botón en la pantalla principal se podría incluir en dicha función un código similar al siguiente en el lugar correspondiente:

```
  xmlMessage = xmlMessage +
"<new_button_image_source>" + newButton.getBackgroundSource() +
"</new_button_image_source>"
+"<new_button_image_width>" + newButton.getWidthImage() +
"</new_button_image_width>"
+"<new_button_image_height>" + newButton.getHeightImage() +
"</new_button_image_height>"
+"<new_button_x_position>" + newButton.getXPosition() +
"</new_button_x_position>"
+"<new_button_y_position>" + newButton.getYPosition() +
"</new_button_y_position>" + "\n";
```

A continuación, como se quiere que estos parámetros aparezcan en el juego se incluiría en la plantilla (carpeta *Templates\android_platform_vehicles\src\com\DSL\DSL.java*) algo similar a lo siguiente teniendo en cuenta que se deben mantener los nombres del XML para que los parámetros se inyecten de manera automática.

```
 public final static String imageNewButton =
%new_button_image_source% ;

public final static int heightNewButton =
%new_button_image_height% ;

public final static int widthNewButton =
%new_button_image_width% ;

public final static int posXNewButton =
%new_button_x_position% ;

public final static int posYNewButton =
%new_button_y_position% ;
```

En general, hay que tener en cuenta que todo cambio en el editor implica guardar el nuevo elemento en un objeto persistente para recuperarlo a la hora de realizar en la sustitución en la plantilla. La plantilla, por tanto, también debe ser modificada para apreciar los cambios en el videojuego final.

9.4.1.4 Incluir otras plataformas de despliegue

El presente proyecto tenía como objetivo realizar un editor de videojuegos Android por lo que el resto de plataformas que ofrece el editor no están implementadas.

Si se quisiera añadir otra plataforma de exportación toda la funcionalidad asociada se encontraría en la clase *PageSelectGamePlataform.xaml.cs* dentro de la carpeta *Tasks.* Una parte de la clase *buttonLaunchAnalizer_Click,* la cual se ejecuta cuando se pulsa el botón "Exportar juego" de la interfaz, presenta la siguiente estructura:

```
if (textboxRouteDestinoAndroid.Text != "")
{
   switch (Store.Instance.type){ … resto código }
}

if (textboxRouteDestinoIphone.Text != "")
{
   switch (Store.Instance.type){ … resto código }
}

if (textboxRouteDestinoHTML.Text != "")
{
   switch (Store.Instance.type){ … resto código }
}
if (textboxRouteDestinoWindowphone.Text != "")
{
   switch (Store.Instance.type){ … resto código }
}
```

Lo que está comprobando dicho código es a qué plataforma debe realizar la importación. Dentro del *if*, el *switch* comprueba cuál de las seis tipologías se está exportando. Por tanto, si se quiere definir otra plataforma de exportación para la tipología de rampas, la estructura del código a incluir dentro del *if* correspondiente sería el siguiente:

```
case "platform_vehicles":
analizadorPlatformVehicles.Program.Main(routes_<<nombre_plataforma>>);
                              break;
```

Como es lógico, para que la importación se realice correctamente debe haberse incluido la plantilla correspondiente en la carpeta *Templates* y el nombre de dicha plantilla debe cumplir el siguiente esquema para que funcione:

<<nombre_plataforma>>_<<tipología>>

En este caso, la tipología se sustituiría por *platform_vehicles* y el nombre de la plataforma podría ser cualquier de las otras tres que aporta el editor:

- Html5
- iOS
- WindowsPhone

9.4.1.5 Añadir nueva tipología

Añadir un nuevo tipo de juego al editor es una tarea larga y compleja que implica crear nuevas pantallas, reutilizar código, adaptar código ya existente a las necesidades de la nueva tipología, implementar nueva funcionalidad, etc.

A grandes rasgos, el proceso empezaría incluyendo en la clase *Store (*carpeta *Global)* la nueva tipología con su identificador, pues existirán varias pantallas comunes a todas las tipologías donde se deberá conocer el tipo del juego en edición. Por tanto, es importante incluir en todos los *switch (Store.Instance.type)* el caso de la nueva tipología.

