Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

Conceptualización, Análisis, Diseño e Implementación de un Videojuego estilo retro para plataforma Android

Autor: Óscar Manrique Martínez

Tutor: Jorge Ruiz Magaña

Leganés, Julio de 2013

Agradecimientos

Por fin he concluido la carrera que tanto deseaba tener, pero el mérito no es únicamente mío y por eso quiero agradecerlo:

A mi madre, por su sacrificio y esfuerzo para que yo pudiera estudiar sin que me faltara nada.

A mi hermana Marta, por todo el trabajo que la he dado por mi perfeccionismo con el diseño gráfico.

A mi cuñado Luis, por ser mi tutor personal durante toda la carrera.

A mi tutor Jorge, por haberme dado la oportunidad de introducirme en el mundo de Android.

A mi familia, por ser pacientes conmigo durante todo este tiempo y por haberme apoyado.

A mis amigos, que desde la E.S.O ya me apoyaban para que estudiara aunque ellos ni siquiera habrían un libro.

A todas las personas que me conocen desde hace poco y me han apoyado en mis metas y siempre han tenido una sonrisa para mí.

A Paloma, la primera persona que conocí en la Universidad y con la que comencé esta aventura que tanto ansiábamos los dos. Gracias por todo lo que me has ayudado desde entonces y, gracias, por todo lo que nos queda por vivir juntos.

Resumen

A medida que la industria de los videojuegos avanza de forma inexorable, uno podría pensar que se corre el peligro de olvidar el pasado. Juegos como Gears of War 3 o RAGE llevan las expectativas gráficas y jugabilidad más allá, haciendo que los juegos clásicos parezcan excesivamente simples en comparación. Sin embargo, los videojuegos retro están teniendo gran auge en contra de lo que se podría esperar. Esto se debe a un público, entre los 30 y los 45 años, que recuerda con nostalgia su época de infancia o de adolescencia y mediante los videojuegos de estilo retro pueden rememorar esos momentos.

El público para el qué están enfocados los videojuegos retro favorece los aspectos comerciales, ya que es un sector de la población, en principio, con una estabilidad económica, dispuesta a abonar una pequeña cuantía de dinero por algunas funcionalidades adicionales.

Aunque servicios como la Consola Virtual de Wii y próximamente Wii U, Xbox Live Arcade o PlayStation Network nos permiten disfrutar de algún que otro viaje al pasado, la rápida expansión de los smartphones han tenido un importante papel en mantener el *retrogaming* vivo. La mayoría de estos dispositivos no tiene problemas para replicar el rendimiento de las máquinas de 8, 16, 32 y 64 bits. Además, al poder jugar con un dispositivo de tamaño reducido puedes disponer de él en cualquier lugar y momento.

Debido a esta necesidad del mercado, en el presente proyecto se llevará a cabo una prueba de concepto de un videojuego de estilo retro ambientado en el espacio, denominado, Mysterious Galaxy. La misión del jugador será destruir las naves enemigas y esquivar los proyectiles que lanzan contra él. Una de las posibilidades que incorpora el videojuego es la compra de armamento mediante una transacción con una cuenta Paypal.

Palabras clave: Videojuego, Android, teléfono móvil, Paypal, *smartphone, sprite,* colisiones, *retrogaming*.

Abstract

As the game industry progresses inexorably, someone might think that there is a danger of forgetting the past. Games like Gears of War 3 or RAGE have expectations beyond graphics and gamely, making classic games seem too simple in comparison. However, games are having retro boom against what you might expect. This is because an audience between 30 and 45 years, fondly recalling his days of childhood or adolescence and through retro-style games can relive those moments.

The audience for video games that focus favors retro commercial aspects, as it is a sector of the population, in principle, economic stability, willing to pay a small amount of money for some additional functionality.

While services like Virtual Console of Wii and Wii U soon, Xbox Live Arcade or PlayStation Network allow us to enjoy the occasional trip to the past, the rapid expansion of smartphones has had an important role in maintaining the retrogaming. Most of these devices do not have trouble replicating the performance of the machines of 8, 16, 32 and 64 bits. Furthermore, the power play with a small size device can dispose of it in any place and time.

In this project will be carried out a proof of concept of a retro-style game set in space, named, Mysterious Galaxy. The player's mission is to destroy enemy ships and dodge projectiles thrown at him. One possibility is that the game incorporates the purchase of weapons in a transaction with a Paypal account.

Keywords: Video game, Android, mobile phone, Paypal, smartphone, *sprite,* collisions, *retrogaming*.

Índice general

Índice de figuras

Índice de Tablas

Índice de código

1. Introducción y objetivos

Este capítulo pretende ser el punto de entrada al resto del documento, ofreciendo una visión general sobre el tema que nos ocupa y pudiendo encontrar en él los objetivos y motivaciones que me han llevado a realizar el presente Proyecto Fin de Carrera, además de una breve descripción de las fases que se han llevado a cabo para el desarrollo de dicho proyecto.

1.1 Introducción

La industria de videojuegos ha experimentado en los últimos años altas tasas de crecimiento, debido al desarrollo de la computación, capacidad de procesamiento, imágenes más reales y la estrecha relación entre películas de cine y los videojuegos, con lo cual los consumidores reconocen los títulos más pronto. En la década de 2000, los videojuegos han pasado a generar más dinero que la del cine y la música juntas. La industria del videojuego generó 51.137 millones de euros durante 2011 en todo el mundo. [1]

Los videojuegos llevan presentes muchos años en los terminales móviles; pero al igual que el resto de aplicaciones, en sus comienzos eran muy limitados. Un ejemplo es el clásico Snake, una serpiente que se alimenta de "puntos" y se va haciendo cada vez más grande hasta interceptar consigo misma.

Con la llegada de los *smartphone* las posibilidades de los videojuegos han crecido exponencialmente, lo cual hace que el producto sea más atractivo para los usuarios, llegando a ser uno de los sectores con mayores ingresos dentro de las aplicaciones móviles. Por estos motivos, en la actualidad algunas de las empresas dedicadas al desarrollo de videojuegos para las consolas portátiles como Electronic Arts están desarrollando videojuegos para teléfonos móviles. De hecho la situación va más lejos, ya que el éxito de algunos de los videojuegos para las plataformas móviles ha promovido la creación de PlayStation Mobile, una tienda virtual de Sony, con un catalogo de juegos compatible con PS Vita y terminales Android con certificación Playstation. [2]

La gran capacidad de procesamiento de los nuevos terminales móviles ha favorecido la creación de emuladores de consolas antiguas como la Game Boy o la Super Nintendo. El escaso tamaño de las pantallas hace que los detalles de los videojuegos más modernos no se puedan apreciar y en algunos casos se complica la jugabilidad. En cambio, los videojuegos de estilo retro tienen gráficos más simples que se adaptan perfectamente a las dimensiones de los smartphones, lo que ha ayudado a este tipo de juegos a ocupar una posición destacada. Además de estas características técnicas existe otro factor que ha favorecido a este tipo de juegos, los usuarios de mediana edad. El público de entre a los 30 y 45 años, son las primeras generaciones que disfrutaron durante su infancia de los videojuegos y el anhelo a esa época les hace ser el principal consumidor de videojuegos de estilo retro. Este sector del mercado es muy

apreciado desde el punto de vista comercial ya que debido a su estabilidad económica están dispuestos a mejorar su experiencia a cambio de abonar una pequeña suma de dinero.

Otro tipo de juegos que también se han beneficiados son los juegos online, ya que las nuevas tecnologías para la transmisión de datos (GPRS, EDGE, 3G) permite la interacción de los jugadores desde cualquier punto de una forma óptima, percibiendo así una buena experiencia.

1.2 Motivación

A continuación, se explicaran los motivos por los que decidí concluir mis estudios con el desarrollo de un videojuego para la plataforma Android.

Como estudiante y aficionado a los videojuegos siempre me atrajo la idea de programar uno, crear mi propio entorno de juego, idear sus reglas, en definitiva, que todo se ajustase a mi elección. Pero hasta la llegada de los smartphones el desarrollo de videojuegos había estado limitado para unas pocas empresas y no era sencillo acceder a esa posibilidad. Gracias a plataformas como Android, esta situación ha cambiado, dándome la oportunidad de cumplir una de mis metas profesionales.

Además, mi intención es emplear este proyecto como punto de inicio de mi carrera profesional. Mi interés en este sector nace de la gran cantidad de posibilidades que los smartphones proporcionan a los usuarios. Desde cualquier lugar, con tan solo utilizar este dispositivo de reducido tamaño puedes disfrutar de muchísimas funcionalidades que antes sólo eran posibles para un ordenador. Al ser una decisión tan importante para mí, también hay que conocer las posibilidades de futuro de este campo. Por eso, a continuación, se presenta las explicaciones por las que he llegado a la conclusión de un futuro próspero de esta tecnología, haciendo hincapié en el desarrollo de un videojuego de estilo retro para la plataforma Android.

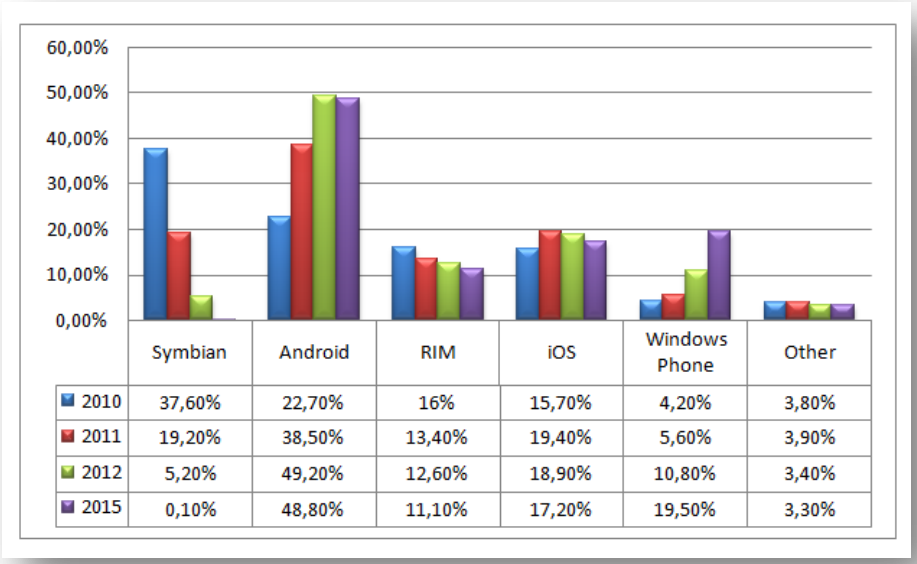
Hoy en día existen muchas personas aficionadas a los videojuegos clásicos, tal es esta situación que para las videoconsolas más modernas, como la Playstation 3 o la Wii, existen versiones de estos juegos. Una de las ventajas que ofrecen los smartphones frente a los videoconsolas es que el precio de los videojuegos, ya que es inferior en los smartphones. Además, la cantidad de títulos para la tecnología móvil está aumentando más rápidamente. Por otro lado, al disponer del juego en un dispositivo que siempre llevamos encima, tenemos la posibilidad de jugar en cualquier momento desocupado, como puede ser en el transporte público, a la espera de una persona o en los anuncios de la televisión. Estos motivos, junto con las características técnicas que permiten un perfecto funcionamiento de este estilo de videojuegos en los smartphones, hacen que se convierta en la principal plataforma para el *retrogaming*.

