UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Notebook Wars

Videojuego tipo shoot'em'up 2d para dispositivos móviles

Trabajo Fin de grado

Grado en Ingeniería Informática

Autor: Francisco Ferreres García
Director: Ramón Pascual Mollá Vallá
Julio 2014

Resumen

El trabajo consiste en el desarrollo de un videojuego destinado a plataformas móviles, mediante el entorno de desarrollo de videojuegos Unity 3D. El videojuego es de tipo *shoot'em'up*, en el que van apareciendo naves enemigas, y el jugador debe destruirlos para recoger las monedas que estos sueltan, y así poder comprar nuevas naves y armamento, para ser cada vez más poderoso.

Palabras clave

shoot'em'up, disparos, aviones, Unity, móviles, videojuego, app, android, iOs

Summary

The project consists in the development of a mobile-oriented video game, using the development environment Unity 3D. The video game is a shoot'em'up; multiple enemies come across the screen, and the player must destroy them to get all the coins they drop, so he can buy new weapons and planes, to be more and more powerful.

Keywords

shoot'em'up, shooter, planes, Unity, mobile, videogame, app, Android, iOs

Tabla de contenidos

Introducción

En este capítulo se dan a conocer las motivaciones responsables del desarrollo de este videojuego, así como los principales objetivos propuestos para el proyecto.

Motivación

Durante muchos años el género de los videojuegos *shoot'em'up* ha sido de los más exitosos, tanto en maquinas recreativas, como en consolas, saliendo al mercado miles de títulos con multitud de temas y variantes como por ejemplo de las guerras mundiales, futuristas, de naves espaciales, e incluso algunos más originales con frutas, dragones, o personajes de animación voladores.

Actualmente, los videojuegos que desarrollan las compañías suelen presentar enormes mundos en 3D completamente explorables, y con gráficos 3D difíciles de diferenciar de la realidad, pero se ha perdido el encanto que tenían aquellos juegos llenos de colores, luces y explosiones, quedando este género bastante abandonado, saliendo únicamente algunos títulos a manos de desarrolladores de videojuegos independientes.

El mercado de los videojuegos en dispositivos móviles ha presenciado un crecimiento exponencial en los últimos años, y se prevé que en los próximos años siga creciendo.

Figura 1. Billones de Euros movidos por la industria de los videojuegos en plataformas móviles.

Observando las principales listas de aplicaciones de iOs y Android, difícilmente se encontrará algún videojuego de este tipo, a no ser que se busque específicamente, y se podrá observar que no hay mucha variedad ni calidad, además de ser simples imitaciones de los juegos antiguos, sin aportar ningún tipo de novedad.

Tras analizar estos hechos, han sido motivación más que suficiente para decidirme a desarrollar un videojuego de este tipo, con algunos cambios y mejoras, que lo harán mucho más entretenido y adictivo, aprovechando además el auge que hay en el uso de aplicaciones móviles, en el que veo una gran oportunidad de negocio.

Objetivos

El objetivo primordial de este proyecto es el desarrollo de un videojuego mediante el entorno de desarrollo de videojuegos Unity 3D, el cual permite desplegar la aplicación en multitud de plataformas, entre ellas los sistemas operativos iOs y Android, que son los sistemas objetivo al reunir entre los dos la mayor parte del mercado de aplicaciones.

Mediante el desarrollo del videojuego se pretende:

- Aprender a usar la herramienta de desarrollo de videojuegos Unity 3D.
- Iniciarse en el mundo del desarrollo de videojuegos y adquirir experiencia.
- Desarrollar un videojuego comercial, con el cual poder obtener beneficios económicos a través de un patrocinador.

La decisión de comercializar el juego mediante un patrocinador se debe a que el patrocinador dispondrá de muchos más medios para promocionar y distribuir el juego, consiguiendo así una mayor difusión, y por lo tanto, mayores beneficios.

El videojuego es un *shoot'em'up* con los gráficos dibujados en papel de libreta y escaneados, lo que será una de las características únicas del videojuego respecto a la competencia. El jugador debe conseguir todo el dinero posible, eliminando a los enemigos y recogiendo las monedas que estos sueltan al explotar, para así poder comprar una gran variedad de aviones, y armas que equipar en estos aviones. La posibilidad de personalizar el avión, con los gustos y estrategias del jugador, es uno de los puntos fuertes que hacen el juego muy divertido y adictivo.

El videojuego será creado desde cero, por lo que se deberán realizar las siguientes tareas:

- Diseño del videojuego.
- Creación del diagrama de Gantt para gestionar la planificación.
- Adquisición y creación del material audiovisual.
    - Dibujar, escanear y preparar gráficos.
    - Crear y adquirir música y efectos de sonido.
- Programación del videojuego.
    - Programación de menús e interfaces.
    - Programación de la tienda de aviones y armas.
    - Programación del juego.
        - Generación procedimental de fondos.
        - Inteligencia Artificial de los diferentes enemigos.
        - Aplicación de las personalizaciones al avión del jugador.
        - Control del jugador y del entorno.
- Testear y ajustar la dificultad de los niveles.
- Testeo en varios dispositivos.
    - Testeo en la propia maquina.
    - Testeo en iOs.
    - Testeo en Android.
- Creación de la documentación.

Estado del arte

Los primeros videojuegos desarrollados para dispositivos móviles aparecieron en la década de los 90, el primer juego en aparecer fue uno en el que el jugador manejaba una serpiente que crecía y aceleraba al conseguir puntos, y lo desarrollaron los ingenieros de Texas Instruments para sus calculadoras. El juego tuvo tanto éxito, que en 1997 Nokia decidió incluirlo en sus móviles, y en los años siguientes más de 350 millones de teléfonos móviles han ofrecido este juego por defecto.

Figura 2. Nokia 5110, el primer modelo en incluir el juego *Snake*.

Hoy en día los nuevos videojuegos se caracterizan por tener unos gráficos increíblemente realistas, ya que, al disponer cada vez de maquinas más potentes, los desarrolladores siempre buscan sacar el máximo partido al hardware del usuario. Se usan modelos y texturas difíciles de distinguir de la realidad, y aunque la potencia no es tanta en los dispositivos móviles, cada día es mayor, y en poco tiempo conseguirán tener la misma potencia que algunas consolas y equipos de sobremesa. Ya existen videojuegos en dispositivos móviles que alcanzan la calidad grafica de los mejores juegos de plataformas de sobremesa de hace 3 o 4 años, con gráficos en alta resolución, sonido de alta calidad y grandes mundos explorables por el jugador.

Figura 3. ***Ravensword: Shadowlands. Videojuego para iOs y Android lanzado a finales de 2013***

Por supuesto esto solo ocurre en el caso de los grandes títulos AAA, tras los cuales hay grandes empresas que invierten millones de dólares, y varios meses, o incluso años, para el desarrollo de un videojuego. En el caso de los desarrolladores independientes todo es muy diferente; suelen ser equipos de entre 1 y 10 personas como máximo, aunque lo más habitual es disponer de 1 ó 2 programadores, 1 ó 2 artistas gráficos, y un artista de audio. Otras tareas más generales necesarias para la producción del videojuego se reparten equitativamente en el equipo, o se hacen en conjunto, por ejemplo, desarrollar nuevas ideas, promocionar el juego, buscar un patrocinador, manejar la contabilidad, etc.

Estos equipos no pueden permitirse invertir grandes cantidades de dinero, la mayoría simplemente invierte gran parte de su tiempo en el desarrollo de los videojuegos que ellos mismos diseñan, pudiendo desarrollar un juego completo en años, semanas, o incluso algunas horas, dependiendo de la complejidad del videojuego.

La popularidad del desarrollo independiente de videojuegos a aumentado en los últimos años, y la industria se ha adaptado, permitiendo así a los desarrolladores alcanzar los mercados de las principales videoconsolas, por ejemplo Xbox Live, PlayStation Network, Steam, o Nintendo Eshop.

Figura 4. ***Cave Story 3D*****: Videojuego independiente que se lanzó en la consola de Nintendo** ***N3DS*****.**

Crítica al estado del arte

No hay duda de que hoy en día el colectivo que mayor innovación aporta al mundo de los videojuegos es el de los desarrolladores independientes, ya que estos pueden arriesgar mucho más en el diseño del videojuego, al no haber invertido grandes cantidades de dinero, ni tener tras ellos grandes juntas de accionistas ni directivos, a los que les incomode arriesgar su dinero sin la seguridad de recuperarlo.

Los grandes estudios de desarrollo, raramente lanzan un juego realmente novedoso, y la mayor parte de las ocasiones el concepto del videojuego viene de uno creado por un desarrollador independiente, con éxito demostrado, y con muchos aspectos que una empresa con muchos más recursos podría mejorar, o es fruto del trabajo de un estudio de desarrolladores independientes, que se ha convertido en un gran estudio gracias a su éxito, y llegando a lanzar su producción en multitud de plataformas y formatos. Algunos de estos juegos son Angry Birds, Braid, World of Goo ó Minecraft.

Figura 5. Imágenes de Angry Birds, Braid, World of Goo y Minecraft.

Otro de los género que los grandes estudios de desarrolladores de videojuegos suelen olvidar es el de los juegos retro, el cual sigue siendo el preferido de gran parte de la comunidad de jugadores, y en el cual se puede innovar mucho, al añadir nuevas características ahora posibles gracias a los avances en la potencia de las maquinas, o que simplemente no se le habían ocurrido a nadie. Personalmente, pienso que este género puede ser de los más favorables para iniciarse en el desarrollo de videojuegos, ya que son juegos con éxito demostrado, no son muy complejos, y se les puede añadir nuevas funcionalidades que muchos jugadores siempre desearon ver en su videojuego favorito.

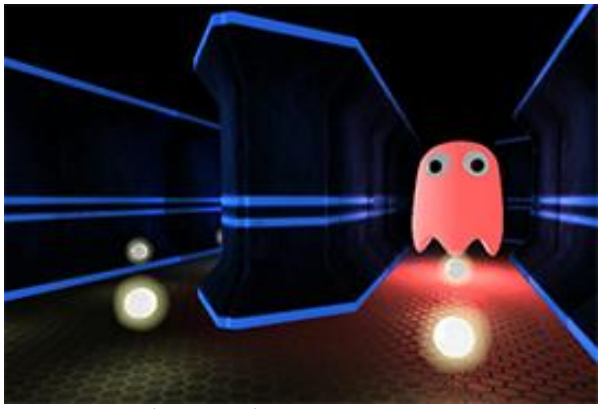

Figura 6. First-person Pacman

Propuesta de trabajo

En este capítulo se realiza un análisis del mercado de los videojuegos antecesores al diseñado, así como un análisis DAFO. También se especifica en qué consiste el videojuego, mediante una descripción detallada de sus elementos y estados. Finalmente se muestra un plan del trabajo realizado mediante un diagrama de Gantt, con las previsiones temporales estimadas inicialmente.

Análisis de mercado

Todo género de videojuegos tiene su origen, un videojuego que innovó en el mercado marcando tendencia, y el caso de los *shoot'em'up* no es distinto, el precursor de este género fue Spacewar!, creado en 1961 por unos estudiantes del MIT, aunque no puede considerarse realmente un *shoot'em'up*, ya que en este juego el jugador no se enfrentaba a múltiples enemigos controlados por inteligencia artificial, si no que se enfrentaban dos jugadores humanos entre sí, pero sin duda fue el juego que causó el nacimiento del género.

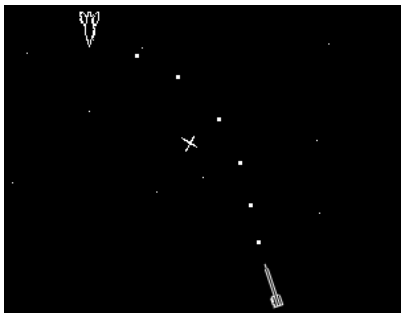

Figura 7. Spacewar!