En segundo lugar se debería crear un flujo de ventanas tal y como se vio en la subsección 9.4.1.1. que defina cuáles van a ser las pantallas de la tipología e implementar cada una de esas pantallas con su funcionalidad correspondiente. Hay que tener en cuenta que se podrá reutilizar mucho código dependiendo de la similitud con otras tipologías pero será necesario implementar, entre otros, un *Store* propio (clase encargada de almacenar algunas propiedades del juego) o un *Serializer* (clase encargada de transformar los datos a binario y recuperarlos para que el usuario pueda guardar el estado de la edición).

También se incluiría la plantilla debidamente implementada con anterioridad en la carpeta *Templates* de la manera que se explicó en la subsección 9.4.1.4.

Por último se debería crear o adaptar un analizador, una serie de clases que obtienen los ficheros XML del editor y sustituyen los datos de la plantilla además de mover los recursos gráficos y sonoros de la salida del editor a las carpetas del proyecto, generando el proyecto Eclipse final.

9.4.2 Subsistema videojuego

9.4.2.1 Modificar funcionalidad

Si se quiere modificar la funcionalidad relacionada con los parámetros establecidos en el DSL, el cambio debe realizarse en el editor como se comentó en la subsección 9.4.1.3.Sin embargo, si lo que se quiere cambiar es la propia lógica del videojuego (no parametrizable desde el editor) es necesario cambiar la plantilla directamente, es decir, el propio código del videojuego.

Las principales clases que intervienen en la lógica del videojuego son: *GameView*, *Level* y *Player*.

La clase *GameView* gestiona los diferentes estados en los que puede encontrarse el mismo (modo normal, nivel perdido, pausa, nivel ganado, etc.) e inicializa la interfaz del juego (marcador de tiempo, número de vidas, botón de pausa y de reinicio). Además, recibe los eventos de pulsación del usuario en la función *onTouchEvent.* Cualquier modificación relacionada con estos aspectos del juego (por ejemplo incluir un nuevo estado, incluir un nuevo botón de interacción, añadir nueva funcionalidad a ese botón...) se llevaría a cabo en la clase *GameView.*

La clase *Level* contiene y gestiona los elementos de cada nivel. Si se quisiera incluir nuevos elementos (por ejemplo ítems a recolectar, enemigos móviles, etc.), modificar el comportamiento del *scrolling,* añadir nuevos *Tiles,* cambiar la comprobación de colisiones u otras posibilidades, los cambios deberían efectuarse en esta clase.

La clase *Player* alberga las propiedades del personaje principal y gestiona la mayor parte de la física del juego. En esta clase se modificarían todos los parámetros de comportamiento del personaje principal: incluir movimientos diferentes como saltos o rebotes, mejorar la física con el entorno para darle una estética más realista, etc.

9.4.2.2 Añadir pantallas nuevas

Para que el videojuego sea coherente con el editor, se debería modificar el mismo para que contenga las nuevas pantallas (para más información sobre cómo hacerlo consultar sección 9.4.1.3 del presente manual).

Para añadir una nueva pantalla en el videojuego se debería crear una clase *Activity* donde se gestionaría el funcionamiento y una clase *View* donde se gestionaría la vista. En la parte de vista se establecerían los botones y el fondo de dicha pantalla de manera análoga al resto de clases *View* del videojuego. Es necesario incluir el nombre de la nueva pantalla en el archivo *"AndroidManifest.xml"* para que funcione correctamente.

Por último, para que la pantalla esté incluida en la navegabilidad del videojuego debe añadirse un botón que acceda a dicha pantalla de la siguiente manera:

```
Intent actStart = new Intent(<<Nombre de la activity inicial>>.this,
<<nombre de la activity nueva final>>.class);

startActivity(actStart);
```

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

El primer paso a la hora de desarrollar el proyecto fue estudiar las diferentes tipologías de juego que ya ofrecía Gade4All y elegir una que no estuviera implementada. Se optó por juegos basados en el control de vehículos por ser un tipo de juego bastante popular actualmente.