Observando el top por ingresos del Google Play en los 10 primeros puestos, aunque estos van cambiando constantemente, la mayoría son siempre videojuegos, lo que nos demuestra la importancia de éstos dentro del ecosistema smartphone.

Juego	Nº de descargas
Zynga Poker	50.000.000 – 10.000.000
Textas Poker	50.000.000 – 10.000.000
Live Holdem Poker Pro	50.000.000 – 10.000.000
Slotomania	5.000.000 – 1.000.000
Sygic: GPS Navigation	50.000.000 – 10.000.000
Slot City	5.000.000 – 1.000.000
Arcane Empires	1.000.000 – 500.000
Angry Birds	500.000.000 – 100.000.000
Top Eleven	5.000.000 – 1.000.000
Airport City	5.000.000 – 1.000.000

Tabla 1. Top 10 ingresos de las Aplicaciones Android (20/09/2012)

En Abril del 2011 la prestigiosa empresa Gartner (líder mundial en investigaciones de tecnologías de la información) publicaba que la previsión de dispositivos con Android supondría el 49.2 % de las ventas de smartphones, siendo así el sistema operativo líder. [3]



	Symbian	Android	RIM	iOS	Windows Phone	Other
2010	37,60%	22,70%	16%	15,70%	4,20%	3,80%
2011	19,20%	38,50%	13,40%	19,40%	5,60%	3,90%
2012	5,20%	49,20%	12,60%	18,90%	10,80%	3,40%
2015	0,10%	48,80%	11,10%	17,20%	19,50%	3,30%

Figura 1. **Previsión de las ventas mundiales de smartphones según su S.O**

Un año después publicaron las ventas del primer cuatrimestre del 2012, en las que las ventas de Android suponían un 56,1%, superando con creces las previsiones de la importante empresa. Esto resalta el impresionante crecimiento que está experimentando y la gran oportunidad que supone desarrollar aplicaciones para una plataforma con semejante número de usuarios. [4]

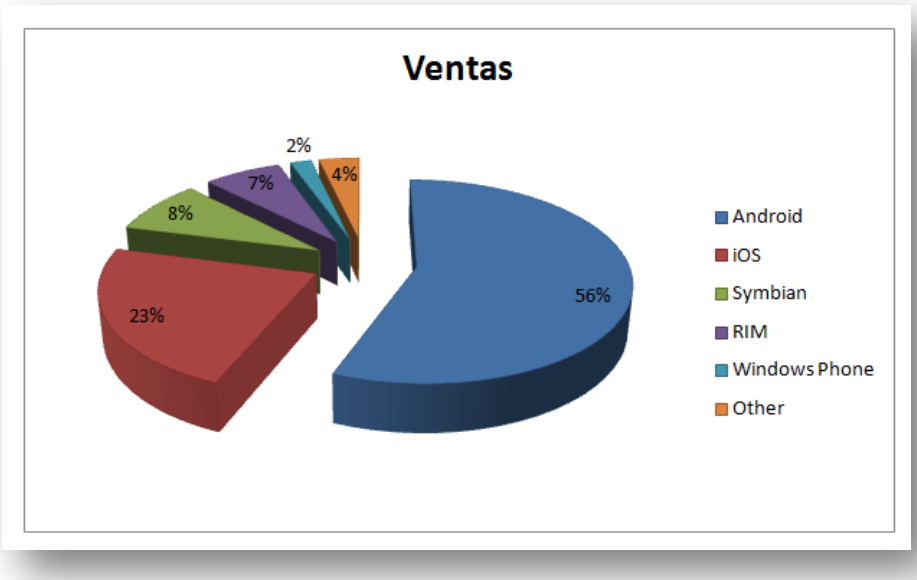


Figura 2. **Ventas mundiales de smartphones en el primer cuatrimestre de 2012 según su S.O**

Las funcionalidades que ofrecen los smartphones (conexión a internet, geolocalización, cámara,..), y en concreto las aplicaciones nativas, hacen que la demanda sea cada vez mayor, por lo que es un sector unas grandes expectativas de futuro. Asimismo, hay una ventaja muy importante en el desarrollo de aplicaciones móviles: los puntos de venta centralizados que ofrecen las plataformas a los desarrolladores. No es lo mismo crear una pequeña utilidad y publicarla en cualquier web no especializada, que servir ese mismo contenido en el App Store o el Google Play. Estos dos lugares ofrecen un punto de conexión entre desarrollador y posibles clientes que nunca hasta ahora se había tenido en la industria del software.

De todo lo explicado anteriormente podemos concluir el magnífico futuro que se espera de los smartphones y que dentro de estos se prevé que se sitúe en cabeza la plataforma Android. También se demuestra que la mayoría de los usuarios utilizan el terminal como centro de ocio por lo que existen grandes posibilidades de ingresos en el ámbito de los videojuegos y que sus características han favorecido a los videojuegos de estilo retro haciendo del presente proyecto un buen punto de partida para una carrera profesional.

1.3 Objetivos

El principal objetivo del presente Proyecto Fin de Carrera es desarrollar un videojuego, llamado Mysterious Galaxy, de estética y funcionalidad retro enfocado desde un punto de vista comercial. Por ello durante todo el proceso de creación del videojuego se han tenido presentes las preferencias que podría tener el jugador final, implementando algunas funcionalidades disponibles previo pago, así como facilitando la forma del mismo de un modo sencillo y cercano al usuario como es *Paypal*.

El objetivo secundario consiste en el estudio de la plataforma Android centrándose en la programación de videojuegos. Para ello, se analizará la arquitectura de Android y el ciclo de vida de las aplicaciones bajo el enfoque de los videojuegos, por lo que se tendrá muy en cuenta características tales como la velocidad de interacción, la optimización de los recursos , el empleo de hilos, la pantalla táctil o el uso de bases de datos SqlLite.

Al elaborar este proyecto se ha considerado la posibilidad de que pueda significar un punto de partida para futuros alumnos, de manera que puedan empezar a crear sus videojuegos tomando como base este proyecto.

1.4 Fases del desarrollo

En este apartado se comentan las distintas fases que conforman la elaboración de este proyecto.

- **Plataforma Android**: Estudio y documentación de la plataforma Android en cuanto a sus posibilidades para el desarrollo de videojuegos.
- **Conceptualización**: Concepción de las bases del videojuego para obtener el estilo retro deseado, las reglas del juego y las soluciones para los objetivos comerciales marcados.
- **Análisis**: Ha sido necesario evaluar exhaustivamente los requisitos necesarios para la elaboración de este proyecto además de tener en cuenta las funcionalidades que se deseaban desarrollar.
- **Diseño**: Se ha realizado valorando qué se pretendía lograr en cada una de las pantallas y como se quería mostrar al usuario, de manera que sirviera de guía para la etapa posterior.
- **Implementación**: Después de cubrir las etapas anteriores, se desarrolla el videojuego, intentando maximizar la optimización de los recursos. Esta fase se divide en subetapas bien diferenciadas que finalmente se entrelazarán para formar el conjunto. Dichas subetapas son:
    1. Interfaces de usuario de cada *Activity*.
    2. Lógica del primer nivel del videojuego.
    3. Incorporación de la base de datos y adaptación del código para la incorporación de los niveles restantes.
    4. Interfaz y lógica de la tienda e implementación de las armas especiales.
    5. *App-In-Purchase*.
- **Pruebas**: Durante el desarrollo de la aplicación se han llevado a cabo pruebas en los diferentes módulos para garantizar que está exento de errores.

2. Estado del arte

En este bloque, se pretende proporcionar una visión general de las tecnologías que se han empleado o están relacionados con este proyecto. De este modo la comprensión será más cómoda para lectores no familiarizados con este entorno de trabajo.

En el primer apartado se expondrá los videojuegos retro y sus emuladores y la evolución que han sufrido los videojuegos desde su aparición en el año 1952 hasta su situación actual con la aparición de plataformas como Android. Posteriormente se hará referencia a los dispositivos móviles y los smartphones. Finalmente, se hará un análisis de las diferentes alternativas de sistemas operativos para dispositivos móviles, entrando más en profundidad en el elegido para este proyecto.

2.1 Videojuegos

2.1.1 Retrogaming

El *retrogaming*, también conocido como *old-school gaming*, es la afición de jugar y coleccionar viejos PC, consolas y videojuegos arcade. Para el uso de estos juegos antiguos no es necesario poseer el hardware original si no que también se puede disfrutar de ellos mediante un hardware más moderno atreves de la emulación.

Los juegos retro más populares son los que se producen en torno a las décadas de 1980 y 1990, e incluyen títulos para las consolas como la Atari 2600 , Nintendo Entertainment System , Sega Master System , Mega Drive o para las maquinas arcade, especialmente de los primeros juegos de Komani, Sega, Atari o Taito.