El primer *shoot'em'up* autentico fue Space Invaders, desarrollado por la compañía nipona Taito en 1978, y generando todo un revuelo en la época debido al gran éxito que tuvo, causando una escasez de monedas en Japón, y tras el cual aparecieron muchísimos más juegos de este género en los siguientes años y en la década de los 80, algunos de sus primeros sucesores fueron juegos tan conocidos como Galaxian, Asteroids, o Galaga.

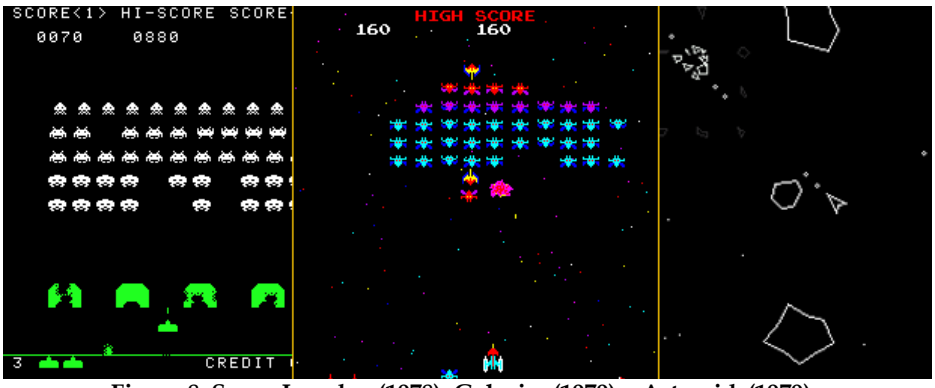

Figura 8. Space Invaders(1978), Galaxian(1979) y Asteroids(1979).

La época dorada de los *shoot'em'up* comenzó en la década de los 80, para terminar aproximadamente en 1995, y en estos años se lanzaron cientos de títulos en los que se trataba de innovar y mejorar la experiencia de juego con nuevas funcionalidades, por ejemplo con fondos mas dinámicos, nuevos objetivos del juego como defender a la población, o la inclusión de diferentes niveles con dificultad progresiva.

Uno de los grandes hitos en el género fue la aparición de los videojuegos con grandes y detallados fondos que se desplazaban hacia abajo y cambiaban en cada nivel, siendo el primer juego en implementar esta funcionalidad Xevious, creado por Namco en 1982, y tras el cual vinieron muchos más, creando su propio subgénero, el de los *vertical scrolling shooters*, con títulos de grandísimo éxito, como la saga 1942, ó Raiden.

Figura 9. Xevious(1982), 1942(1984), y Raiden(1990)

Al llegar los años 90 las bases de este subgénero ya estaban establecidas, y los nuevos juegos tenían pocas variaciones respecto a los antiguos, ya que no se quería cambiar la formula con éxito demostrado de estos.

Esta fue una de las causas del fin de la época dorada de los *shoot'em'up*, junto con la aparición de nuevos géneros, entre ellos los videojuegos de lucha, que comenzaban a gozar de una gran popularidad. Todo terminó en 1994 cuando el estudio nipón Toaplan, uno de los mayores desarrolladores de *vertical scrolling shooters*, cerraba sus puertas, lo que para muchos marcó el fin de la época dorada.

Desde el fin de la época dorada pocos estudios se han vuelto a aventurar en el desarrollo de títulos para este género, cuatro de estos estudios surgieron de las cenizas de Toaplan: Raizing, Cave, Gazelle y Takumi. Creando los dos primeros un nuevo subgénero de los *shoot'em'up*, el *bullet hell*, en el que el jugador debe evitar una infinidad de proyectiles disparados por los enemigos, siendo uno de los mas jugados la saga DodonPachi.

Figura 10. Dodonpachi(1997)

Estos videojuegos eran extremadamente difíciles, al tener que esquivar y memorizar el jugador cientos de balas en movimiento con sus respectivos patrones, pero aquí se encuentra la clave del éxito de este subgénero, ya que hacían creer al jugador que esquivaba todas las balas con maniobras casi milagrosas, cuando lo que realmente ocurría es que el *hit box*, o caja de colisiones del jugador, era mucho más reducido que el propio avión.

Figura 11. Ejemplo de caja de colisiones en DoDonPachi.

Mientras este tipo de juegos tenían bastante éxito en las salas de recreativos, en las nuevas consolas aparecían muy pocos videojuegos del tipo *shoot'em'up*, pero estos eran de una gran calidad, por ejemplo R-Type Delta (1999) en la PSX, Gradius V (2004) en la PS2, o Radian Silvergun (1998) en la Sega Saturn, considerado este último el mejor *shoot'em'up* de la historia.

Figura 12. R-Type Delta (1999), Gradius V (2004), y Radian Silvergun (1998)

En el panorama de los videojuegos desarrollados para plataformas móviles también hay un hueco para el género *shoot'em'up*, a pesar de no tener muchos títulos, pero estos son de buena calidad ya que la mayoría son versiones para móvil de los antiguos clásicos, o adaptaciones con nuevos gráficos y prácticamente el mismo estilo de juego.

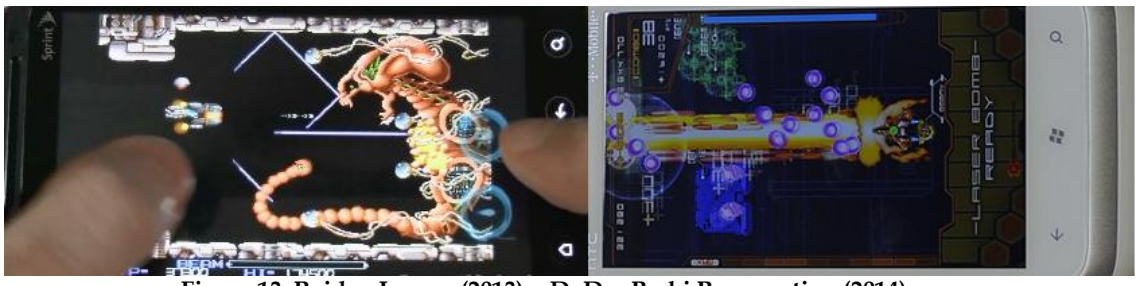

Figura 13. Raiden Legacy (2013) y DoDonPachi Resurrection (2014)

Análisis DAFO

El análisis DAFO es un análisis estratégico destinado a estudiar la situación de una empresa o proyecto, analizando sus características internas (**D**ebilidades y **F**ortalezas) y su situación externa (**A**menazas y **O**portunidades), el cual es de gran utilidad para diagnosticar la capacidad competitiva del proyecto y así determinar la mejor estrategia con la que alcanzar los objetivos propuestos.

Matriz DAFO

Fortalezas	**Oportunidades**
• Originalidad y creatividad, al ser joven y nuevo en el desarrollo de videojuegos para móvil. • Pasión por los videojuegos. • Gran motivación en el proyecto. • Conocimientos avanzados en programación. • Aprendizaje rápido de nuevos lenguajes y técnicas de programación.	• El mercado es global, por lo que hay mas posibles jugadores. • La distribución a través de la red, es muy rápida y sencilla. • Aprendizaje de nuevos entornos de desarrollo. • Los *shoot'em'up* existentes en el mercado móvil no aportan muchas novedades a títulos anteriores.
Debilidades	**Amenazas**
• Experiencia nula en el desarrollo de videojuegos destinados a plataformas móviles. • Pocos fondos disponibles para el proyecto. • Sin conocimientos suficientes para generar audio y gráficos de calidad. • Un único miembro en el equipo de desarrollo. • Pocos recursos para realizar acciones de marketing.	• El mercado es global, por lo que hay más competencia. • Plazo de entrega bastante limitado. • El género *shoot'em'up* no es tan popular como solía ser.

Estrategias: Análisis CAME

El análisis CAME es el paso posterior a la realización de la matriz DAFO, en el que usamos la información extraída para:

- **C**orregir las **D**ebilidades: Estrategia adaptativa.
- **A**frontar las **A**menazas: Estrategia de supervivencia.
- **M**antener las **F**ortalezas: Estrategia defensiva.
- **E**xplotar las **O**portunidades: Estrategia ofensiva.

	Fortalezas	**Debilidades**
Oportunidades	Estrategia ofensiva • El videojuego será muy original, con características que lo diferencie del resto. • Se usará un entorno de desarrollo nuevo para el desarrollador, con alta productividad.	Estrategia adaptativa • Se usarán herramientas gratuitas. • Se usarán efectos de sonidos de 16 bits, fáciles de generar. • Los gráficos del juego serán dibujados en libreta y escaneados.
Amenazas	Estrategia defensiva • Se centrará la mayor parte del trabajo en la programación. • Se añadirán nuevas características al videojuego, nunca vistas en un *shoot'em'up* clásico.	Estrategia de supervivencia • La calidad de los materiales audiovisuales usados no será muy alta. • Se buscará un patrocinador que promocione el juego y con el que se repartirán los beneficios.

Especificaciones

Notebook Wars es un videojuego del genero de los *shoot'em'up*, específicamente un *vertical scrolling shooter*, de ambientación moderna en el que cientos de enemigos aparecen por la parte superior y por los laterales de la pantalla, con diferentes formaciones y patrones de movimiento, y diferentes tácticas de ataque al jugador.

El jugador debe destruir a todos los enemigos que pueda, para así obtener la mejor puntuación posible y la mejor clasificación, que se mide en estrellas, siendo 0 el mínimo y 3 el máximo, y basándose ésta en la proporción de enemigos eliminados sobre el total. Pero la principal motivación para tratar de eliminar el mayor numero de enemigos posibles es conseguir la mayor cantidad de dinero que se pueda, al recoger las monedas que los enemigos sueltan al explotar., y se dispersan por el escenario.

Con el dinero recogido a lo largo de los niveles el jugador puede comprar multitud de distintas armas y naves, para así ser capaz de progresar en los sucesivos niveles, ya que los enemigos son cada vez mas y mas fuertes, y sería imposible superar los primeros niveles sin mejorar la nave y el armamento.

En la tienda se pueden comprar lar armas y las naves, y modificarlas, cada nave tiene un numero distinto de huecos para armas, entre 1 y 4, en el que el jugador puede colocar las armas que desee, siempre que las haya comprado antes, y así poder personalizar su nave con sus gustos y estrategias, lo que será un factor decisivo para superar los niveles con mayor facilidad.

En total se pueden elegir entre 7 naves distintas, cada una con mejores atributos que las anteriores, los cuales son:

- Armadura.
- Velocidad.
- Potencia del imán de monedas.
- Numero de slots para armas.

También se dispone de 22 armas distintas, cada cual más poderosa y cada una con sus cualidades únicas, con las que se puede cambiar la estrategia, por ejemplo, comprando un cañón de bolas de nieve que ralentizará a los enemigos, a pesar de causar menos daño.

La lista de armas es la siguiente:

- Ametralladora.
- Pistola laser.
- Pistola de plasma.
- Cañón de bolas de nieve.
- Pistola de clavos.
- Lanzador de plasma.
- Ametralladora de titanio.
- Cañón de metralla incendiaria.
- Lanzacohetes.
- Laser de alta energía.
- Cañón de energía.
- Ametralladora de cohetes.
- Bombas napalm.

- Pistola Gauss.
- Cañón de materia oscura.
- Cañón de onda sónica.
- Ametralladora de plasma.
- Pistola de rayos.
- Cañón de plasma.
- Ametralladora de laser.
- Ametralladora de rayos.
- Ametralladora Gatling.

El juego dispone de 15 niveles, en los que se van alternando los 3 diferentes escenarios que se generan aleatoriamente: prados, desiertos y mar. En cada nivel aparecen nuevos enemigos, con distintas formaciones, las cuales son:

- Fila horizontal.
- Fila vertical.
- Fila diagonal.
- Aleatorio.

Pueden aparecer por la parte superior de la pantalla, o por los laterales, e independientemente de la formación que tengan, también pueden tener distintos patrones de movimiento, que son los siguientes:

- Avance lineal.
- Avance con una vuelta en medio de la pantalla.
- Avance moviéndose hacia los lados para esquivar los disparos.
- Avance lineal, pausa para disparar, y más avance lineal.