En segundo lugar se analizaron varios juegos comerciales descargados de Google Play para extraer sus características y funcionalidad común y así establecer las propiedades el juego a implementar.

Por tanto, el tercer paso consistió en desarrollar un videojuego dentro de esta tipología para plataformas Android partiendo de un conocimiento muy reducido sobre desarrollo de videojuegos. Gracias a tutoriales en Internet, foros y la inestimable ayuda de los directores de proyecto se desarrolló un prototipo de videojuego orientado a vehículos teniendo en cuenta los requisitos definidos en la parte de análisis establecida previamente.

El desarrollo de dicho prototipo ayudó a comprender de manera más exhaustiva la lógica de implementación de un videojuego y las partes del mismo que debían ser externalizadas para personalizarlo al máximo. Tras esta fase se propusieron una serie de elementos iniciales del DSL que variaron poco a lo largo del desarrollo de toda la aplicación y se hicieron grandes modificaciones en la estructura del juego para que éste aceptara dichos parámetros.

Una vez fueron identificados los elementos parametrizables, éstos se incluyeron en ficheros XML de tal manera que los gráficos, sonidos, estructura, física y reglas del juego se definieran a partir de dichos ficheros externos.

El último paso consistió en el estudio del editor gráfico actual de Gade4All para la inclusión de la tipología de videojuegos basados en el control de vehículos. Una vez se comprendió la estructura del mismo se realizaron diversos cambios en la herramienta reaprovechando código de otra tipologías e incluyendo código nuevo para que un usuario pudiera definir toda la instancia DSL utilizando el editor.

Gracias a este proyecto se ha comprobado la utilidad del desarrollo de software dirigido por modelos como metodología para crear abstracciones de software complejo que permitan al usuario generar instancias del mismo de manera sencilla e intuitiva. Las pruebas de usabilidad han demostrado que la generación del videojuego resulta sencilla para los usuarios y que se reduce significativamente la dificultad y el tiempo empleado en el desarrollo del videojuego. Las reiteradas pruebas unitarias que se han llevado a cabo durante todo el proceso de implementación aseguran que los objetivos de funcionalidad del juego resultante también se cumplen.

A nivel personal, estoy muy satisfecha con el trabajo realizado en este proyecto, pues me ha aportado una gran variedad de nuevos conocimientos sobre el desarrollo Android y el desarrollo de videojuegos que seguramente influirán en la elección de proyectos futuros relacionados con este campo.

10.2 Ampliaciones

10.2.1 Exportación del videojuego a diferentes plataformas

El objetivo de este proyecto era desarrollar un editor que exportara videojuegos de vehículos a la plataforma Android. Sin embargo, el editor deja abierta la posibilidad de incluir la exportación también a HTML5, iPhone y Windows Phone; aspecto que no se ha implementado por falta de tiempo y que se podría llevar a cabo incluyendo la plantilla correspondiente en el editor Gade4All.

10.2.2 Incluir nuevos elementos en el videojuego

Las posibilidades de un videojuego de vehículos son inmensas. Podrían añadirse una gran variedad de nuevos elementos que aumenten el dinamismo del juego tales como rampas curvas, *loopings,* cuestas de otras inclinaciones, etc. Aparte de la fisonomía del terreno, también pueden añadirse otros elementos como ítems a recolectar, *bonus* que aumenten la velocidad, etc.

10.2.3 Mejorar física del videojuego

Una de las primeras decisiones que tuvieron que tomarse a la hora de implementar el videojuego era si debía utilizarse un *framework* que se encargara de la física del mismo o implementarla desde cero. Dada la tipología del proyecto se optó por la segunda opción lo que ha desembocado en un mayor control del código pero en una física más pobre. La mejora en este caso pasa por mejorar el código de la física o, lo que es más lógico si estuviéramos en un proyecto real, utilizar motores como jBox2D o AndEngine para conseguir algunos efectos aparentes en el juego: rotaciones completas del coche, suspensión en las ruedas, choques, etc.