A raíz de la creciente nostalgia y el éxito de retro-compilaciones en las consolas de sexta y séptima generación, el *retrogaming* se ha convertido en un motivo de los juegos modernos. Estos juegos imponen limitaciones en la paleta de color, la resolución y la memoria siendo estas muy por debajo de los límites actuales del hardware con el fin de imitar el aspecto de hardware antiguo. Algunos de los títulos con estas características son Mega Man 9, Retro Game Challenge y Fantasy Zone II. [5]

2.1.2 Emuladores para Android

Las características actuales de los dispositivos móviles permiten que se pueden ejecutar sobre ellas diferentes emuladores que acercan a los usuarios lo juegos clásico que añoran. Algunos de estos emuladores para la plataforma Android son:

SNES9X EX

SNES9X EX replica de forma muy precisa la consola Super Nintendo Entertainment System de 16 bits. Algunos de los grandes clásicos, como Super Castlevania IV, Zelda: A Link to the Past o Chrono Trigger, se pueden jugar sin

ningún problema. Este emulador es tan completo que incluso soporta los juegos que llevaban chip SuperFX, como Star Fox o Super Mario World 2: Yoshi's Island. Debido a una denuncia reciente, SNES9X EX se eliminó del Google Play, pero se puede descargar de forma totalmente gratuita en la página de su desarrollador.

Tiger Arcade

Tiger Arcade no emula todas las recreativas pero si es capaz de hacerlo con algunos títulos sorprendentemente complejos y no tiene problemas para emular la mayoría de títulos Neo Geo MVS y CPS2 de Capcom. Po lo que se puede disfrutar de joyas como Metal Slug, King of Fighters, Street Fighter Alpha o DoDonPachi en la pantalla de tu móvil. Se pueden conseguir desde el SlideMe Market.

FPse for Android

Aunque Sony recientemente ha sacado PlayStation Mobile, también se puede obtener juegos de 32 bits descargando este conseguido emulador. FPse ofrece un rendimiento casi perfecto y es compatible con una larga lista de títulos de PlayStation. Incluso juegos exigentes como Gran Turismo funcionan a velocidad completa, aunque necesitarás un teléfono que tenga un procesador de al menos 1GHz para conseguir dichos resultados.

N64oid

Nintendo 64, claramente la máquina más avanzada emulada actualmente en Android, fue el hogar de clásicos como Super Mario 64, The Legend of Zelda: Ocarina of Time o F-Zero X. N64oid te permite emular estos y muchos otros juegos. El único inconveniente es que el rendimiento y la compatibilidad no son muy altos, algunos juegos funcionan mejor que otros, mientras que algunos ni siquiera cargan. N64oid está disponible en el SlideMe Market. [6]

2.1.3 Los primeros videojuegos

Los videojuegos tienen su origen en la década de 1940 cuando, tras el fin de la Segunda Guerra Mundial, se construyeron las primeras supercomputadoras programables como el ENIAC, de 1946. Los primeros intentos por implementar programas de carácter lúdico (inicialmente programas

de ajedrez) no tardaron en aparecer, y se fueron repitiendo durante las siguientes décadas. [7]

En 1952, Alexander “Sandy” Shafto Douglas presenta su tesis de doctorado en la Universidad de Cambridge (Inglaterra) basada en la interactividad entre seres humanos y computadoras. Dicha tesis incluía una versión computarizada del tres en raya llamada *OXO*. Éste sería el primer juego de computadora en usar una pantalla gráfica digital, por lo que al mostrar gráficos, tenía todos los requisitos para poder ser considerado como el primer videojuego de la historia. [8]

Figura 3. **OXO, de Alexander Douglas (1952).**

Después de todos estos primeros pasos, llegó el momento para *Tennis for Two*, el considerado oficialmente como primer videojuego de la historia por las discrepancias existentes con el *OXO*. Fue el 18 de Octubre del año 1958 cuando William Higinbotham, un físico estadounidense, presentó un programa que simulaba un partido de tenis a través de un osciloscopio del Laboratorio Nacional de Brookhaven (Upton New York, Long Island). Se presentaba una visión lateral de la pista con una red en medio y líneas que representaban las raquetas de los jugadores.

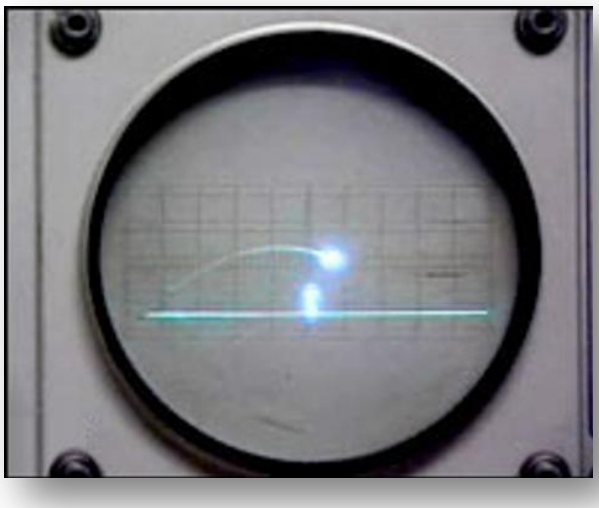


Figura 4. **Tennis for Two, de William Higinbotham.**

Después de la creación de algunos juegos y la aparición de las máquinas recreativas, sin mucho éxito en el mercado, nació la primera videoconsola. Esta apareció en 1972, gracias a Ralph Baer considerado el padre de los videojuegos y fue bautizada como *Magnavox Odyssey*. Contaba con un total de 28 juegos diferentes y dado su reducido hardware, carecían de sonido y los jugadores debían memorizar sus puntuaciones. [9]

Figura 5. ***Magnavox Odyssey*, de Ralph Baer**

2.1.4 Las videoconsolas portátiles

En la década de los 80', el mercado de los videojuegos se expandía a pasos acelerados. Mientras una división de Nintendo trabajaba arduamente en su proyecto de crear una consola para el hogar, otra división más modesta y con menos recursos que la primera, daba un golpe tremendo a los mercados del entretenimiento al introducir la primera consola portátil del mercado, la Game & Watch.

Su creador Gunpei Yokoi lanzó más de 59 juegos distintos en este formato desde 1980 hasta 1988, una producción inmensa e increíble, teniendo en cuenta que cada juego necesitaba el desarrollo de su propio Hardware. Algunos de los títulos del formato Game & Watch fueron 'Donkey Kong', 'The Legend of Zelda', 'Mario Bros', 'Mickey Mouse', y 'Balloon Fight'. [10]

Figura 6. **Game & Watch, de Gunpei Yokoi**

El dos de abril de 1989 fue la fecha escogida por Nintendo para el lanzamiento de una consola que trascendería décadas por su fama y comodidad a la hora del juego. La Game Boy salió al mercado para imponerse como la reina de las videoconsolas portátiles. Gunpei Yokoi corrigió con esta consola el error de las Game y Watch. Ya no era necesario crear un hardware para cada juego, ahora venían en formato cartucho y cualquier empresa podría programar los suyos propios y trasladarlos al mercado sin mayor complejidad. Entre el modelo clásico y la versión con Color, se llegaron a vender 118,69 millones de unidades. [11]

Figura 7. **Game Boy de Nintendo**

No sólo Nintendo sacó al mercado consolas portátiles, también salieron: la Game Gear (Sega), LYNX (Atari), LYNX II (Atari), Nomad (Sega), Neo Geo Pocket (SNK)… Pero ninguna tuvo un éxito comparable con los diferentes modelos la Game Goy.

En el año 2004 Nintendo decide sacar del mercado la marca Game Boy y lanza la Nintendo DS (*Dual Screen*), una consola que posee dos pantallas LCD, una táctil para crear una experiencia nueva de juego. Otra de sus ventajas es el wireless integrado, esto permitía jugar de dos a ocho jugadores sin la necesidad de cables hasta en una distancia máxima de 20 metros. Incluye a su vez WI-FI lo que permite jugar a través de Internet hasta cuatro jugadores.

Figura 8. **Nintendo DS**

Sony lanza en forma de respuesta a la Nintendo DS, la PSP, que en sus siglas quiere decir Playstation Portable. Su apuesta fue el centro multimedia portátil, reproductor de video, audio y fotos, que se convertiría en más que en una consola de juegos. Su sistema de datos es un disco UDM (Universal Media Disc), el formato esta patentado sólo por Sony por lo cual su desarrollo se limita a esta empresa. Su capacidad es de 1.8GB, y la PSP ofrece soporte para *memory stick*. [12]

Figura 9. **PSP de Sony**

En la actualidad existen las versiones mejoradas de estas dos consolas:

- **Nintendo 3DS**: Es la primera consola en 3D y no es necesario la utilización de gafas especiales.
- **PS Vita**: Incorpora conexión mediante 3G, pantalla táctil, panel táctil trasero, sensor de movimiento…

2.1.5 Videojuegos en Android

Los juegos para teléfonos móviles han ido evolucionando conforme lo hacían los terminales. En sus inicios se caracterizaban por ser juegos de un sólo color, reproducir sonidos polifónicos, estar almacenados en la memoria ROM del móvil y utilizar la pulsación del teclado. Con el paso del tiempo, estos clásicos han dado paso a los juegos en color, con sonidos multimedia, posibilidad de descargarlos de internet, almacenarlos en memorias externas, interacción mediante pantallas táctiles y otras muchas posibilidades.

La empresa Gameloft desarrolló algunos de los primeros videojuegos en blanco y negro que resultaron ser un éxito, poniendo así de manifiesto el potencial de los videojuegos en los terminales móviles. Pero no es hasta la

época en la que nos encontramos cuando se origina una gran revolución en el sector con la creación de nuevos sistemas operativos y dispositivos cada vez más potentes, que permiten ejecutar videojuegos de gran complejidad.