En el juego hay 26 distintos enemigos, cada uno con sus características y atributos únicos, sus estrategias de ataque se basan en 4 cualidades:

- Dispara ó no dispara.
- Velocidad de disparo.
- Ráfaga de balas por cada disparo.
- Apunta al jugador o dispara hacia delante.

Cada enemigo tiene más armadura y poder de ataque, pero sus formaciones y patrones de movimiento no dependen del tipo de enemigo, por lo que cualquier enemigo puede tener cualquiera de estos.

El juego está diseñado para que haya una correcta progresión de la dificultad, calibrada con las naves y aviones que podemos comprar con el dinero aproximado que se va consiguiendo en los niveles superados. La única forma de conseguir esto es a base de muchas pruebas y pequeñas correcciones a lo largo de todo el juego, pero es un paso imprescindible para lograr un juego divertido a la vez que desafiante, sin que llegue a aburrir.

Plan de trabajo

El tiempo del que se dispone para la realización del proyecto es de aproximadamente 2 meses, por lo que hay que organizar muy bien el tiempo destinado a cada actividad para que todo se pueda terminar con el máximo nivel de calidad posible. Se tratará de seguir la planificación planteada, aunque en algunos casos será difícil ya que la previsión del coste de algunas tareas es muy complicado, y más en el caso del desarrollo de un videojuego, ya que es un proceso en el que se pueden añadir, quitar, y modificar características en cualquier punto del desarrollo

Diagrama de Gantt

El diagrama de Gantt es una herramienta que sirve para mostrar gráficamente el tiempo que se tiene previsto dedicar a las diferentes tareas de un proyecto. A continuación se muestra el diagrama de Gantt del proyecto, realizado mediante la herramienta de libre uso GanttProject.

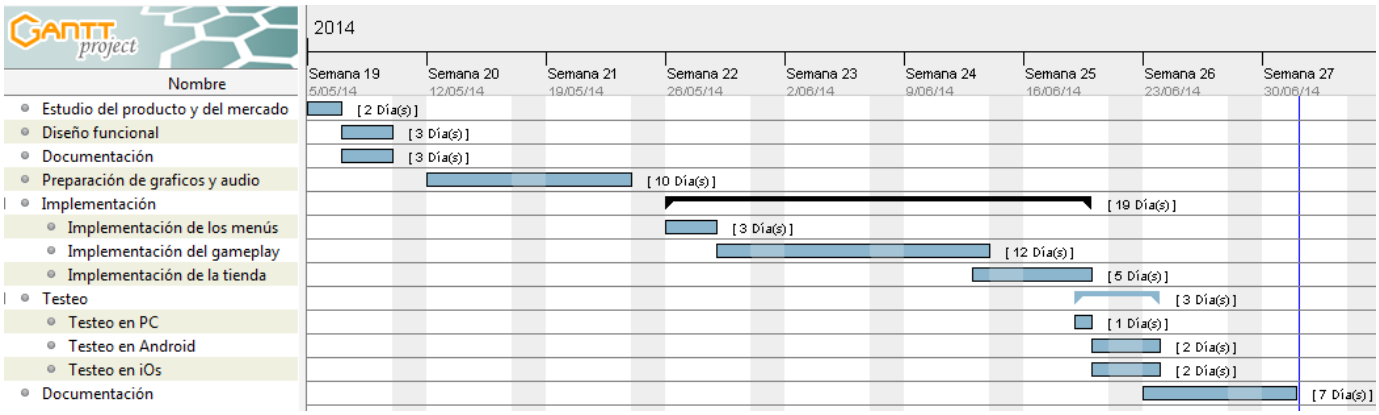

Figura 14. Diagrama de Gantt del proyecto

Diseño y especificación

En este capítulo se especifica la tecnología que se va a emplear para el desarrollo y el despliegue del videojuego. También se muestran los bocetos del diseño funcional de las diferentes pantallas, y finalmente se define la estructura del videojuego mediante un diagrama de clases y un diagrama de flujo.

Tecnología a emplear

Lo primero de todo era la elección del entorno de desarrollo a utilizar, y para ello había que fijarse en varios factores: precio, eficiencia, tamaño de la comunidad de usuarios, documentación disponible, complejidad de aprendizaje del nuevo entorno, dependencia de otras herramientas, etc.

Se analizaron varias herramientas y se muestra en la siguiente comparativa.

Herramienta	Precio	Eficiencia	Comunidad	Documentación
Corona SDK	$16/$49 al mes	Buena eficiencia usando herramientas adicionales Corona Editor y Composer GUI.	Comunidad de usuarios grande.	Amplia documentación y tutoriales.
Unity 3D	Gratuito	Incluye editor de mundos y de código.	Comunidad de usuarios muy grande.	Amplia documentación y muchos tutoriales en la red.
Marmalade	$150 Anual	Ritmo de trabajo más lento, al ser solo compilador de código, y C++ en el que no tengo demasiada experiencia.	Comunidad escasa debido a que no tiene tanta popularidad como la competencia.	Documentación algo escasa.
Flash+Starling	36,89€ al mes	Eficiencia de desarrollo muy buena con flash, pero no tan buena al ejecutarse en dispositivos móviles.	Comunidad pequeña, debido al decline que está sufriendo la comunidad Flash frente a HTML5.	Documentación decente.

Tras realizar el análisis se ha decidido que el entorno de desarrollo que se va a utilizar para la creación del videojuego es Unity 3D, que además de ser uno de los entornos de desarrollo de videojuegos más utilizados, también es un motor para juegos, y permite el despliegue de la aplicación en multitud de plataformas sin ningún tipo de programación extra, desde formato para PC o web, pasando por consolas y los principales sistemas operativos de móviles, como Android o iOs.

La elección de esta herramienta para el desarrollo del proyecto es un factor decisivo, ya que es de uso libre, incluso para proyectos comerciales, y es mucho más productiva que cualquier otra herramienta de desarrollo de videojuegos, ya que en el mismo entorno se incluyen todas las herramientas que podamos necesitar, además de la posibilidad de desplegar en diferentes sistemas.

Unity 3D consta de 2 grandes zonas principales, el editor del mundo, y el editor de *scripting*.

El editor del mundo es muy parecido a una herramienta de edición 3D, se pueden realizar las mismas tareas, como añadir objetos, cámaras o luces, modificar sus propiedades, y visualizar el mundo que se está creando en diferentes vistas y perspectivas, pero también se pueden realizar tareas como asignar *scripts* a objetos, modificar las propiedades físicas de estos, añadir animaciones a los materiales, etc.

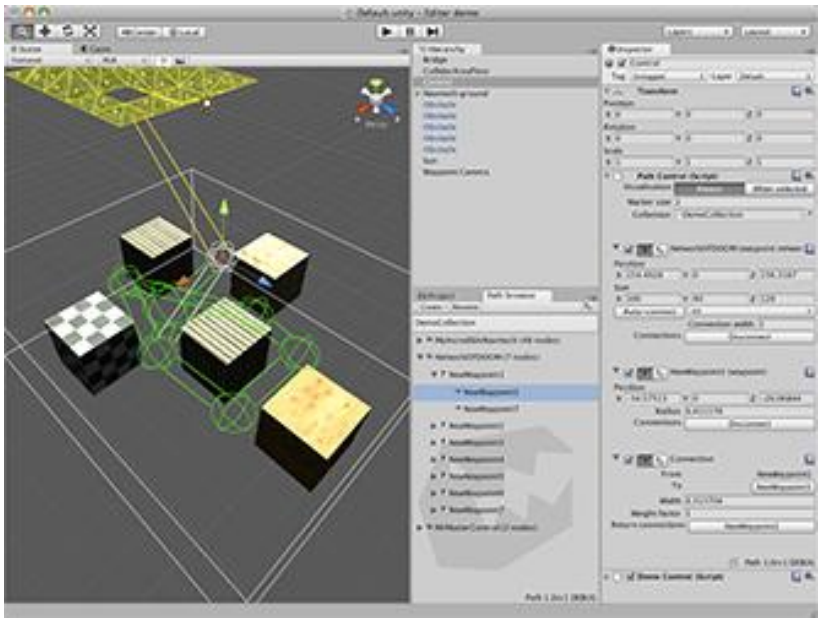

Figura 15. Editor de Unity 3D.

El editor de *scripting* es una versión modificada de MonoDevelop, un entorno de desarrollo diseñado principalmente para C# y otros lenguajes .NET, pero este compila el código directamente desde el editor de Unity, pudiendo ver cómo afectan al juego los cambios en el código de forma inmediata.

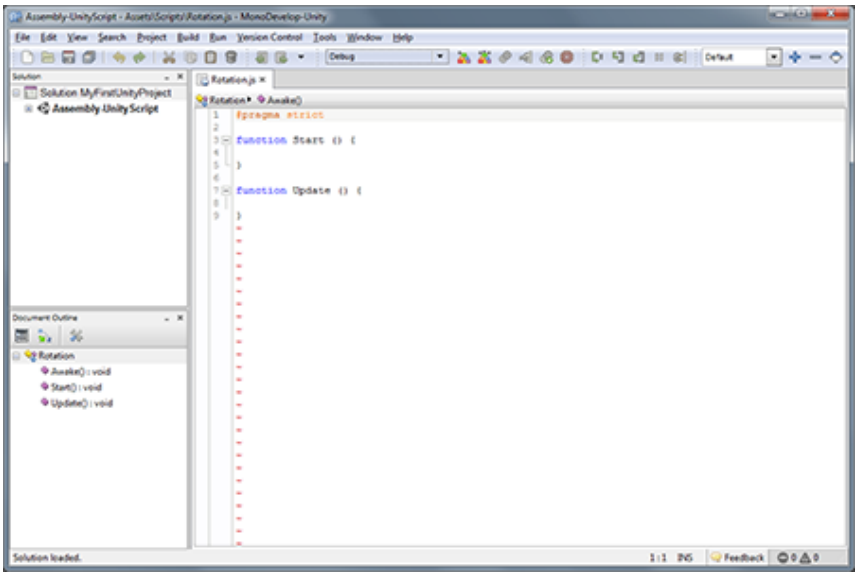

Figura 16. MonoDevelop.

Unity 3D permite el uso de 3 lenguajes para el desarrollo de los videojuegos, incluso usar varios lenguajes a la vez, en diferentes *scripts*, por supuesto. Los 3 lenguajes que soporta Unity3D son C#, Boo, y UnityScript, el cual es una variación de JavaScript adaptada para Unity.

Para el desarrollo de este proyecto se va a utilizar únicamente el lenguaje UnityScript, al ser el más familiar para el desarrollador, y haber mayor numero de tutoriales al respecto.

Metodología

Para el desarrollo de este proyecto se ha recurrido a metodologías de desarrollo agiles, en concreto, eXtreme Programming (XP). Se ha elegido esta metodología al ser un proyecto para el que no se dispone de mucho tiempo, y la simplicidad es algo imprescindible.

Por lo que se tratará de simplificar al máximo el diseño, y realizar refactorizaciones del código siempre que se vea necesario.

Las principales características de esta metodología en las que se va a prestar especial atención son las siguientes:

- Refactorización del código.
- Simplicidad del código.
- Desarrollo iterativo e incremental.
- Pruebas unitarias frecuentes.
- Corrección de todos los errores antes de añadir nueva funcionalidad.

Diseño funcional

A continuación se muestran los bocetos que fueron dibujados en la fase de diseño para tener claros los diferentes estados en los que puede estar el juego.

Menú principal

Esta es la primera pantalla que aparece al iniciar el juego, tiene un botón de jugar que lleva a la pantalla de selección de nivel, un botón para salir del juego, y seguramente se incorporará un botón con un link que redirigirá a la pagina del patrocinador.

Figura 17. Diseño del menú principal.

Pantalla de selección de nivel

En esta pantalla se muestra un mapa, que se puede arrastrar con el dedo, al ser más grande que la pantalla, y desde el cual se puede acceder a los niveles o al hangar, pulsando en los correspondientes botones, y también se puede volver al menú principal.