10.2.4 Visualización previa de la física

Existen diversos parámetros personalizables por el usuario que afectan a la física del videojuego (gravedad, fricción en cuestas, etc.) Sin embargo, cualquier cambio efectuado sobre estos parámetros no se verá reflejado hasta que el usuario pruebe el videojuego en el emulador o en el dispositivo. Si por ejemplo la gravedad es muy pequeña, el personaje principal no caerá o si los factores de aceleración son muy bajos el personaje apenas se moverá en esa dirección.

Todos estos aspectos influyen en que la salida final sea un juego que no funciona correctamente. Para que el usuario perciba esta deficiencia antes de ejecutar el juego, sería útil incluir un visualizador en el propio editor con un escenario ejemplo y con los parámetros de física elegidos por el usuario para que éste pueda ver de manera directa qué valores le conviene tomar y cuáles generarían un videojuego erróneo.

10.2.5 Uso de redes sociales

Actualmente casi todos los juegos populares para dispositivos móviles cuentan con una opción de compartir resultados en las redes sociales más populares (Facebook y Twitter) Sería atractivo para los usuarios contar con esta opción pues fomentaría la competitividad entre ellos y aportarían una manera gratuita de publicidad para el juego.

10.2.6 Cambiar proceso de generación del videojuego

La salida del editor gráfico actual es un proyecto Android que debe ser importado al Eclipse para posteriormente ser compilado y ejecutado. Como se puede comprobar, no es un proceso directo y precisa que un usuario sin conocimientos técnicos ejecute una herramienta desconocida para él como es el Eclipse.

Una mejora del editor daría al usuario la posibilidad entre generar el proyecto Eclipse o generar automáticamente el fichero apk que debe instalarse en el dispositivo móvil ahorrándonos así el paso intermedio.

10.2.7 Posibilidad de cargar juegos DEMO en el editor

Aunque el editor aporta muchos elementos por defecto, el usuario debe definir al menos un nivel para obtener el videojuego básico. Para mostrar las posibilidades de los videojuegos resultantes y ayudar a los usuarios menos expertos o que quieran desarrollar un juego en el menor tiempo posible, el editor puede ofrecer la posibilidad de cargar videojuegos ya finalizados y completo, con sus propios gráficos y niveles ya definidos.

Capítulo 11. Aplicaciones reales del proyecto

El objetivo final del proyecto era generar videojuegos basados en el control de vehículos de una forma sencilla y rápida mediante un editor intuitivo y fácil de utilizar. Para justificar la utilidad del mismo y mostrar las posibilidades de los eventuales juegos de esta tipología que pueden generarse se incluyen dos videojuegos realizados con la herramienta. Ambos juegos están disponibles para su descarga en Google Play.

11.1 Ski to the Sky

El primer juego completo creado con el editor fue Ski to the Sky. Aunque en un principio estaba previsto realizar un juego típico de coches se optó por diseñar uno relacionado con el mundo de la nieve y donde el personaje principal fuese un esquiador para probar que la tipología ofrece buenos resultados con distintos tipos de personajes.

Ilustración 116. Personaje principal de Ski to the Sky

Los *Tiles,* fondos del nivel, puntos de control y trampas se diseñaron en consonancia con la temática del juego en busca del aspecto más realista posible.

Ilustración 117. Trampa de Ski to the Sky

También a los botones y fondos de los menús se les dio un aspecto invernal.

Ilustración 118. Pantalla principal del juego Ski to the sky

Una vez diseñados los recursos gráficos se decidió plantear un videojuego donde la velocidad fuera un aspecto necesario para pasar de nivel por lo que se aumentó la velocidad máxima por defecto tanto para derecha como para la izquierda y se definieron tiempos para superar los niveles que exigieran mucha agilidad y velocidad a la hora de controlar el personaje. Para aumentar la dificultad se escogió un factor de fricción menor del que el editor propone por defecto para conseguir el efecto de "resbalarse" tan común en la nieve. Además, se activó la opción de "Inercia" de tal manera que el esquiador baja por las cuestas si permanece quieto encima de ellas tal y como ocurre en la vida real.