Actualmente el mercado de los videojuegos para móviles es más grande que cualquier otro mercado de videojuegos portátiles, obteniendo cifras de ventas muy elevadas. Según un estudio de Futuresource Consulting, los juegos móviles arrojarán beneficios por valor de más de 10.000 millones de dólares para 2014. Además, según sus previsiones el 95% de las ganancias totales de las aplicaciones se deberán a juegos, sin contar los contenidos *premium* adquiridos mediante aplicaciones gratuitas. [13]

Tipos de videojuegos

Tras diversos estudios, se ha observado que los videojuegos son el tipo de aplicación favorita por parte de los usuarios, pero dentro de este tipo, existen variedades de géneros, que son:

- **Arcade y acción**: Implica manipular un personaje a través de una serie de niveles de dificultad progresiva; pero no se produce el desarrollo de una historia. Por ejemplo: *Angry Birds, Grand Theft Auto III, Doodle Jump, Fruit Ninja*…
- **Carreras**: Ambientados en la competencia de coches, motocicletas y cualquier vehículo. Por ejemplo: *Drag Racing, Asphalt 7: Heat, Racing Thunder, GT Racing*…
- **Casuales**: Juegos dirigidos o utilizados por gran número de jugadores siendo éstos ocasionales. Pueden pertenecer a cualquier género y se caracterizan por sus sencillas reglas. Por ejemplo: *Jewels, Sentinel 3: Homeworld, Air Control, Monopoly*…
- **Deportes**: Género muy demandado en todas las plataformas de videojuegos. Títulos Por ejemplo: *FIFA 12, Virtual Tennis, NBA JAM*…
- **Juegos de cartas y casino**: Son los juegos que obtienen más ingresos debido a la posibilidad de realizar apuestas. Por ejemplo: *Live Holdem Poker Pro, Solitaire, Phase 10*…
- **Puzzles y juegos para ejercitar la mente**: En ellos se exige al jugador cierta habilidad mental además de incorporar problemas de lógica, estrategia… Por ejemplo: *World of Goo, Apalabrados, Cut the Rope*…

2.2 Dispositivos móviles

Un dispositivo móvil se puede definir como un aparato de reducido tamaño, con algunas capacidades de procesamiento, con conexión permanente o intermitente a una red, con memoria limitada, que ha sido diseñado específicamente para una función concreta, pero es capaz de llevar a cabo funciones adicionales. [14]

Entre todos los dispositivos móviles, el que más sobresale por encima de los demás debido al número de usuarios es el teléfono móvil. Según los datos de la Comisión del Mercado de Telecomunicaciones a fecha de Diciembre del 2011, en España el número de líneas es de 56.189.000, lo que supone que la media de teléfonos móviles por persona es superior a un teléfono por habitante, es decir, se habla de más líneas que personas. En cifras, esto se traduce en 121,7 líneas por cada 100 habitantes. [15]

2.2.1 Smartphone

Dentro de los dispositivos móviles, un *smartphone* (cuya traducción en español sería "teléfono inteligente") es una evolución del teléfono móvil tradicional que cuenta con ciertas características y prestaciones que lo hacen más cercano a un ordenador personal alejándolo de la idea de un teléfono tradicional.

Entre dichas características, se puede encontrar una mejora en la capacidad de proceso y almacenamiento de datos, conexión a Internet mediante Wi-Fi, pantalla táctil, acelerómetro, geolocalizador (GPS), teclado QWERTY y diversas aplicaciones de usuario como navegador web, cliente de correo, aplicaciones ofimáticas, reproductores de vídeo y audio, etc. incluyendo la posibilidad de descargar e instalar otras nuevas.

A pesar de estas importantes mejoras con respecto a sus predecesores móviles, el reducido tamaño de los smartphones conlleva inexorablemente limitaciones de hardware que los mantienen claramente diferenciados de los ordenadores convencionales. Estas limitaciones se reflejan principalmente en pantallas más pequeñas, menor capacidad del procesador, restricciones de memoria RAM, de almacenamiento persistente y de consumo de energía.

2.3 Sistemas operativos para dispositivos móviles

Un Sistema Operativo (SO) es el software básico de una computadora que provee una interfaz entre el resto de programas del ordenador, los dispositivos hardware y el usuario. Las funciones básicas son administrar los recursos de la máquina, coordinar el hardware y organizar archivos y directorios en dispositivos de almacenamiento. En el caso de los S.O móviles la diferencia radica en su orientación a la ''conectividad inalámbrica', los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos. [16]

2.3.1 Cuota de mercado

Antes de entrar en el análisis de los principales sistemas operativos para teléfonos móviles se presentarán dos graficas de la cuota de mercado de éstos en España y en el mundo.

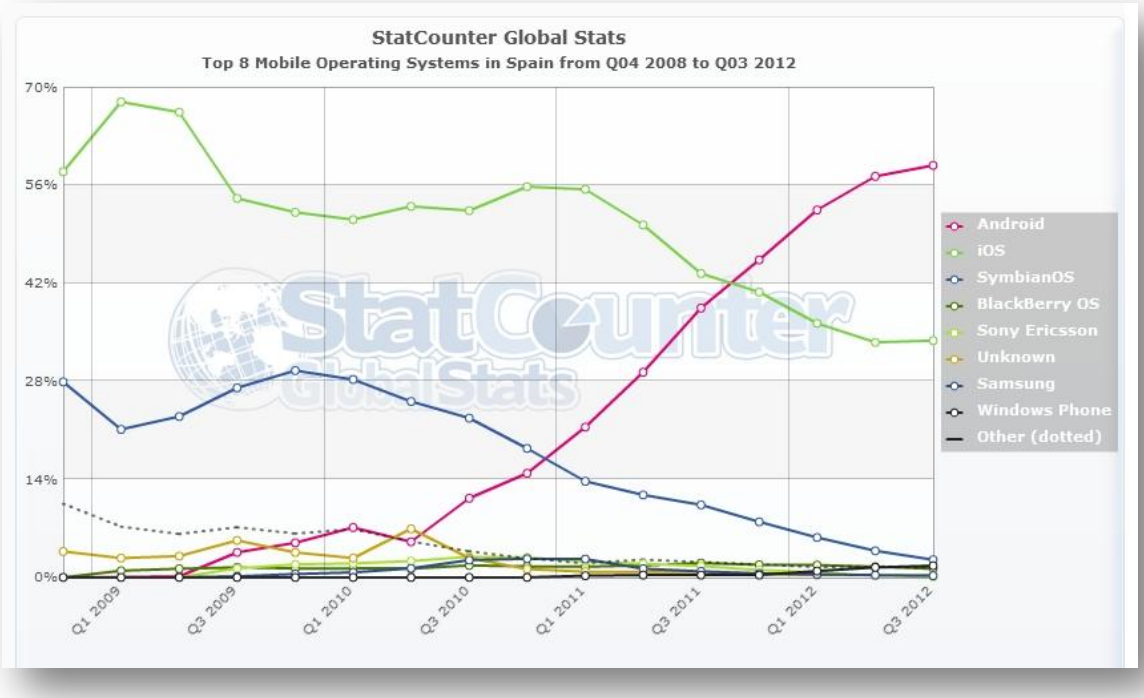


Figura 10. **Cuota de mercado de S.O. móvil en España**

En esta gráfica se muestran los datos de la cuota de mercado de los diferentes sistemas operativos en España durante los últimos tres años. Se puede observar como la cuota de la plataforma Android ha conseguido más del 50% en tan sólo dos años y medio, situándose así como sistema operativo líder en España.

También cabe destacar el importante descenso que ha sufrido el S.O. de Samsung que en apenas tres años ha descendido de un 28 % a un 1 %. Esto es debido a que Samsung ha optado por modernizarse, sustituyendo en sus teléfonos móviles su sistema operativo por Android. Ante este panorama podemos extraer la conclusión más importante: el mercado está prácticamente dominado en todo momento por dos S.O en 2009 por IOS y Samsung y en la actualidad por IOS y Android.

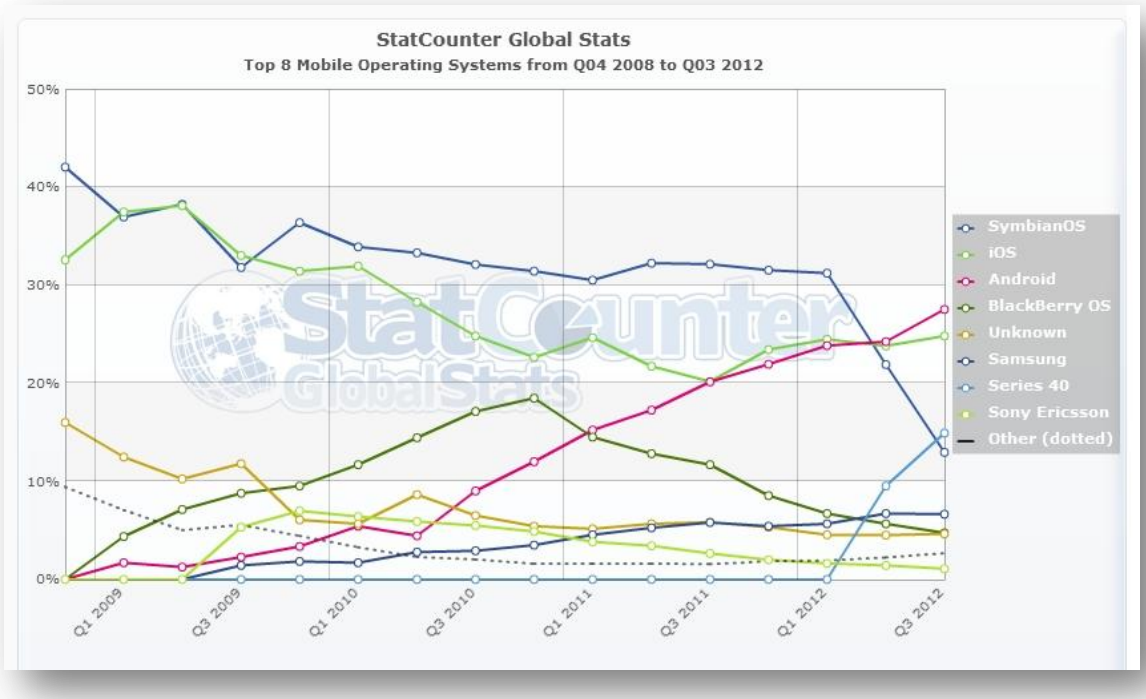


Figura 11. **Cuota de mercado de S.O. móvil en el mundo**

La figura anterior corresponde con la cuota de mercado de los S.O en todo el mundo desde 2009 hasta la actualidad. Los datos destacables en esta gráfica son: el gran descenso en 2012 por Symbian OS ha dejado de ser el líder del mercado para pasar al cuarto puesto, la lucha que existe entra IOS y Android desde 2011 por hacerse con más cuota y convertirse en líder y el crecimiento de Series 40 que en tan sólo seis meses se ha colocado en tercer lugar superando a la gigante Symbian OS.