Figura 18. Diseño de la pantalla de selección de nivel.

Hangar

Esta es la pantalla en la que se pueden comprar distintas armas y naves, y personalizarlos. Se puede pulsar en la nave para mostrar una lista con todas las disponibles, y así cambiarla o comprar una nueva. Con las armas se puede hacer lo mismo, y dependiendo del número de huecos para armas que tenga la nave, se mostraran los correspondientes botones.

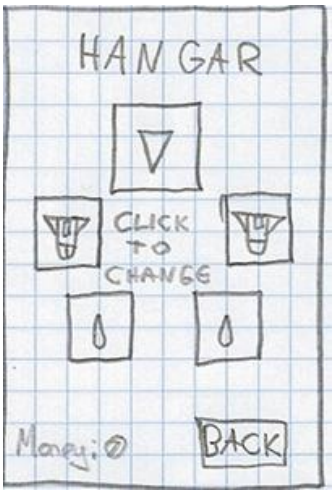

Figura 19. Diseño de la pantalla del hangar.

Pantalla de juego

En esta pantalla se desarrolla toda la acción del juego, la nave del jugador se coloca X pixeles por encima del dedo al tocar la pantalla, para que así el dedo no la tape, y la nave dispara automáticamente cuando se está tocando la pantalla.

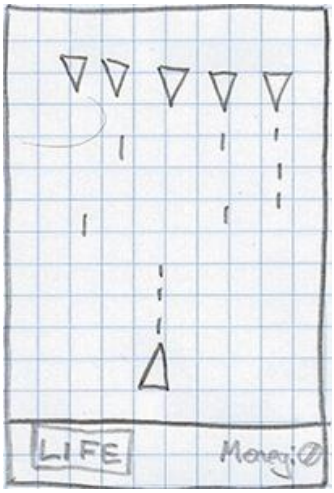

Figura 20. Diseño de la pantalla de juego.

Pantalla de nivel superado

Esta es la pantalla que se muestra al superar el nivel, que no es más que un dialogo que se muestra indicando la calificación obtenida, entre 0 y 5 estrellas, y un botón para continuar, que dirige al jugador a la pantalla de selección de nivel.

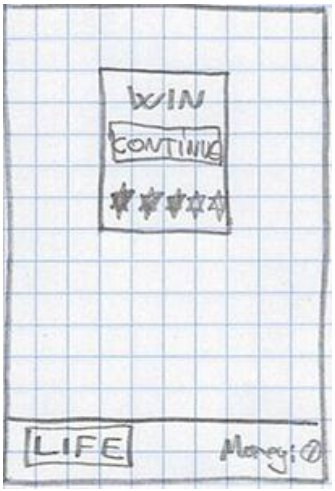

Figura 21. Diseño del dialogo de nivel superado.

Pantalla de nivel no superado

Esta otra pantalla se muestra cuando la vida de la nave del jugador llega a cero, por lo que la nave explota y se muestra este dialogo, que permite volver a intentar el nivel o mostrar la pantalla de selección de nivel.

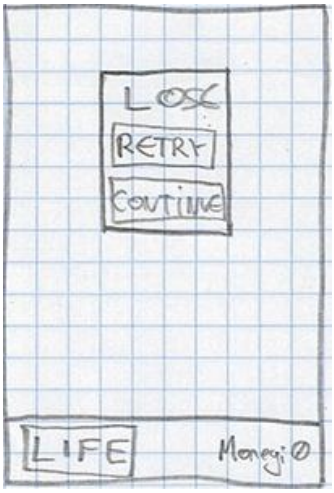

Figura 22. Diseño del dialogo de nivel no superado.

Pantalla de pausa

Al pulsar el botón de pausa en la pantalla de juego aparece un dialogo de pausa, que permite resumir el juego, o volver al menú de selección de nivel.

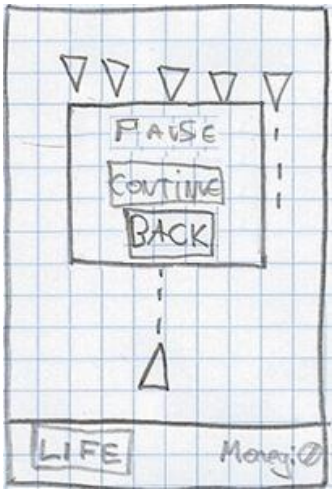

Figura 23. Diseño del dialogo de pausa.

Estructura del proyecto

Diagrama de clases

Un diagrama de clases es una representación grafica de las estructura de un sistema, mostrando sus clases, con sus atributos, métodos, y relaciones entre ellas.

Cada clase se muestra como una caja con 3 secciones:

- Nombre de la clase en la parte superior.
- Atributos de la clase en la parte media.
- Métodos de la clase en la parte inferior.

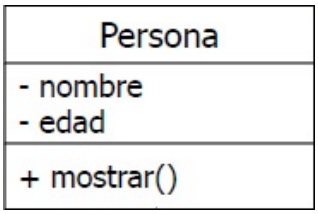

Figura 24. Ejemplo de clase en un diagrama de clases.

Además, a la izquierda de cada atributo o método hay unos símbolos que indican la visibilidad de estos, los cuales pueden ser:

- + : Publico
- - : Privado
- # : Protegido
- ~ : Paquete

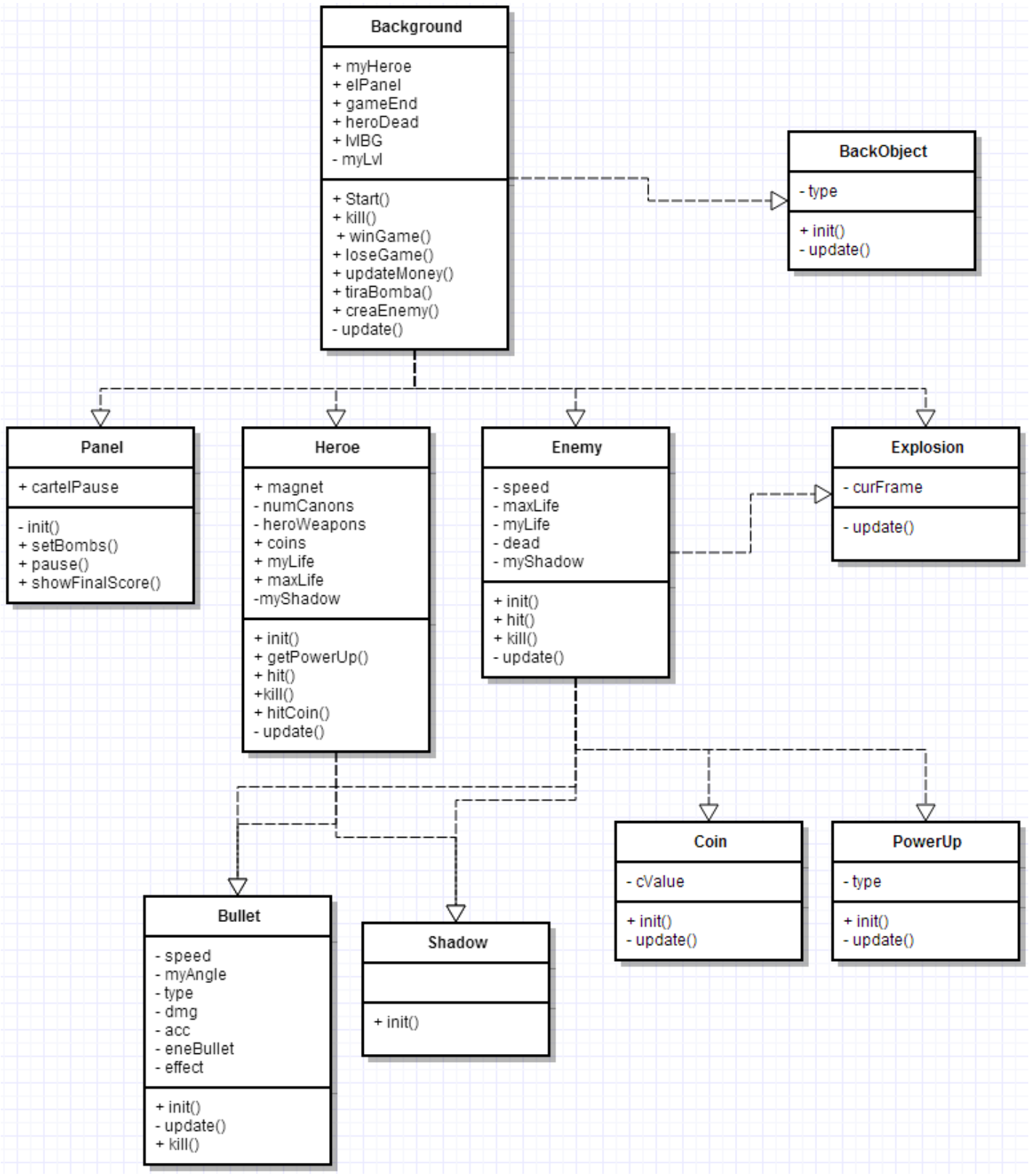

Figura 25. Diagrama de clases (1/2)

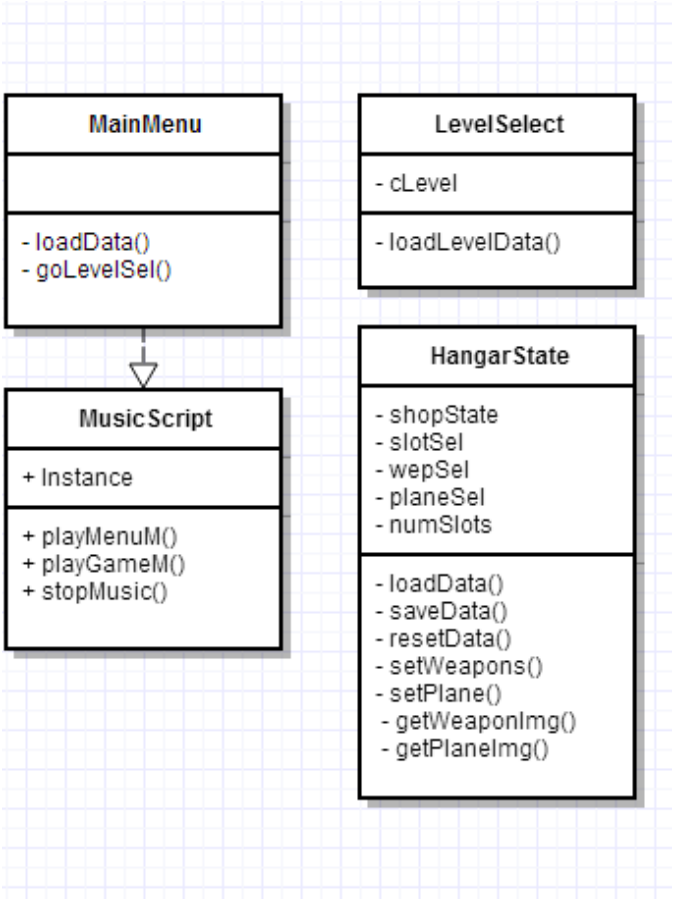

Figura 26. Diagrama de clases (2/2)

Diagrama de flujo

Un diagrama de flujo es la representación grafica de un algoritmo, flujo de trabajo o proceso, en este caso se representan los posibles estados en los que puede estar el videojuego.

Cada estado se corresponde con una captura de pantalla, indicada por el numero de figura.