Ilustración 119. Captura de la pantalla del juego Ski to the Sky en ejecución

También se utilizó música ambiente en los menús y en el juego, además de efectos para la pérdida de una vida y activación del punto de control. El videojuego final es una aventura de cinco niveles donde cada nuevo nivel es más difícil que el anterior.

El videojuego está disponible para su descarga en Google Play (véase sección 12.2 Referencias en Internet)

Ilustración 120. Captura de Ski To the Sky en Google Play

11.2 Happy Dolly

Otro juego realizado mediante la herramienta de Gade4All dentro de la tipología de control de vehículos es Happy Dolly, donde el personaje principal es una oveja que corretea por las colinas del campo.

Ilustración 121. Personaje principal de Happy Dolly

Se eligió este personaje para comprobar que la tipología podía aceptar cualquier tipo de personajes, incluso aunque éstos no fueran vehículos o medios de transporte. Para aumentar el dinamismo del juego se incluyó un movimiento en las patas de la oveja cuando ésta estuviese en movimiento.

Las trampas también se definieron como elementos dinámicos de tal manera que tuvieran sentido dentro del mundo de las granjas o el campo.

Ilustración 122. Animación del lobo que hace de trampa en Happy Dolly

Ilustración 123. Animación del fuego que hace de trampa en Happy Dolly

Para definir los botones se siguió un diseño común basado en la madera así como los fondos, respetando los colores verde (hierba) y azul (cielo). Los *Tiles* fueron diseñados para representar la hierba o la tierra por la cual camina la oveja y los puntos de control fueron definidos como flechas de madera que giran cuando están activas.

Ilustración 124. Pantalla principal de Happy Dolly

Para diferenciarse del juego anterior se decidió hacer una captura de pantalla de cada nivel una vez se definieron los cinco de los que consta el juego y utilizar esta imagen como miniatura de cada nivel en la pantalla de selección de nivel con su tiempo a batir correspondiente.

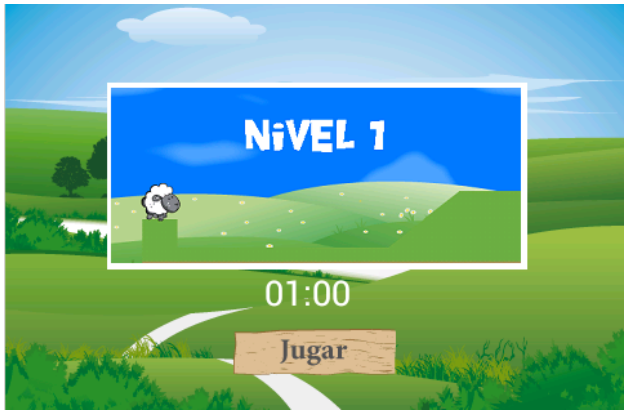


Ilustración 125. Pantalla de selección de nivel de Happy Dolly

Otra diferencia respecto al juego anterior consistió en definir una pantalla de pausa transparente, de tal manera que el usuario puede ver el escenario del juego detrás aunque no pueda interactuar con él.

Ilustración 126. Pantalla de pausa del videojuego Happy Dolly

Por último, la física imperante en el juego se basó en una gravedad menor para poder llegar más lejos en los saltos, un factor de fricción algo mayor para que la oveja no resbale tanto y se controle mejor y velocidades máximas menores para que la oveja no se mueva tan rápido. Este hecho era importante porque, al contrario que en el juego anterior, *Happy Dolly* se basa más en encontrar el camino correcto hacia la meta y no caer en las trampas o en los precipicios, más que en completar los niveles con rapidez. Al igual que en el videojuego anterior también se añadieron sonidos y música ambiente para agregar dinamismo a la experiencia de juego.