Analizando la situación entre España y el resto del mundo llama la atención que el líder mundial durante un largo periodo de tiempo, Symbian OS, no tiene una cuota significativa en España. Además, mientras que en el mundo hay una gran competencia entre cinco S.O, en España esta competencia únicamente existe entre dos. Finalmente, en ambas gráficas el líder actual es Android aunque con una diferencia muy elevada en la cuota, en España 60% y en el mundo con un 28%.

2.3.2 iOS

iOS es un sistema operativo para móviles desarrollado por Apple en 2007. Originalmente fue nombrado el iPhone OS, pero fue renombrado a iOS en junio de 2009. El iOS actualmente se ejecuta en el iPhone, iPod touch y iPad.

La arquitectura iOS está basada en capas, donde las más altas contienen los servicios y tecnologías más importantes para el desarrollo de aplicaciones, y las capas más bajas controlan los servicios básicos. [17]

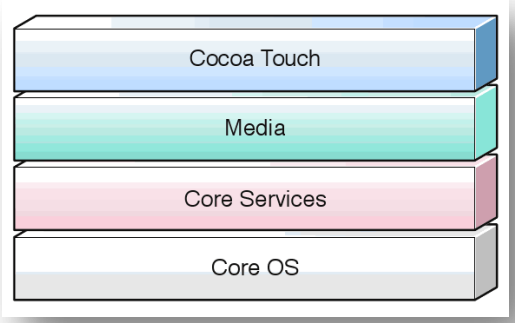


Figura 12. **Arquitectura de capas iOS**

2.3.3 Symbian OS

Symbian OS es un sistema operativo que fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia, Sony Ericsson, PSION, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp, etc. Sus orígenes provienen de su antepasado EPOC32, utilizado en PDA's y Handhelds de PSION. [18]

El objetivo fue crear un sistema operativo que pudiera competir en su momento con el de Palm o el Windows Phone de Microsoft y actualmente además con Android de Google, iOS de Apple y BlackBerry OS.

2.3.4 BlackBerry OS

El BlackBerry OS es un sistema operativo móvil desarrollado por Research In Motion(RIM) para sus dispositivos BlackBerry. El sistema permite multitarea y tiene soporte para diferentes métodos de entrada adoptados por RIM para su uso en computadoras de mano, particularmente la *trackwheel*, *trackball*, *touchpad* y pantallas táctiles. El SO BlackBerry está claramente orientado a su uso profesional como gestor de correo electrónico y agenda. Desde la cuarta versión se puede sincronizar el dispositivo con el correo electrónico, el calendario, tareas, notas y contactos de *Microsoft Exchange Server*. Por otro lado es compatible también con *Lotus Notes* y *Novell GroupWise*. [19]

2.3.5 Windows Phone 7

Windows Phone es un sistema operativo móvil desarrollado por Microsoft, como sucesor de la plataforma Windows Mobile. Está pensado para el mercado de consumo generalista en lugar del mercado empresarial por lo que carece de muchas funcionalidades que proporciona la versión anterior.

Una de sus más aclamadas novedades es la interfaz de usuario, denominada “Metro”. Dicha interfaz se basa en unos simples, pero efectivos, mosaicos dinámicos que muestran información útil al usuario. [20]

2.4 Plataforma Android

A continuación, se va a proceder a presentar los principales conceptos relacionados con el SO Android. Dicha sección es necesaria para aprender las nociones básicas de Android y entender cómo estas se relacionan entre sí, un paso fundamental antes de diseñar e implementar una aplicación.

La plataforma Android se caracteriza principalmente por el hecho de ser código libre, lo que posibilita que todo aquel que posea conocimientos para ello pueda cambiar y rehacer a su medida el código de las aplicaciones. Esto hace que Android sea un SO muy completo y con miles de aplicaciones, siendo la mayoría gratis o muy baratas y encontrándose todas ellas en un mercado virtual llamado Play Store.

2.4.1 Historia

En el año 2007 se funda la OHA (Open Handset Alliance), un grupo de más de 80 empresas tecnológicas en las que están incluidos fabricantes de hardware, distribuidores de dispositivos móviles y desarrolladores de software. Entre las compañías más importantes de este consorcio cabe destacar las empresas de telecomunicaciones Motorola, HTC, T-Mobile y Qualcomm. La OHA representa, según sus palabras, un compromiso de conseguir de forma unida la aceleración de la innovación en los móviles y ofrecer a los consumidores experiencias móviles más enriquecedoras, mejores y menos costosas [21].

El 5 de Noviembre del 2007 concretamente, tuvo lugar la presentación por parte de Google y Open Handset Alliance de Android, una compleja y a la vez, completa plataforma software para dispositivos móviles: un sistema operativo, middleware y las principales aplicaciones del dispositivo móvil.

Más que un sistema operativo móvil creado para una única implementación de hardware, Android fue diseñado para soportar una gran variedad de plataformas hardware, desde teléfonos con pantalla táctil a dispositivos sin ninguna pantalla. Además de eso, la no existencia de tasas de licencia ni software propietario, hacen que el coste para los fabricantes de dispositivos móviles al proporcionar variaciones de sus terminales móviles compatibles con Android sea relativamente bajo. [22]

El primer teléfono que se pudo comprar con un SO Android fue el *T-Mobile G1* de la compañía telefónica T-Mobile, en Octubre del 2008 en Estados Unidos. En España los primeros terminales con Android fueron el *HTC Dream*, por medio de Movistar, y el *HTC Magic*, por parte de Vodafone, ambos en el año 2009.

Figura 13. **T-Mobile G1**

2.4.2 Características generales

Android no diferencia entre las aplicaciones básicas del teléfono y las aplicaciones desarrolladas por terceros, lo que permite que los usuarios personalicen completamente sus dispositivos móviles mediante la instalación de nuevas aplicaciones.

Algunas de sus características principales son [23]:

- **Framework de aplicaciones**: Permite la reutilización y reemplazo de los componentes, proporcionando un alto grado de personalización por parte del usuario.
- **Máquina virtual Dalvik**: Optimizada para dispositivos móviles. Posee un funcionamiento de llamadas de instancias muy similar al de Java.
- **Gráficos optimizados**: Con una librería de gráficos 2D y los gráficos 3D están basados en la especificación *OpenGL ES 1.0*.
- **SQLite**: Base de datos para almacenamiento estructurado que se integra directamente con las aplicaciones.
- **Soporte para medios**: Con varios formatos comunes de audio, video e imágenes planas (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).

- **Hardware adicional**: Dependiendo del hardware puede incluir distintas características para el ocio como pueden ser la cámara, GPS, brújula, y acelerómetro.
- **Desarrollo accesible**: Amplio ambiente de desarrollo incluyendo un emulador de dispositivo, herramientas para depuración, perfiles de memoria y de ejecución, y un *plugin* para el IDE Eclipse.

2.4.3 Arquitectura de Android

Android es un sistema diseñado por capas. Utiliza el *kernel* de Linux 2.6 que le da acceso a la parte hardware de los dispositivos a la par que le permite ser compatible con muchos de los drivers creados para Linux. Esta arquitectura está perfectamente plasmada en el diagrama que encontramos en la siguiente figura. Algunas de las secciones se explican con mayor detalle a continuación.

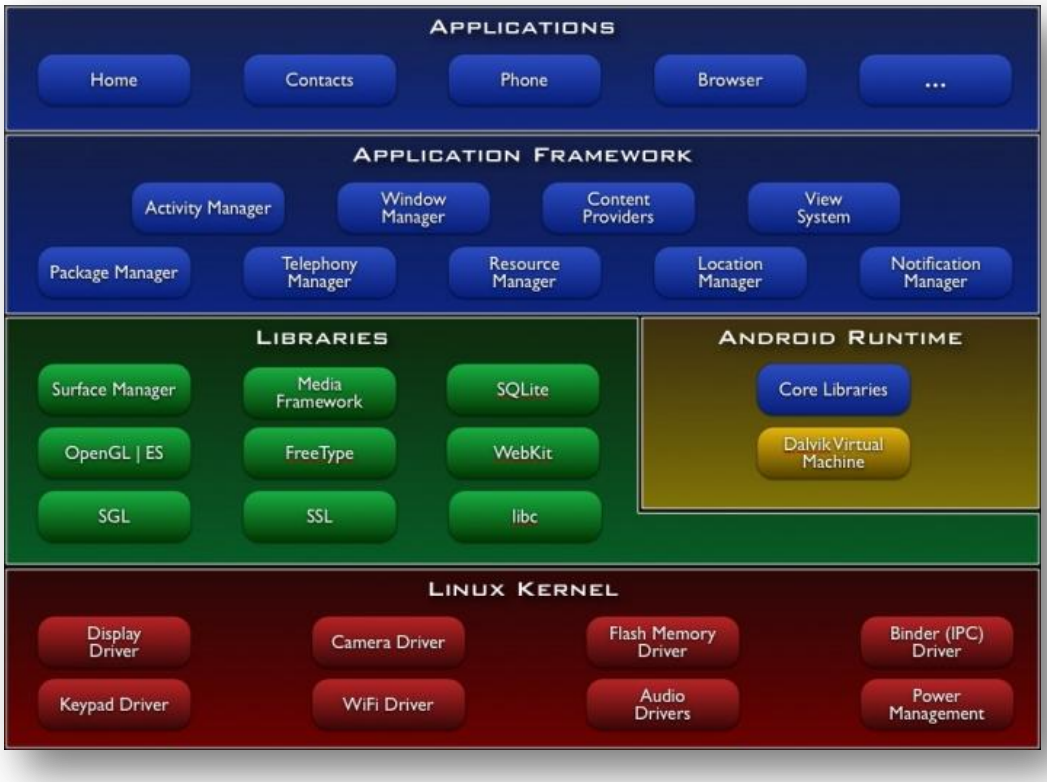


Figura 14. **Diagrama de la arquitectura de Android**

Aplicaciones

Todas las aplicaciones, tanto las incluidas en el propio Android, como las creadas por desarrolladores, están escritas en lenguaje Java y pueden estar compuestas por cuatros elementos principales: *Activity*, *Services*, *Broadcast Reveicers* y *Content Providers*. No es obligatorio que aparezcan todos ellos, simplemente se emplearán los necesarios para llevar a cabo los objetivos para los que fue diseñada dicha aplicación.