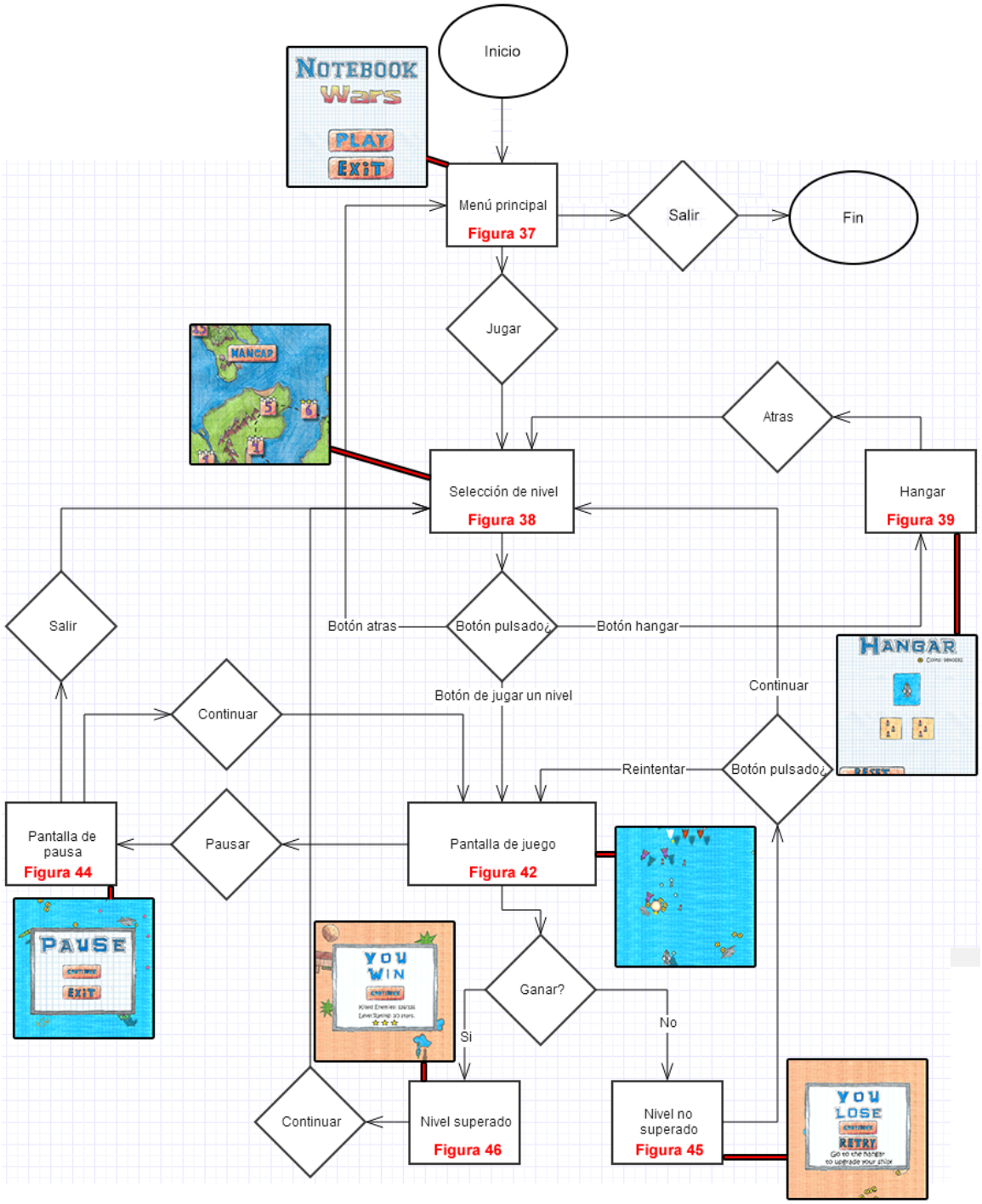

Figura 27. Diagrama de flujo del videojuego.

Desarrollo

A partir del diseño anteriormente detallado se realizó el desarrollo de la aplicación, comenzando con la preparación de todos los elementos gráficos, y finalizando con la creación e implementación de los efectos de sonido, ya que los elementos gráficos son fundamentales para el desarrollo del videojuego, al contrario que los efectos de sonido, que pueden implementarse en el último momento sin problema.

Creación de los gráficos

Los gráficos de este videojuego tienen un estilo bastante innovador, ya que son dibujados a libreta y previamente escaneados, sin apenas ningún tratamiento digital.

A continuación se muestra una de las hojas escaneadas con grafico del juego.

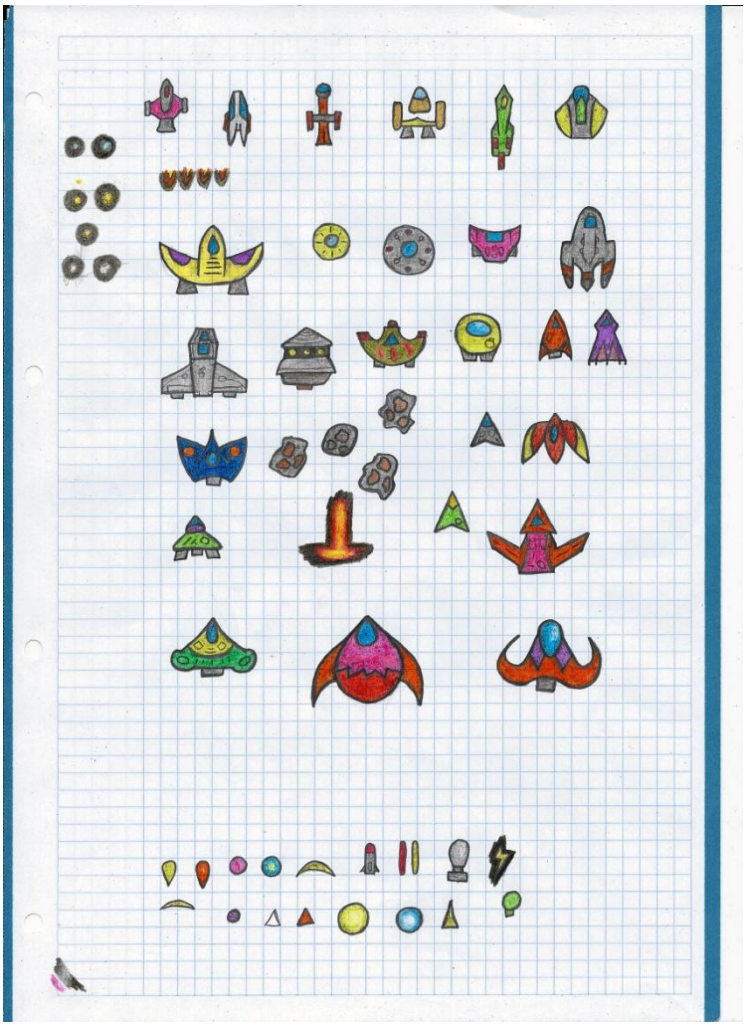

Figura 28. Una de las hojas escaneadas con gráficos del juego.

Creando un juego 2D con Unity 3D

Para desarrollar un juego 2D con Unity, que está enfocado principalmente a desarrollar videojuegos en 3D, hay que conocer varios fundamentos y realizar varios ajustes a las cámaras del videojuego, así como a las texturas y a varios elementos del editor.

Lo primero que se necesita es un plano, un objeto básico con 2 dimensiones a partir del cual se crearan todos los objetos del juego. El problema con los planos del editor de Unity es que en realidad están compuestos por 10x10 cuadrángulos planos, 100 cuadrángulos cuando solo necesitamos 1.

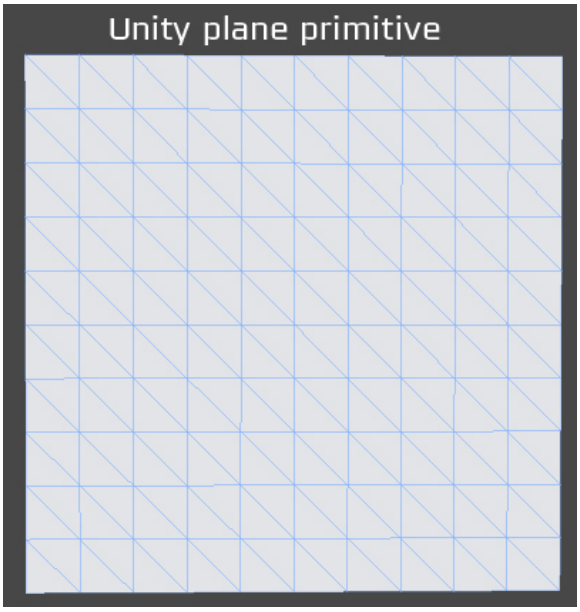

Figura 29. Plano del editor de Unity.

El problema de esto es que al usar estos planos, cada objeto del juego usaría 100 veces más geometría de la que realmente necesita.

Hay varias soluciones para este problema, por ejemplo crear un plano en un editor 3D externo, por ejemplo Maya, o en mi caso use la opción de usar un Script externo de Unity para crear un plano simple. Este método me pareció el más sencillo, ya que solo debía copiar el script en la carpeta del proyecto, y la opción para crear planos simples aparecía en el menú del editor de Unity.

Una vez que se dispone de los planos simples, hay que añadirle su correspondiente textura, y de nuevo aparece un problema, al estar Unity enfocado a 3D, las texturas están suavizadas y preparadas para ser usadas en objetos 3D, por lo que no aparecen en su tamaño correcto, y no se aprecian los detalles.

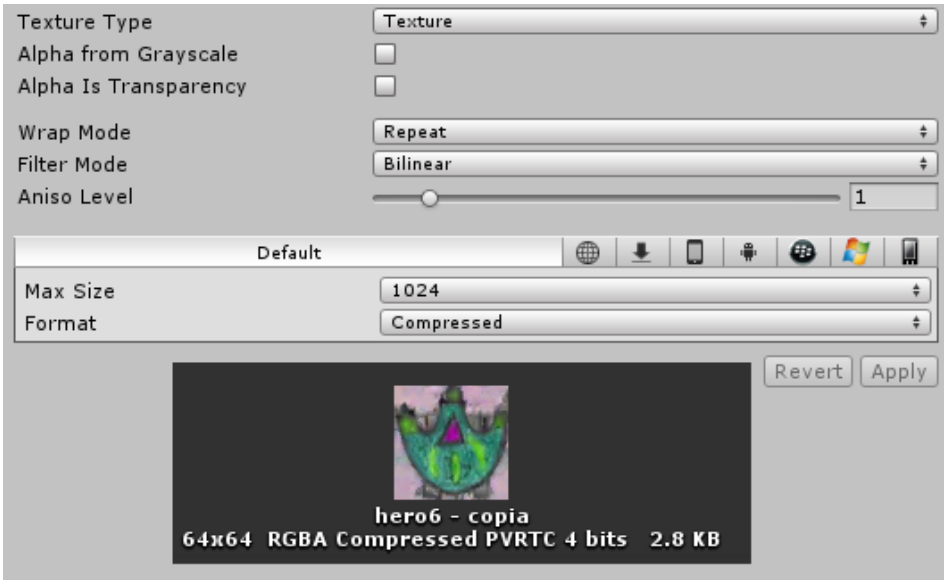

Figura 30. Textura con las opciones por defecto de Unity.

Para corregir esto hay que realizar varios ajustes a cada textura, cambiar el tipo de textura a avanzado, para así poder modificar más detalladamente sus propiedades, quitar los suavizados y filtrados, y quitar los *Mip-maps*, ya que estos no son necesarios en un juego 2D, y usarían recursos innecesariamente. Tras estos ajustes, la textura se mostrará correctamente.