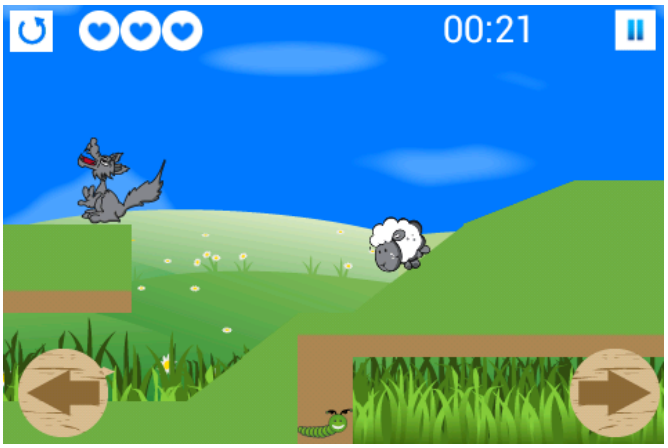


Ilustración 127. Captura del juego Happy Dolly en ejecución

El videojuego está disponible para su descarga en Google Play (véase sección 12.2 Referencias en Internet)

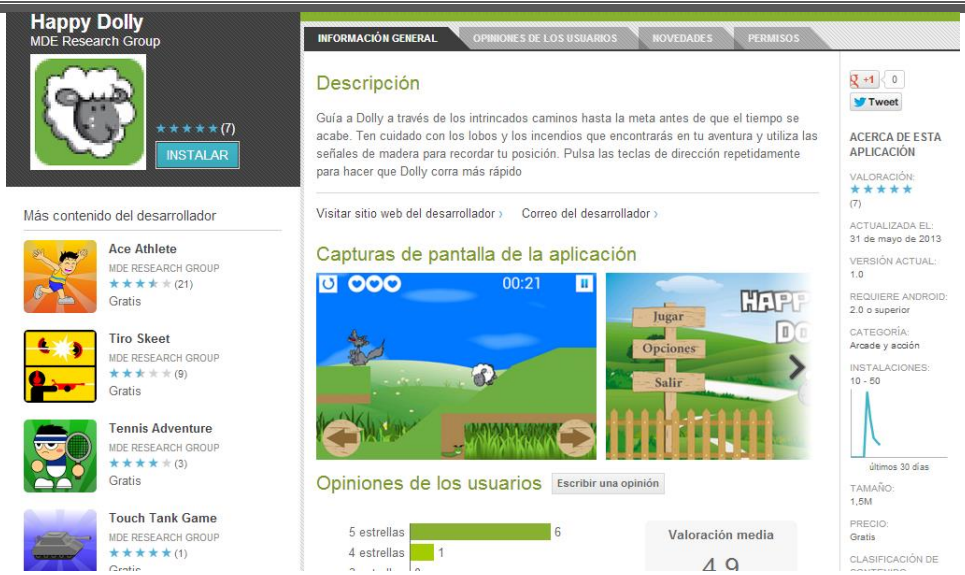


Ilustración 128. Captura de Happy Dolly en Google Play

Capítulo 12. Referencias Bibliográficas

12.1 Libros y Artículos

[Stack12] Patrick Stack. "History of video game consoles". Time magazine. 2012

[Donovan10] Tristan Donovan. "The history of video games". Yellow Ant. 2010

[Kent01] Steven L. Kent. "The Ultimate History of Video Games: From Pong to Pokemon--The Story Behind the Craze That Touched Our Lives and Changed the World". Three Rivers Press. 2001

[García12] Vicente García Díaz. "Una breve introducción a la Ingeniería Dirigida por Modelos". Universidad de Oviedo. 2012.

[Byous98] Jon Byous. "Java technology: The early years". Sun developer. 1998.