Framework

El marco de aplicaciones da acceso completo a los programadores a las mismas APIs utilizadas por las aplicaciones básicas. La arquitectura está diseñada para que la reutilización de componentes sea sencilla, ya que cualquier aplicación puede dar un perfil público a sus datos o capacidades para que otra aplicación haga uso de esas cualidades. Todo esto es posible partiendo de la base de que las normas de seguridad impuestas por el *framework* no se vean comprometidas, de esta manera se consigue incorporar un importante grado de reutilización del código. En la figura anterior podemos ver algunas de las librerías más importantes de esta capa.

Bibliotecas

El sistema incluye un conjunto de librerías en C y C++ que proporcionan la mayor parte de las funcionalidades presentes y que son utilizadas por varios de los componentes del sistema Android. Estas capacidades son expuestas a los desarrolladores a través del *framework* descrito anteriormente para que puedan ser utilizadas.

2.4.4 Máquina virtual DALVIK

Dalvik es una máquina virtual especialmente diseñada para Android, desarrollada por Dan Bornstein y su equipo de Google. Desde sus inicios, Dalvik fue pensada para los dispositivos móviles, haciendo frente a las dos principales limitaciones que éstos conllevan, la duración de la batería y la potencia de procesamiento. El hecho de estar específicamente dirigida al entorno móvil la hace diferente de la máquina virtual de Java, que fue diseñada para ser una solución universal en todos los entornos. [24]

Las aplicaciones Android tienen su código fuente escrito en Java pero Dalvik, a diferencia de Java VM (Virtual Machine), no trabaja directamente con el *bytecode* de Java, sino que lo transforma en un código más eficiente que el original pensado para procesadores pequeños. En la Figura 12 se observa como ambas máquinas virtuales parten por igual del código fuente, que una vez compilado da lugar al *bytecode* de Java. Éste será ejecutado en el caso de la Java VM, mientras que en el caso de la Dalvik VM, tendrá que ser recompilado para conseguir el *bytecode* de Dalvik.

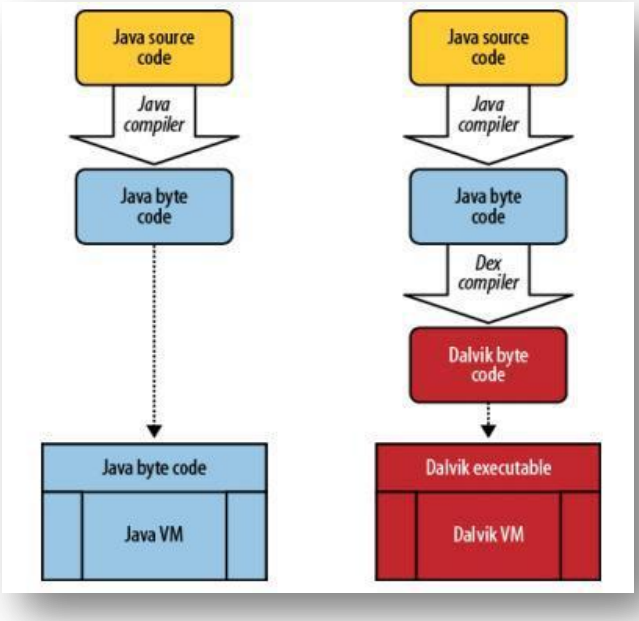


Figura 15. **Máquinas Virtuales, Java vs Dalvik**

3. Conceptualización

Mysterious Galaxy es un videojuego de naves con un estilo claramente retro. La utilización de gráficos 2D junto con la música de fondo y los efectos sonoros convierten a este juego en propio de dicha estética.

El usuario contrala a un personaje que avanza por el escenario encontrándose con una serie de enemigo a los que debe destruir o esquivar para poder completar el nivel. Dichos enemigos disparan proyectiles contra la nave del jugador con la intención de destruirla y finalizar la partida.

Figura 16. **Mysterious Galaxy**

Los aspectos comerciales se han resuelto con la compra de puntos mediante una transacción con Paypal. Estos puntos se emplean para la adquisición de diferentes armas. Los puntos también se obtienen destruyendo los enemigos a los que se enfrenta al jugador, pero los precios están ajustados para que sea complicado conseguir las armas especiales de este modo. La puntuación obtenida por la eliminación de naves es baja en comparación con los necesarios para la compra de estas armas. De esta forma se incita al usuario a pagar para conseguir los puntos necesarios.

La prueba de concepto de este videojuego se realizara en la versión de Android 2.2 (Froyo) en un terminal Samsung Galaxy Mini debido a la limitación de recursos de los que disponía. Realizar este proyecto para la versión 2.2 es una decisión que se hubiera mantenido de haber tenido acceso a más alternativas porque aunque existen versiones superiores (actualmente alcanza la versión 4.2) la mayoría de los usuarios disponen de la versión Froyo.

3.1 Elementos gráficos y sonoros

El aspecto más importante para la creación de un videojuego de estilo retro es que los elementos gráficos y sonoros se asemejen a los de entonces. Para ellos se han buscado imágenes con pocos detalles y simples. En cuanto al sonido se ha optado por una melodía sin complejos efectos sonoros.

Las *sprites* de las naves se han creado utilizando la galería libre del SketchUp de Google donde se pueden encontrar diferentes naves en tres dimensiones. A partir de los diseños en 3D se han sacado diferentes *frame* para la composición de los *sprites.*

Figura 17. ***Sprites* de las Naves**

Los fondos del videojuego, las balas y el *sprite* de explosión de las naves se han sacado de imágenes libres de derecho de autor. Siempre siguiendo el criterio mencionado anteriormente para conseguir una estética retro.

Los botones se han realizado mediante una página web de diseño de botones, pero la mayoría han sido retocados mediante Photoshop para obtener el resultado deseado. Algunas de las modificaciones que se han hecho en los botones son: modificación de la transparencia, composición de imágenes, redimensionado…La barra de vida y la puntuación se han creado a partir de los botones para mantener el mismo diseño en todos los componentes del videojuego.

En cuanto al sonido, para el efecto del disparo se ha utilizado un sonido polifónico perteneciente a la plataforma Android que rápidamente recuerda a los clásicos sonidos de disparo de naves espaciales.

La melodía de fondo del videojuego ha sido creada mediante un programa llamado Beaterator usando diferentes *beats* de temáticas espaciales consiguiendo transportar al jugador a la época que se desea.

3.2 Nave del jugador

La nave del jugador está controlada por cuatro botones de dirección (derecha, izquierda, arriba y abajo), un botón para el disparo y otro para el arma especial. Al integrarse los botones en la propia pantalla del juego se han hecho translucidos para que el usuario pueda ver los elementos que se ubiquen detrás de dichos controles.

El jugador cuenta con una vida inicial correspondiente a siete colisiones ya sea con una nave o con una bala.

El *sprite* que representa esta nave está formado por un conjunto de imágenes con diferentes inclinaciones laterales para que al jugar el usuario perciba la sensación de desplazamiento lateral de la nave. De forma que cuando la nave se desplaza a la derecha esta se va inclinando progresivamente a dicho lado. Cuando el jugador suelta el botón la nave vuelve a su posición original.

Figura 18. ***Sprite* de la nave**

El jugador tiene dos tipos de armas diferentes:

- **Disparos**: Existen tres tipos de disparos que se diferencian en el número de balas que lo componen, siendo así de uno, dos o tres proyectiles. El disparo no puede ser continuado, tiene que pasar un pequeño intervalo de tiempo entre cada disparo.

Figura 19. **Tipos de disparos**

- **Armas especiales**: Existen tres tipos de armas especiales:
  - Vida extra: Regenera la barra de la vida de la nave.

Figura 20. **Arma especial Vida extra**

- Pausar tiempo: Paraliza las naves y balas enemigas durante 15 segundos.

Figura 21. **Arma especial Pausar tiempo**

- Escudo: La nave es cubierta por un escudo que impide ser dañada tanto por la colisión con naves enemigas como por proyectiles.

Figura 22. **Arma especial Escudo**

3.3 Niveles

El juego consta de 15 niveles. Inicialmente solo se puede jugar al primer nivel, de forma que cuando se supere éste se desbloquea el segundo y así sucesivamente. Para aumentar la dificultar de los niveles cada nivel tiene 10 naves enemigas más que el nivel anterior. Teniendo en cuenta que se ha iniciado con 20 naves en el primer nivel, el nivel 15 consta de 170 naves.

Figura 23. **Ventana de Niveles**

El videojuego tiene tres tipos de enemigos diferentes que se diferencian en el *sprite*, el desplazamiento, el número de disparos y en la puntuación que obtiene el jugador por eliminarla.

En la parte superior de todos los niveles se muestra la barra de vida del jugador y la puntuación que va obteniendo el jugador en la pantalla correspondiente. Además, para mejorar la experiencia del jugador se ha creado el fondo de la pantalla de modo que se desplace constantemente dando la sensación del movimiento de las naves.

3.4 Compra de armas y pago mediante Paypal

La tienda de armas tiene dos funcionalidades: comprar las armas que se deseen y seleccionar cual es la que se va a utilizar. El arma de un solo disparo está disponible desde el comienzo del juego pero el resto de armas tienen que ser compradas. Las armas especiales son de un único uso por lo que se tendrá que pagar con ella cada vez que se quiera usar, mientras que para los disparos sólo es necesario pagar por ellos una vez.

Figura 24. **Ventana de Tienda, modo comprar**

Figura 25. **Ventana de Tienda, modo seleccionar**

Desde la tienda se puede acceder a la compra de puntos. El jugador tiene diferentes importes que puede gastar que oscilan desde los 0,80 a los 5 €. Una vez seleccionado el importe que se desea canjear se accede a una ventana para introducir la cuenta de Paypal y se inicia la transacción. Cuando se recibe la confirmación del pago por parte de Paypal los puntos son incrementados.

Figura 26. **Ventana Compra de Puntos**

3.5 Otras funcionalidades

A parte de jugar a los diferentes niveles y de la compra de armas, Mystetious Galaxy ofrece otras posibilidades como el almacenamiento de puntuaciones y las instrucciones.

En la pantalla de puntuaciones se almacena la mayor puntuación obtenida en cada nivel, por lo que inicialmente todas las puntuaciones están a cero.