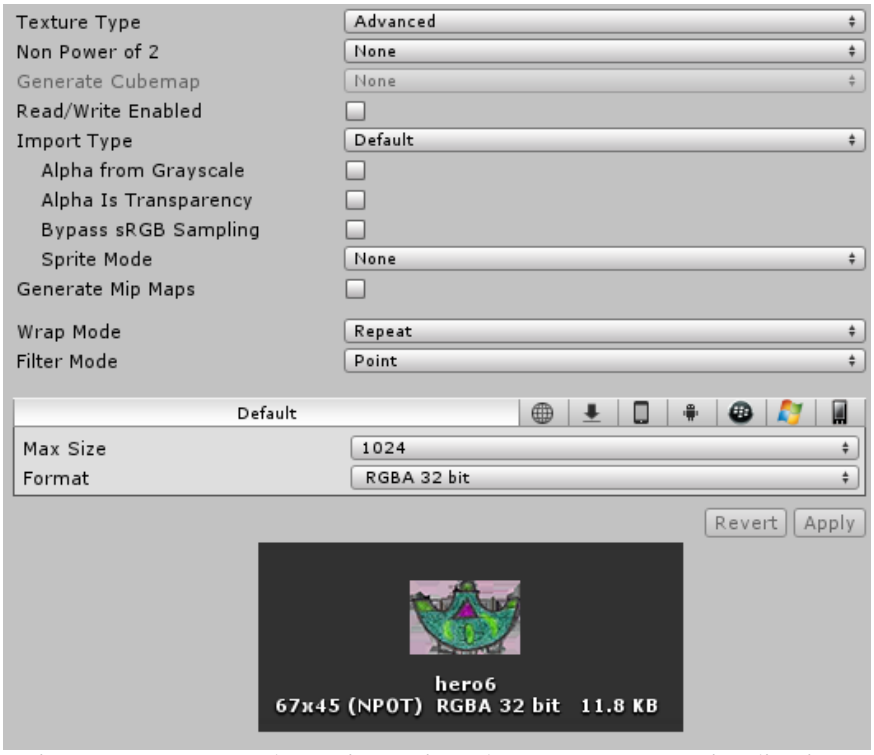

Figura 31. Textura con las opciones ajustadas para su correcta visualización.

Ya están solucionados los problemas de los objetos planos, y de mostrar las texturas correctamente, pero a la hora de visualizar el juego, se ven los planos en 3 dimensiones, con perspectiva en 3D, por lo que no se muestran correctamente.

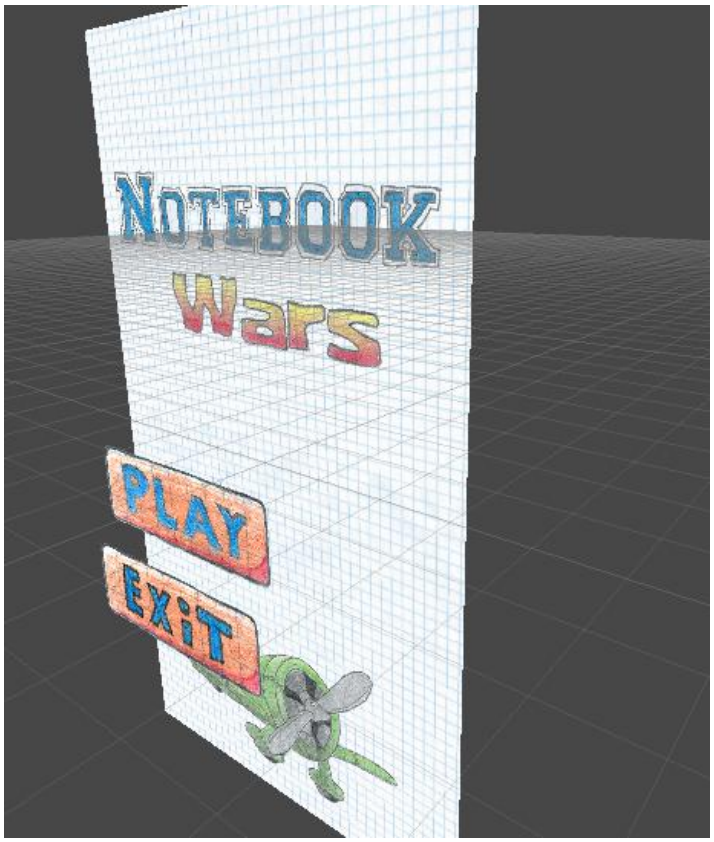

Figura 32. Escena 2D mostrada en 3D.

Para solucionar este problema hay que realizar varios ajustes en la cámara de cada escena.

Lo primero es cambiar el tipo de proyección de la cámara, de perspectiva a ortográfica, de este modo la escena se visualizará en 2 dimensiones. Lo siguiente es cambiar el tamaño de la proyección de la cámara, para que las texturas se muestren en el mismo tamaño que se han diseñado.

Obtener el tamaño adecuado de la proyección ortográfica de la cámara es el proceso más complicado de los ajustes necesarios para visualizar correctamente videojuegos en 2D.

El tamaño de la proyección ortográfica expresa el número de unidades de medida del mundo que caben en la mitad superior de la proyección.

Por lo que el tamaño debe ser de la mitad de la resolución vertical del dispositivo en el que se ejecute el videojuego.

El videojuego se ha diseñado desde el principio para ser mostrado correctamente en una pantalla de iPhone 5 (1136x640), por lo que el tamaño de la proyección ortográfica en este caso será de 568, pero a continuación se mostrará cómo se adapta este tamaño a cada dispositivo, lo que es más complicado de lo que parece, ya que al cambiar el tamaño para otras proporciones de pantalla, se pierde área de visualización en los laterales, ya que el tamaño de la proyección se basa en las medidas verticales, y esto no es aceptable en un videojuego de este tipo, ya que muchos enemigos no llegarían a mostrarse en la pantalla, por lo que el tamaño debe ajustarse a la resolución horizontal del dispositivo.

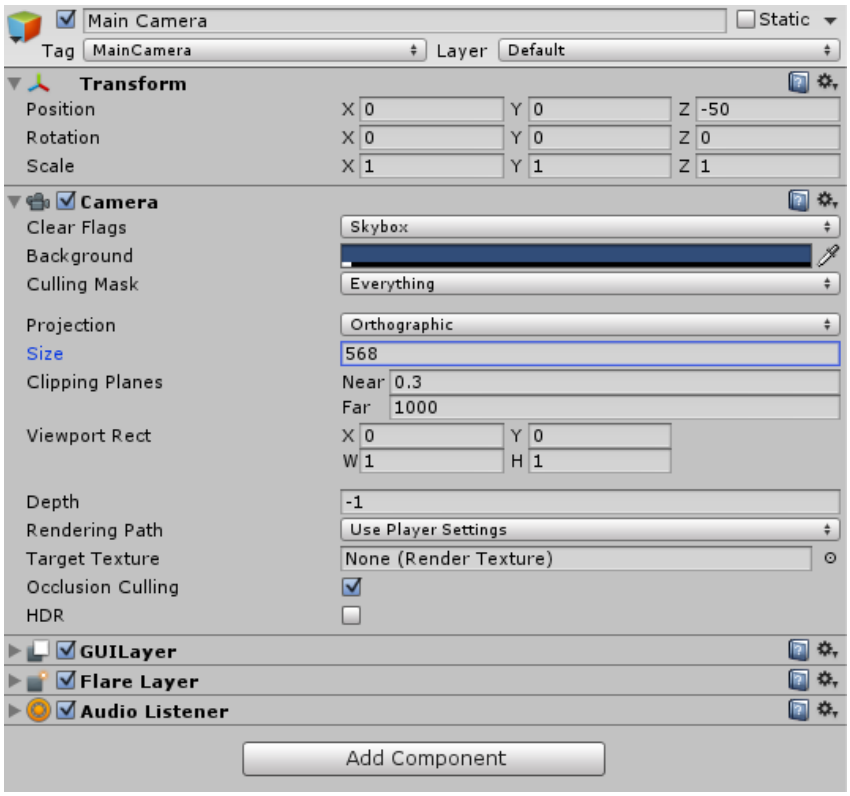

Figura 33. Opciones correctas de la cámara, para la visualización en 2D.

Adaptación del tamaño de cámara a la resolución de cada dispositivo

El videojuego está diseñado desde el principio para una proporción de pantalla de 16:9, pero muchos dispositivos tienen otras proporciones, por ejemplo 3:4, y al ajustar el tamaño de la proyección a la altura, el videojuego no se mostraría correctamente, quedando huecos en negro a los lados, o no mostrando la anchura total del videojuego en la pantalla, lo que sería inaceptable.

La solución para este problema es ajustar el tamaño de la proyección ortográfica de la cámara a la resolución horizontal de cada dispositivo, y luego ajustar la posición vertical de la cámara, ya que esta quedaría ajustada al borde superior del videojuego, cuando debe estar ajustada al borde inferior, ya que el panel con la información del jugador en la pantalla de juego está en la parte inferior.

```
var ratio:float=((640*1000)/(Screen.width));
var scrSize=(Screen.height*ratio/2000);
Camera.main.orthographicSize=scrSize;
Camera.main.transform.position.y=-(1136-(Screen.height*ratio/1000))/2;
```

Figura 34. Código de ajuste horizontal del tamaño de la proyección ortográfica.

De este modo, al usar diferentes proporciones de pantalla solo se puede perder área de visualización en la parte superior, lo que no afecta a la calidad del juego, únicamente los dispositivos con pantallas más largas podrán visualizar a los enemigos un poco antes.

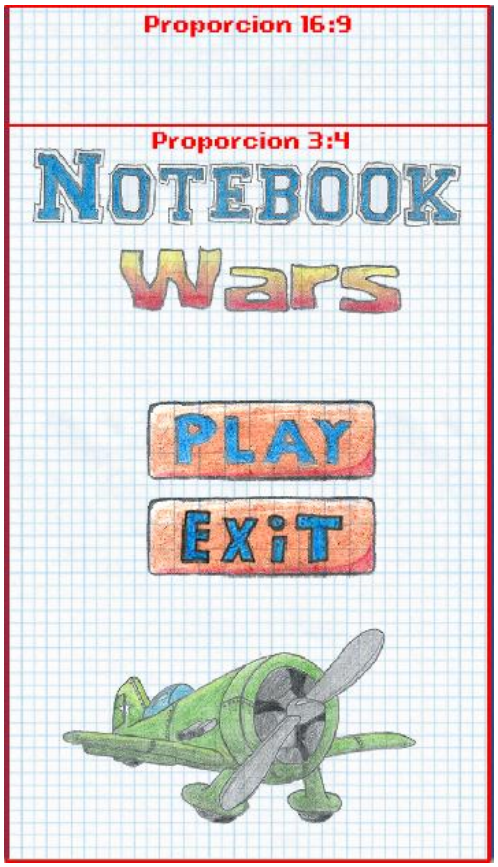

Figura 35. Área de visualización de las correspondientes proporciones.

Por esta razón en ningún menú hay ningún tipo de información vital ni elementos interactivos en la parte superior de la pantalla.

El nuevo problema que aparece con esto, es que Unity muestra el GUI (*Graphical User Interface*) en una capa independiente de la cámara, y este es el único medio en el que podemos mostrar textos generados por código en Unity.

La solución a este problema es obtener la escala a la que se debería redimensionar el GUI para mostrarse correctamente en la pantalla, y aplicarlo a su matriz de visualización en cada llamada a la función *onGUI*, antes de mostrar las etiquetas.

```
private function calculateGuiScale(){
    scale.x = Screen.width/640.0; // calculate hor scale
    scale.y = Screen.height/(Camera.main.orthographicSize*2);
    scale.z = 1;

    extraY=(Camera.main.orthographicSize*2)-960;//aqui se cal
}
GUI.matrix = Matrix4x4.TRS(Vector3.zero, Quaternion.identity, scale);
// draw your GUI controls here:

GUI.Label (Rect(460, 870+extraY, 200, 55), "Coins: "+mainClass.myHero.
GUI.Label (Rect(460, 915+extraY, 200, 55), "Life: "+mainClass.myHero.l
if(showWinScore==1){
    GUI.Label (Rect(200+winDi.transform.position.x, 465+extraY, 500, 5
    GUI.Label (Rect(200+winDi.transform.position.x, 505+extraY, 500, 5
}
// restore matrix before returning
GUI.matrix = svMat; // restore matrix
```

Figura 36. Código de ajuste del GUI.

Creación de los efectos de sonido

Los efectos de sonido del videojuego han sido creados con un software de uso libre bajo la licencia MIT de software libre, llamado sfxr. El programa en sí no es más que un generador aleatorio de sonidos de 8 bits, por lo que la calidad de los sonidos no es muy alta, pero es una forma rápida y gratuita de conseguir todo tipo de efectos de sonido para un videojuego. Al ser un juego con algo de estilo retro, este tipo de sonidos no le sientan mal al videojuego, y se integran perfectamente.