12.2 Referencias en Internet

[Martin Fowler] Domain -specific languages http://martinfowler.com/dsl.html

[Microsoft] .NET http://www.microsoft.com/net/

[Microsoft] Información general sobre XAML (WPF) http://msdn.microsoft.com/es-es/library/ms752059.aspx

[Android] Android http://www.android.com/

[Android] Source Android http://source.android.com/

[Google Play] Google Play Store https://play.google.com/store

[App Store] App Store http://store.apple.com/es

[Google Play] Hill Climb Racing https://play.google.com/store/apps/details?id=com.fingersoft.hillclimb

[Google Play] Skater Boy https://play.google.com/store/apps/details?id=com.game.SkaterBoy

[Google Play] Stickman Ski Racer https://play.google.com/store/apps/details?id=com.djinnworks.StickmanSkiRacer.free

[Google Play] Ski To the Sky https://play.google.com/store/apps/details?id=com.SkitotheSky

[Google Play] Happy Dolly https://play.google.com/store/apps/details?id=com.HappyDolly

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

.NET: framework de Microsoft que pretende integrar una amplia variedad de productos software con independencia de la plataforma de hardware.

Android: sistema operativo diseñado principalmente para dispositivos móviles con pantalla táctil (*smartphones* o *tablets*).

Arquitectura dirigida por modelos: metodología de implementación que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos.

C#: lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.

DSL: lenguaje de dominio específico. Se trata de un lenguaje de programación creado con un objetivo concreto.

Eclipse: entorno de desarrollo utilizado para implementar principalmente con código Java.

Gade4All: gran proyecto en el que incluirá este trabajo fin de Máster. El objetivo de Gade4All consiste en un crear un generador de diferentes tipologías de videojuegos.

Google Play: repositorio de todas las aplicaciones y videojuegos subidos por desarrolladores que pueden descargarse y ser ejecutados en dispositivos Android.

GSL: término adaptado de DSL. Lenguaje de dominio específico propio del proyecto Gade4All destinado a los videojuegos.

Java: lenguaje de programación de propósito general orientado a objetos.

Plantilla: documento que sirve como patrón a partir del cual se crearán unos u otros videojuegos, personalizando sus partes parametrizables.

Videojuego de plataformas: aquellos videojuegos en 2-D en los que el personaje se mueve de izquierda a derecha de la pantalla superando una serie de obstáculos hasta llegar al final del nivel. P. ej.: Mario Bross, Sonic, etc.

Visual Studio: entorno de desarrollo integrado para sistemas operativos Windows que soporta varios lenguajes de programación como C#, C++, Visual Basic, etc.

13.2 Contenido Entregado en el CD-ROM

En esta sección se incluye una descripción de los contenidos que se encuentran en el CD presentado junto con la documentación.

13.2.1 Estructura de directorios

Directorio	Contenido
./ Directorio raíz del CD	Contiene un fichero leeme.txt explicando toda esta estructura.
./DSL_PlatformVehicles_Android	Contiene toda la estructura de directorios del proyecto.
./DSL_PlatformVehicles_Android/HappyDolly	Contiene el proyecto Eclipse de uno de los videojuegos creados mediante la herramienta.
./DSL_PlatformVehicles_Android/SkitotheSky	Contiene el proyecto Eclipse de otro de los videojuegos creados mediante la herramienta.
./ DSL_PlatformVehicles_Android /Editor	Contiene la solución del editor dividido en subcarpetas. Se ha preparado el mismo para que sólo pueda exportar juegos de la tipología implementada en este proyecto. También contiene los ficheros necesarios para realizar la instalación.
./DSL_PlatformVehicles_Android /Documentacion	Contiene en su raíz el presente fichero .pdf que representa la documentación del proyecto.
./DSL_PlatformVehicles_Android /Documentacion/UML	Contiene el fichero generado mediante Enterprise Architect que alberga la definición de los diagramas que aparecen en las secciones de análisis y diseño.
./DSL_PlatformVehicles_Android /Documentacion/Javadoc	Contiene los ficheros de documentación del subsistema videojuego generados automáticamente mediante Javadoc para el videojuego Happy Dolly de ejemplo.
./DSL_PlatformVehicles_Android /Documentacion/Planificacion	Contiene el archivo creado mediante Gannt Project que expone la planificación a seguir para la realización del proyecto con la imagen exportada.

13.3 Índice Alfabético