Figura 27. **Ventana de Puntuaciones**

La ventana instrucciones presenta toda la información referente a las funcionalidades del juego. Es una parte importante dentro del videojuego, ya que es necesario que el jugador entienda correctamente los controles del juego, sea conocedor de los métodos para lleva a cabo la compra de armas y del modo de adquisición de puntos mediante pago por Paypal.

Figura 28. **Ventana de Instrucciones**

4. Análisis

En este apartado se realiza un análisis de la aplicación desarrollada, en el que se incluye el estudio de los casos de uso y el diagrama de actividad del sistema. Finalmente, se concluye con los requisitos de usuario, seguidos de los requisitos de software divididos en funcionales y no funcionales.

4.1 Casos de Uso

Mediante el modelado de casos de uso se consigue identificar las funcionalidades del videojuego. Los casos de uso representan un uso típico del sistema, por lo que permite definir los requisitos que se deben cumplir en una aplicación y las interacciones que existen entre el usuario y el sistema. Este diagrama se representa para alcanzar una mejor comprensión de la aplicación a desarrollar.

Para el videojuego en el que nos centramos, se identifica un único actor que es el jugador o usuario de la aplicación.

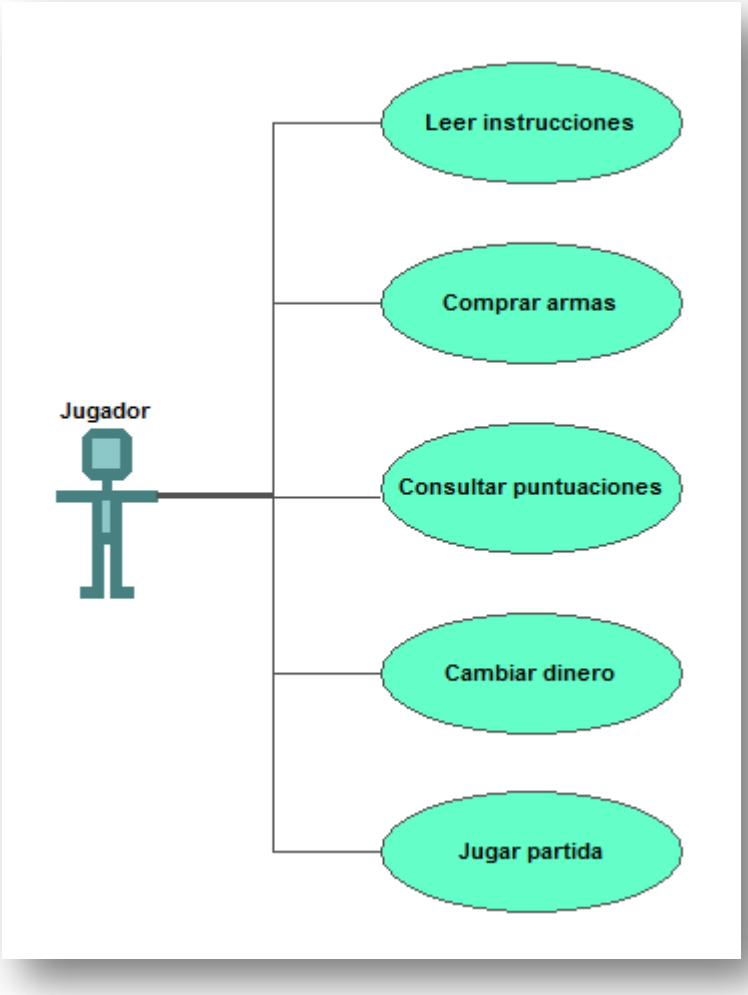


Figura 29. **Diagrama de Casos de Uso.**

Una vez definidos los casos de uso, cada uno de ellos se especificará mediante una tabla con la siguiente información:

- **Identificador**: Registra de forma unívoca un caso de uso siguiendo el formato de nombrado CU_XX. El valor de XX es un identificador numérico y secuencial.
  - **Nombre:** Título que recibe un caso de uso.
  - **Actor**: Determina qué actor ha realizado dicho uso.
  - **Descripción**: Breve explicación de los detalles del caso de uso.
  - **Precondiciones:** Condiciones bajo las cuales se da el caso de uso.
  - **Post-condiciones:** Indican el estado del sistema después de la ejecución del caso de uso.

Identificador	CU-01
Título	Leer instrucciones
Actores	Jugador
Objetivo	El jugador debe acceder a las instrucciones para saber manejar los diferentes controles y conocer el modo de obtención de las armas.
Precondiciones	Situarse en el menú principal.
Post-condiciones	Mostrar en pantallas las instrucciones del juego.

Tabla 2. CU-01 Leer instrucciones.

Identificador	CU-02
Título	Comprar armas
Actores	Jugador
Objetivo	El jugador comprar con los puntos conseguidos distintos disparos y armas especiales.
Precondiciones	Situarse en el menú principal.
Post-condiciones	Adquisición de un arma nueva.

Tabla 3. CU-02 Comprar armas.

Identificador	CU-03
Título	Consultar puntuaciones
Actores	Jugador
Objetivo	Mostrar en pantallas la puntuación más alta de cada nivel. En caso de que el nivel este bloqueado la puntuación será cero.
Precondiciones	Situarse en el menú principal.
Post-condiciones	Mostrar en pantallas las puntuaciones del juego.

Tabla 4. CU-03 Consultar puntuaciones

Identificador	CU-04
Título	Cambiar dinero
Actores	Jugador
Objetivo	El jugador hace un pago mediante Paypal para obtener puntos con los que comprar armas.
Precondiciones	Situarse en la tienda de armas.
Post-condiciones	Aumento de los puntos necesarios para adquirir armas.

Tabla 5. CU-04 Cambiar dinero

Identificador	CU-05
Título	Jugar Partida
Actores	Jugador
Objetivo	El jugador seleccionara el nivel desbloqueado al que desee jugar y comenzara la partida.
Precondiciones	Situarse en el menú principal.
Post-condiciones	Inicio de la partida.

Tabla 6. CU-05 Jugar Partida

4.2 Diagrama de Actividad

Un diagrama de actividad muestra los distintos estados de un sistema describiendo las secuencias que sigue éste. Estas transiciones influyen en el comportamiento y evolución del sistema provocando que éste cambie de un estado a otro como consecuencia de eventos que se producen.

Los casos de uso nos mostraban las distintas funcionalidades que tiene el sistema y con el diagrama de actividad se define el comportamiento del sistema ante los eventos que suceden.

A continuación se muestra el diagrama de actividad del sistema.

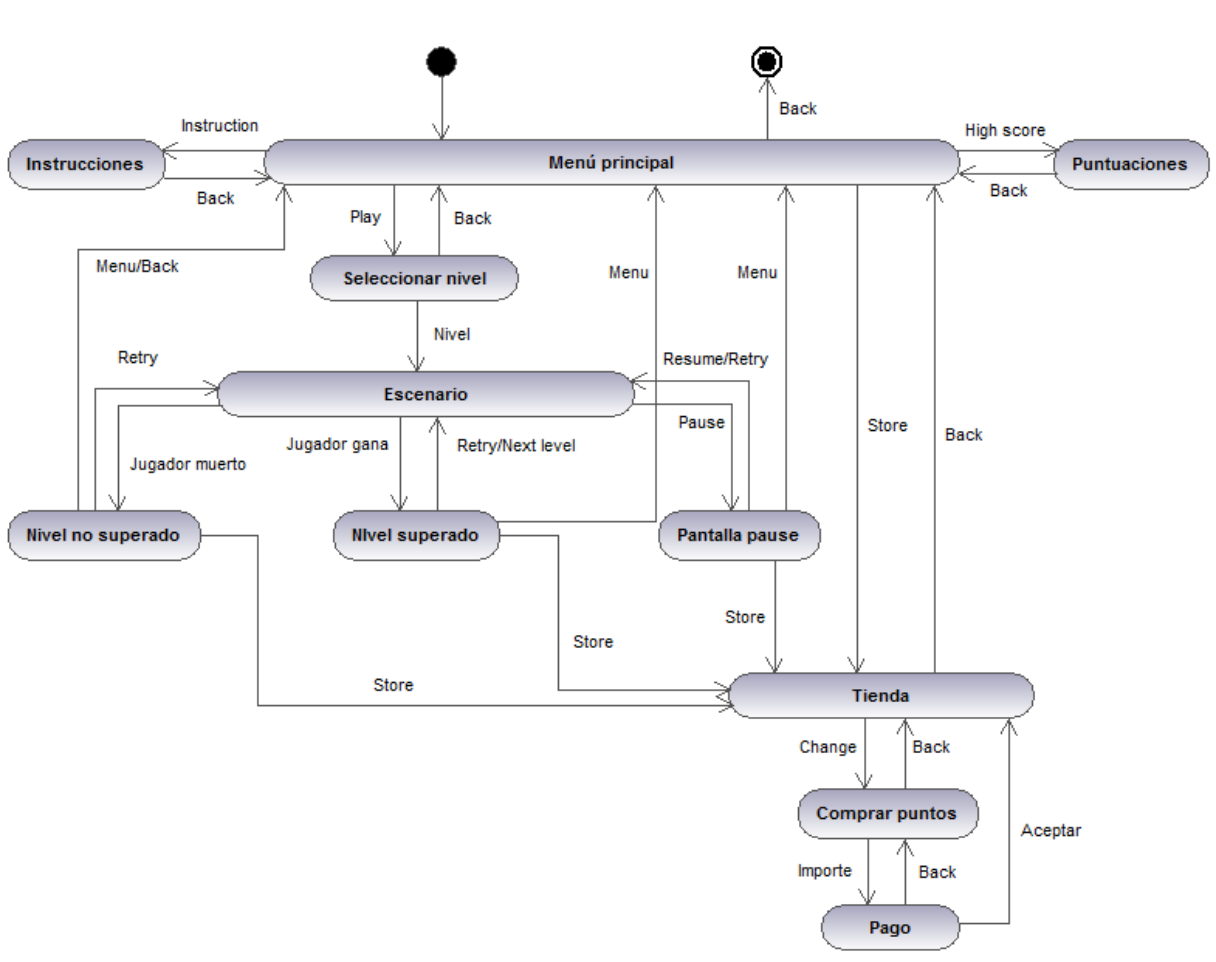


Figura 30. **Diagrama de actividad del sistema**

4.3 Requisitos de usuario

En este apartado del documento se identifican, clasifican y catalogan los distintos requisitos de usuario. Estos son especificaciones de las funcionalidades del sistema que han de ser resueltas.