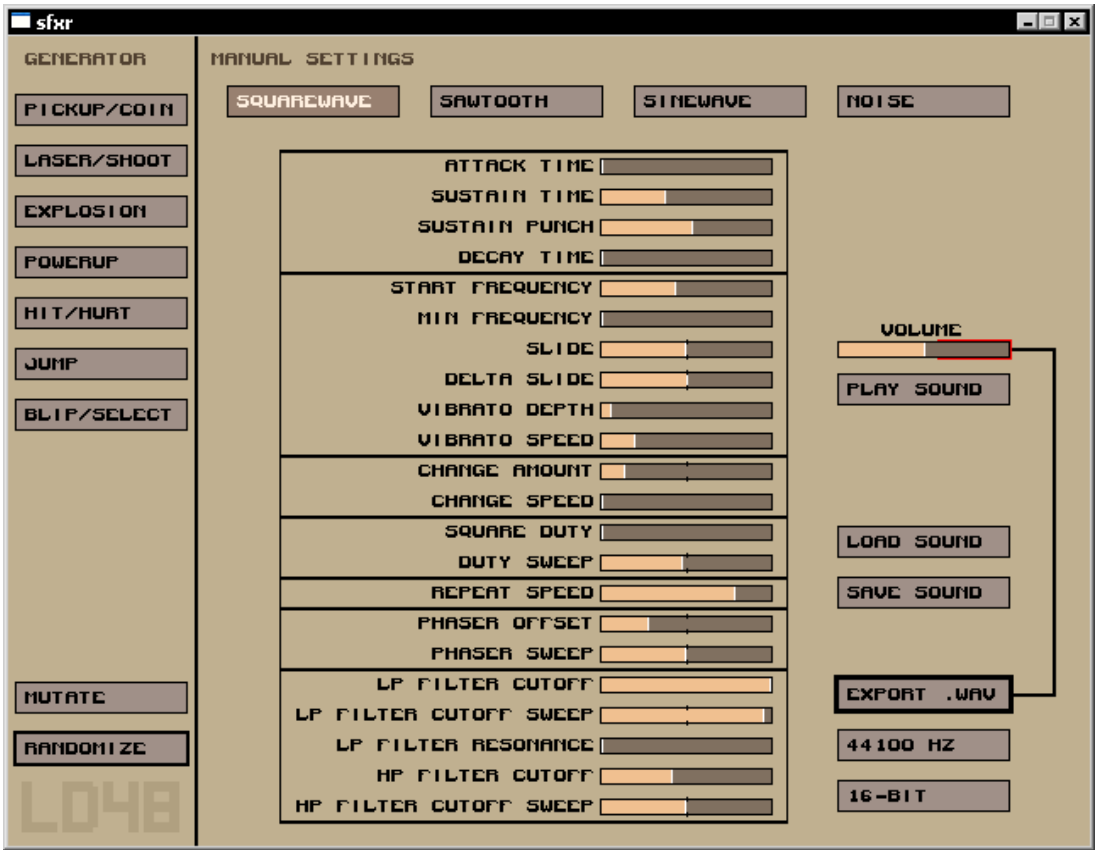

Figura 37. Pantalla principal del software sfxr.

El uso del programa es muy sencillo, ya que solo hay que pulsar en unos de los botones de la parte izquierda, dependiendo del tipo de sonido que se quiera generar, y los usuarios algo más avanzados pueden modificar todas las opciones para generar cualquier tipo de sonido.

Al obtener un sonido deseado, simplemente se pulsa en exportar .wav, y se generara el efecto de sonido deseado en formato .wav.

Conclusiones y resultados

El desarrollo de este TFG me ha permitido aplicar los conocimientos adquiridos en la carrera con el desarrollo de un sistema multidisciplinar como es el caso de un videojuego, en el que se necesitan tanto conocimientos matemáticos, como de algoritmia, diseño de interfaces, estructuras de datos, planificación de proyectos, etc.

Por lo que han sido de utilidad y se han puesto en práctica muchos de los conocimientos obtenidos en asignaturas como: Análisis Matemático, Algebra, Programación, Estructuras de datos y algoritmos, Interfaces persona computador, Ingeniería del software, Gestión de proyectos, Introducción a la programación de videojuegos, etc.

Análisis matemático y algebra han sido de gran utilidad para lidiar con los cálculos y la obtención de distintas formulas para el escalado de la aplicación en distintos dispositivos, o para cálculos trigonométricos de la inteligencia artificial de los enemigos.

Programación, estructuras de datos y algoritmos e ingeniería del software sin duda han sido asignaturas imprescindibles para disponer de los conocimientos necesarios para programar y estructurar el videojuego.

La asignatura de interfaces persona computador me ha permitido diseñar unas interfaces agradables y atractivas, sin llegar a saturar la pantalla de elementos.

La asignatura de gestión de proyectos ha sido de vital importancia para centrarme en las tareas criticas y planificar con antelación las tareas a realizar, consiguiendo de este modo terminar a tiempo.

Y finalmente en la asignatura de introducción a la programación de videojuegos he adquirido muchos conocimientos necesarios para desarrollar un videojuego, desde las motivaciones básicas que tienen las personas para jugar a videojuegos, hasta la gestión de los eventos o la inteligencia artificial.

El producto final obtenido ha sido bastante satisfactorio: un videojuego completo totalmente jugable. No obstante antes de proceder a la comercialización me gustaría implementar varias funcionalidades para las cuales el tiempo no ha sido suficiente, que se detallan en la sección de trabajo futuro.

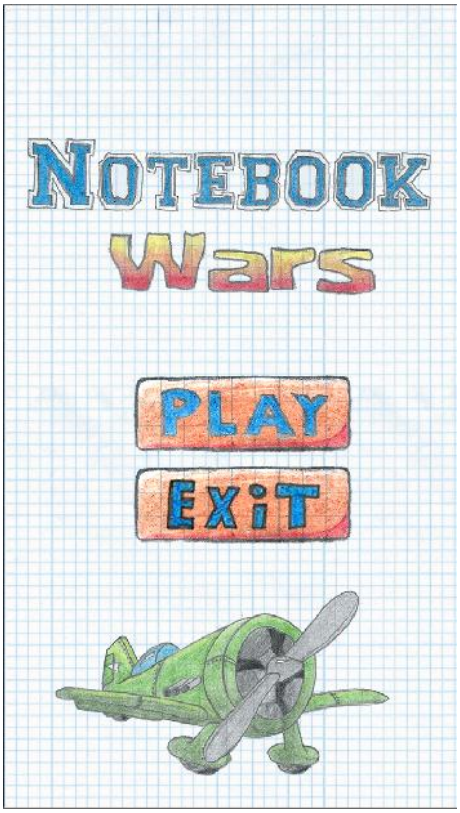

Figura 37. Captura del menú principal.

Figura 38. Captura de la pantalla de selección de nivel.

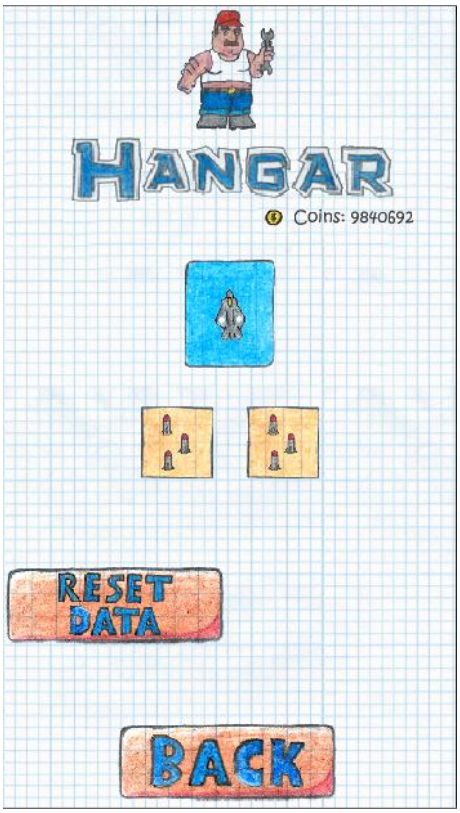

Figura 39. Captura de la pantalla principal del hangar.

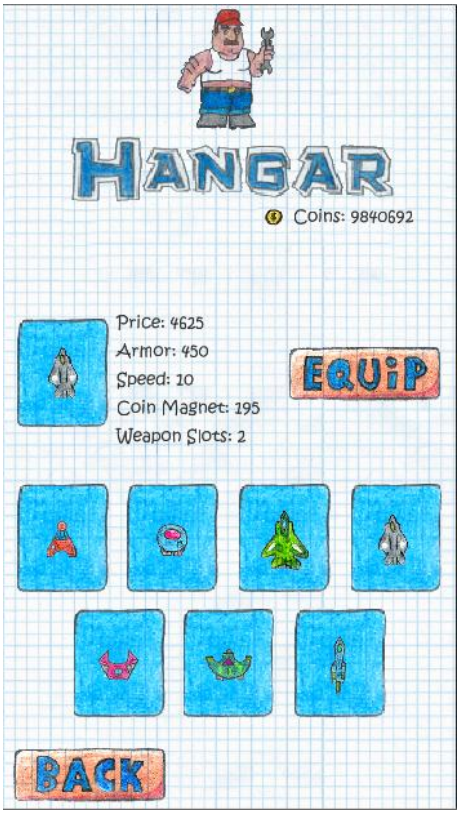

Figura 40. Captura de la tienda de aviones del hangar.

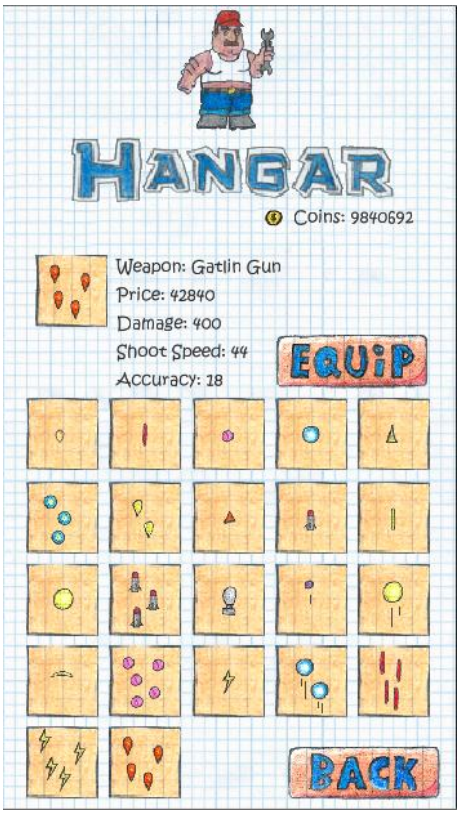

Figura 41. Captura de la tienda de armas del hangar.

Figura 42. Captura de la pantalla de juego.

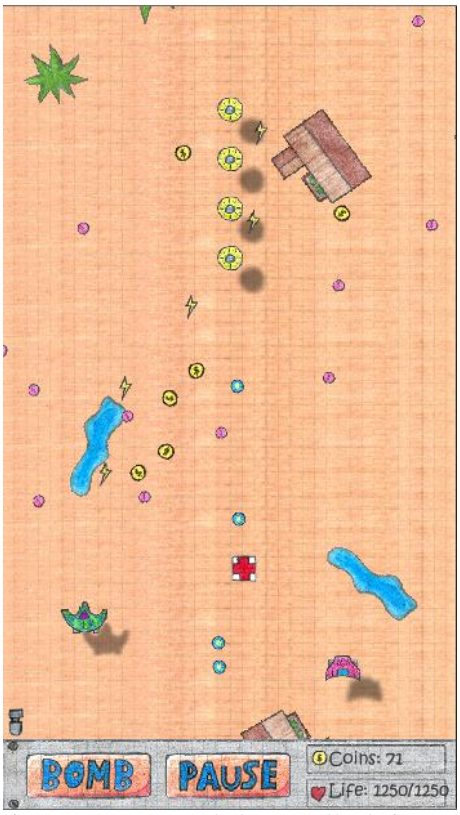

Figura 43. Captura de la pantalla de juego.

Figura 44. Captura de la pantalla de pausa.

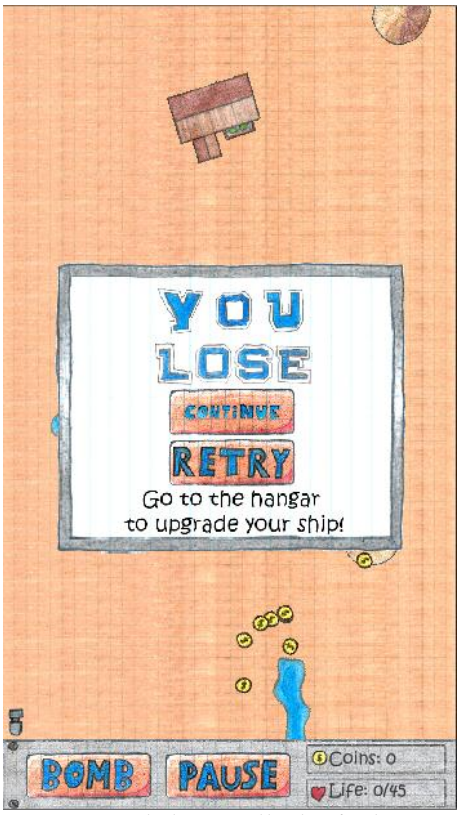

Figura 45. Captura de la pantalla de nivel no superado.

Figura 46. Captura de la pantalla de nivel superado.

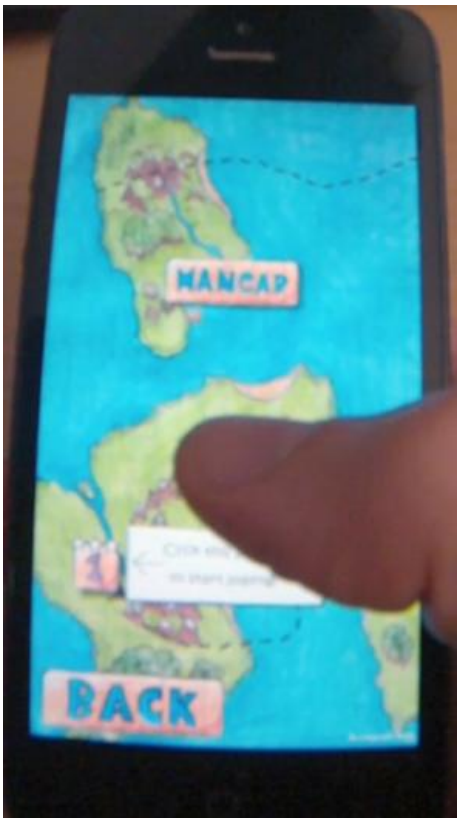

Figura 47. Juego ejecutándose en un iPhone 5.

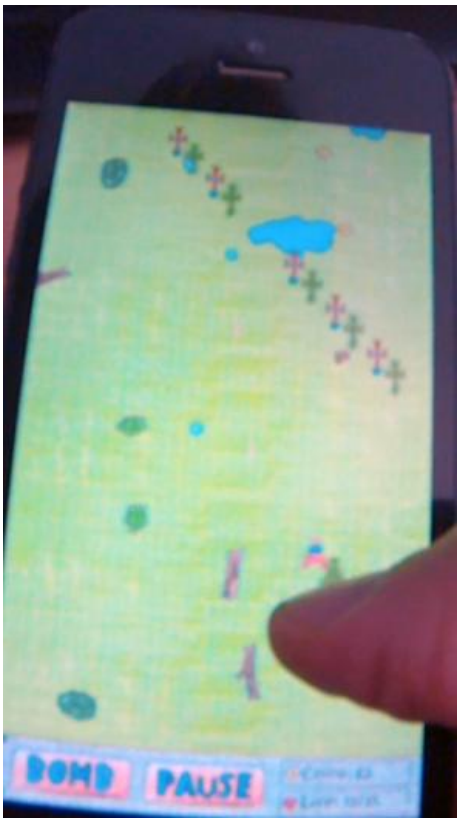

Figura 48. Juego ejecutándose en un iPhone.

Trabajo futuro

El resultado final del videojuego ha sido muy satisfactorio, no obstante, debido a la falta de tiempo han quedado muchas ideas sin poder realizar, por lo que más adelante me gustaría implementarlas, y así terminar el videojuego, obteniendo un producto mucho mas comercializable y con más posibilidades de éxito en los principales mercados de aplicaciones móviles.

Estas son las principales ideas y características que planeo implementar en el futuro:

- Jefes finales en ciertos niveles.
- Logros y medallas.
- Diferentes modos de juego (Misiones, Supervivencia, Extremo)
- Diferentes modos de dificultad (Fácil, Medio, Difícil)
- Armas especiales que solo se pueden comprar con dinero real (in-app purchases)
- Usar atlas de texturas para aumentar la eficiencia.

Agradecimientos

A mi familia, por todo el cariño y apoyo recibido estos años, en especial a mi madre, ya que sin su ayuda no hubiese logrado nada de esto.

A mis amigos y compañeros, por todos los buenos momentos que hemos compartido.

A Ramón, por haberme ofrecido su apoyo y ayuda para la realización de este proyecto.

A muchos de los profesores que he tenido estos años, que han sabido motivarme y despertar mi interés por algunos aspectos de la titulación en los que no tenia tanto.

Bibliografía

En este capítulo se muestra toda la información consultada, para la elaboración tanto del videojuego como de la documentación.

[1] «Mobile gaming could drive entire video game industry to $100B in revenue by 2017,» [En línea]. Available: • http://venturebeat.com/2014/01/14/mobile-gaming-could-drive-entire-game-industry-to-100b-in-revenue-by-2017/. [Último acceso: 7 Mayo 2014].

[2] «Unity Script Reference,» [En línea]. Available: http://docs.unity3d.com/ScriptReference. [Último acceso: 8 Junio 2014].

[3] «Mobile games market to double in size until 2016 and reach $23.9BN,» [En línea]. Available: http://www.gamesindustry.com/mobile-games-market-double-size-2016-reach-23-9bn/. [Último acceso: 7 Mayo 2014].

[4] «GanttProject,» [En línea]. Available: http://www.ganttproject.biz/. [Último acceso: 7 Mayo 2014].

[5] «The Entertainment Software Association - The Evolution of Mobile Games,» [En línea]. Available: http://www.theesa.com/games-improving-what-matters/mobile-games.asp. [Último acceso: 7 Mayo 2014].

[6] «Wikipedia - Independent video game development,» [En línea]. Available: http://en.wikipedia.org/wiki/Independent_video_game_development. [Último acceso: 7 Mayo 2014].

[7] «Essential Facts about the computer and videogame industry,» [En línea]. Available: http://theesa.com/facts/pdfs/ESA_EF_2013.pdf. [Último acceso: 7 Mayo 2014].

[8] «Breve, pero intensa, historia de los Shoot'em up (1 de 4),» [En línea]. Available: http://www.jugabilidad.com/breve-intensa-historia-shootem-up-1-4-365884. [Último acceso: 8 Mayo 2014].

[9] «Breve, pero intensa, historia de los Shoot'em up (3 de 4),» [En línea]. Available: http://www.jugabilidad.com/breve-intensa-historia-shootem-up-3-4-381817. [Último acceso: 8 Mayo 2014].

[10] «Breve, pero intensa, historia de los Shoot'em up (4 de 4),» [En línea]. Available: http://www.jugabilidad.com/breve-intensa-historia-shootem-up-4-4-438274. [Último acceso: 8 Mayo 2014].

[11] «Early Years - Shoot Em Up History,» [En línea]. Available: http://shmup.wikia.com/wiki/Early_Years. [Último acceso: 8 Mayo 2014].

[12] «Golden Age - Shoot Em Up History,» [En línea]. Available: http://shmup.wikia.com/wiki/Golden_Age. [Último acceso: 8 Mayo 2014].

«Evolution & Renaissance - Shoot Em Up History,» [En línea]. Available: http://shmup.wikia.com/wiki/Evolution_%26_Renaissance. [Último acceso: 8 Mayo 2014].

«Wikipedia - Toaplan Co.,» [En línea]. Available: http://en.wikipedia.org/wiki/Toaplan_Co.,_Ltd.. [Último acceso: 2014 Mayo 2014].

«15 Awesome Android Shoot 'Em Ups,» [En línea]. Available: http://www.tomsguide.com/us/pictures-story/507-Android-gaming-shooters-best-round-up-download.html. [Último acceso: 9 Mayo 2014].

«El Análisis DAFO: ¿Qué Es Y Cómo Llevarlo A Cabo?,» [En línea]. Available: http://www.innovacionsocial21.org/2013/07/el-analisis-dafo-que-es-y-como-llevarlo.html. [Último acceso: 9 Mayo 2014].

«Qué es un análisis CAME,» [En línea]. Available: http://mejorartucv.com/que-es-un-analisis-came-y-como-usarlo-en-la-busqueda-de-trabajo/. [Último acceso: 9 Mayo 2014].

«¿Qué es el análisis CAME?,» [En línea]. Available: http://blog.agencialanave.com/que-es-el-analisis-came/. [Último acceso: 2014 Mayo 2014].

«Monodevelop,» [En línea]. Available: http://monodevelop.com/. [Último acceso: 9 Mayo 2014].

«The Unity 3D Engine,» [En línea]. Available: https://unity3d.com/unity. [Último acceso: 9 Mayo 2014].

«Script Reference Unity 3D,» [En línea]. Available: http://docs.unity3d.com/ScriptReference/. [Último acceso: 27 Mayo 2014].

«Wikipedia - Class Diagram,» [En línea]. Available: http://en.wikipedia.org/wiki/Class_diagram. [Último acceso: 25 Mayo 2014].

«Wikipedia - Flowchart,» [En línea]. Available: http://en.wikipedia.org/wiki/Flowchart. [Último acceso: 25 Mayo 2014].

«Wikipedia - Programacion eXtrema (XP),» [En línea]. Available: http://es.wikipedia.org/wiki/Programaci%C3%B3n_Extrema. [Último acceso: 26 Mayo 2014].

«StackExchange - How to get dashed line in Photoshop?,» [En línea]. Available: http://graphicdesign.stackexchange.com/questions/235/how-to-get-dashed-line-in-photoshop. [Último acceso: 8 Junio 2014].

«ArrayPrefs2 - Clase para guardar Arrays en Unity,» [En línea]. Available: http://wiki.unity3d.com/index.php/ArrayPrefs2. [Último acceso: 9 Junio 2014].

«The top 5 things I've learned as a Unity developer,» [En línea]. Available: http://www.gamasutra.com/blogs/JohnWarner/20130910/194559/The_top_5_things_Ive_learned_as_a_Unity_developer.php?print=1. [Último acceso: 10 Junio 2014].

«Gliffy - Creacion de Diagramas Online,» [En línea]. Available:

[28] http://www.gliffy.com/. [Último acceso: 22 Junio 2014].

[29] «Making 2D games with Unity,» [En línea]. Available: http://www.third-helix.com/2012/02/05/making-2d-games-with-unity.html. [Último acceso: 15 Junio 2014].

[30] «Creating 2D Games With Unity3D Part 1,» [En línea]. Available: http://www.rocket5studios.com/tutorials/creating-2d-games-with-unity3d-part-1/. [Último acceso: 15 Junio 2014].

[31] «CreatePlane - Clase para crear planos en Unity,» [En línea]. Available: http://wiki.unity3d.com/index.php/CreatePlane. [Último acceso: 22 Junio 2014].

[32] «SFXR - 8bit sounds generator,» [En línea]. Available: http://www.drpetter.se/project_sfxr.html. [Último acceso: 23 Junio 2014].