Dentro de los requisitos de usuario se hará una diferenciación entre los requisitos de capacidad, que representan una necesidad o servicio requerido por el usuario, y los requisitos de restricción, que representan una restricción o imposición para el usuario.

Para su especificación se hará uso de unas tablas con los siguientes campos:

- **Identificador:** Registra de forma unívoca un requisito. Los requisitos de usuario de capacidad se identificarán mediante la nomenclatura RUC_XX y los requisitos de restricción con RUR_XX. El valor de XX es un identificador numérico y secuencial.
- **Descripción:** Especificación detallada del requisito.
- **Origen o fuente:** El caso de uso del que proviene.
- **Verificabilidad:** Define si el requisito se puede acreditar mediante una prueba.
- **Claridad:** Determina si el requisito se entiende con precisión, o por el contrario, si su definición es ambigua.
- **Prioridad:** Establece la importancia del requisito dentro del proyecto. Toma tres valores: “Alta”, “Media” o “Baja”.
- **Necesidad:** Fija la obligación del cumplimiento del requisito dentro del proyecto. Toma dos valores: “Esencial” u “Opcional”.
- **Estabilidad:** Indica el grado de variabilidad que puede tener un requisito a lo largo del proyecto. Toma tres valores: “Alta”, “Media” o “Baja”.

Identificador	RUC-01
Título	Tienda
Descripción	Tienda donde ser pueda comprar las diferentes armas. La compra se realizara mediante puntos.
Origen o Fuente	CU-02
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 7. RUC-01 Tienda

Identificador	RUC-02
Título	Tipos de armas
Descripción	Existirán tres tipos de disparos diferentes y tres tipos de armas especiales.
Origen o Fuente	CU-02
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Media

Tabla 8. RUC-02 Tipos de armas

Identificador	RUC-03
Título	Compra de puntos
Descripción	Compra de puntos, para la compra de armas, mediante pago por Paypal. El jugador podrá elegir entre diferentes importes.
Origen o Fuente	CU-04
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Media

Tabla 9. RUC-03 Compra de puntos

Identificador	RUC-04
Título	Compra jugando
Descripción	Se debe poder tanto comprar como canjear puntos durante la partida.
Origen o Fuente	CU-02, CU-04
Verificabilidad	Si
Claridad	Si
Prioridad	Media
Necesidad	Opcional
Estabilidad	Media

Tabla 10. RUC-04 Compra jugando

Identificador	RUC-05
Título	Instrucciones
Descripción	Explicar los controles del juego y el modo de compra de las armas.
Origen o Fuente	CU-01
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 11. RUC-05 Instrucciones

Identificador	RUC-06
Título	Puntuaciones
Descripción	Consulta de la mejor puntuación de cada nivel.
Origen o Fuente	CU-03
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 12. RUC-06 Puntuaciones

Identificador	RUC-07
Título	Elección nivel
Descripción	Se mostraran en una pantalla todos los niveles, para que el jugador seleccione al que desee jugar.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 13. RUC-07 Elección nivel

Identificador	RUC-08
Título	Pausa
Descripción	El jugador podrá pausar la partida en cualquier momento.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 14. RUC-08 Pausa

Identificador	RUC-09
Título	Cambio de armas
Descripción	Se debe poder cambiar las armas durante la partida.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Media
Necesidad	Opcional
Estabilidad	Media

Tabla 15. RUC-09 Cambio de armas

Identificador	RUC-10
Título	Vida
Descripción	El jugador pierde cuando la nave reciba siete impactos.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Media

Tabla 16. RUC-10 Vida

Identificador	RUR-01
Título	Tiempo de disparo
Descripción	El tiempo entre dos disparos del jugador tiene que ser igual o superior a 0,5 segundos.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Media
Necesidad	Esencial
Estabilidad	Baja

Tabla 17. RUR-01 Tiempo de disparo

Identificador	RUR-02
Título	Niveles bloqueados
Descripción	Al inicio, el juego solo tendrá desbloqueado el primer nivel y conforme se supere una pantalla se podrá acceder a la siguiente.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 18. RUR-02 Niveles bloqueados

Identificador	RUR-03
Título	Movimiento diagonal
Descripción	La nave no se podrá mover en diagonal.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Media
Necesidad	Esencial
Estabilidad	Media

Tabla 19. RUR-03 Movimiento diagonal

4.4 Requisitos de software

En este apartado se identifican, clasifican y catalogan los principales requisitos de software del sistema. Estos proporcionan una visión completa del comportamiento global del sistema. Cada requisito está descrito mediante la siguiente información:

- **Identificador:** Registra de forma unívoca un requisito, asignando a cada uno el nombre RSF_XX para los requisitos funcionales, y con RSNF_XX para los que no lo son. . El valor de XX es un identificador numérico y secuencial.
- **Descripción:** Especificación detallada del requisito.
- **Origen o fuente:** El caso de uso del que proviene.
- **Verificabilidad:** Define si el requisito se puede acreditar mediante una prueba.
- **Claridad:** Determina si el requisito se entiende con precisión, o por el contrario, si su definición es ambigua.
- **Prioridad:** Establece la importancia del requisito dentro del proyecto. Toma tres valores: “Alta”, “Media” o “Baja”.
- **Necesidad:** Fija la obligación del cumplimiento del requisito dentro del proyecto. Toma dos valores: “Esencial” u “Opcional”.
- **Estabilidad:** Indica el grado de variabilidad que puede tener un requisito a lo largo del proyecto. Toma tres valores: “Alta”, “Media” o “Baja”.

Se debe destacar que dentro de los requisitos no funcionales existe otra clasificación:

- **Interfaz:** Hacen referencia a la interfaz del sistema a través de la cual el usuario deberá interactuar.
- **Documentación:** Relacionados con temas de ayudas y tutoriales.
- **Operacionales:** Concernientes con el entorno de ejecución donde debe funcionar el sistema.
- **Rendimiento:** Garantizan el buen funcionamiento de la aplicación.
- **Seguridad ante Fallos:** Permiten a la aplicación recuperarse ante fallos u operaciones indebidas realizadas por los
- **Recursos:** Indica los recursos que la aplicación necesita para su correcto funcionamiento.

4.4.1 Requisitos funcionales

Identificador	RSF-01
Título	Detectar eventos pantalla
Descripción	El videojuego debe capturar los distintos eventos sobre la pantalla táctil del dispositivo móvil para poder realizar distintas acciones.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 20. RSF-01 Detectar eventos pantalla

Identificador	RSF-02
Título	Música
Descripción	Desde que se inicie la aplicación se reproducirá una melodía.
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Media
Necesidad	Esencial
Estabilidad	Media

Tabla 21. RSF-02 Música

Identificador	RSF-03
Título	Silenciar música
Descripción	Desde la pantalla inicial y desde la pausa del juego se debe poder silenciar la música.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 22. RSF-03 Silenciar música

Identificador	RSF-04
Título	Almacenamiento puntuaciones
Descripción	Las puntuaciones máximas de cada nivel se tienen que almacenar de forma persistente.
Origen o Fuente	CU-03
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 23. RSF-04 Almacenamiento puntuaciones

Identificador	RSF-04
Título	Pantalla principal
Descripción	Desde la pantalla principal se debe poder acceder a las instrucciones, puntuaciones, a la tienda y al inicio del juego.
Origen o Fuente	CU-01, CU-02, CU-03, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 24. RSF-04 Pantalla principal

Identificador	RSF-05
Título	Cambio de dinero
Descripción	Desde la tienda se accederá al pago a través de Paypal.
Origen o Fuente	CU-03, CU-04
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 25. RSF-05 Cambio de dinero

Identificador	RSF-06
Título	Movimiento del fondo
Descripción	El fondo del videojuego se moverá constantemente para dar la sensación de movimiento de las naves.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Media
Necesidad	Esencial
Estabilidad	Media

Tabla 26. RSF-06 Movimiento del fondo

Identificador	RSF-07
Título	Puntuación y vida
Descripción	Durante la partida, el jugador estará informado de la vida y la puntuación de la nave.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 27. RSF-07 Puntuación y vida

4.4.2 Requisitos no funcionales

Requisitos de interfaz

Identificador	RSNF-01
Título	Atractiva
Descripción	El producto debe tener un diseño atractivo capaz de atraer a los posibles usuarios.
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	No
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 28. RSNF-01 Atractiva

Identificador	RSNF-02
Título	Intuitiva
Descripción	La interfaz se debe presentar de manera clara al usuario para facilitar su uso.
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 29. RSNF-02 Intuitiva

Requisitos de documentación

Identificador	RSNF-03
Título	Instrucciones
Descripción	Debe existir un abreve explicación de los diferentes elementos del juego y de la compra de puntos.
Origen o Fuente	CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 30. RSNF-03 Instrucciones

Identificador	RSNF-04
Título	Mensajes de compra
Descripción	Al comprar de un arma se debe emitir un mensaje indicando si se ha podido o no realizar la compra.
Origen o Fuente	CU-04
Verificabilidad	Si
Claridad	Si
Prioridad	Media
Necesidad	Esencial
Estabilidad	Media

Tabla 31. RSNF-04 Mensajes de compra

Requisitos operacionales

Identificador	RSNF-05
Título	Idioma
Descripción	El idioma del videojuego en el que se debe presentar es el inglés.
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Media
Necesidad	Esencial
Estabilidad	Media

Tabla 32. RSNF-05 Idioma

Identificador	RSNF-06
Título	Sistema operativo Android
Descripción	El dispositivo móvil del usuario final debe contener el sistema operativo Android con versión 2.2 (Froyo).
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 33. RSNF-06 Sistema operativo Android

Identificador	RSNF-07
Título	Pantalla
Descripción	La interfaz debe estar diseñada para la correcta visualización en el terminal Samsung Galaxy Mini.
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 34. RSNF-07 Pantalla

Requisitos de Rendimiento

Identificador	RSNF-08
Título	Liberación de recursos
Descripción	Los recursos vinculados a la aplicación deben ser liberados al finalizar su uso.
Origen o Fuente	CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 35. RSNF-08 Liberación de recursos

Identificador	RSNF-09
Título	Permisos
Descripción	Durante la instalación del juego, el usuario debe ser informado los permisos necesarios para que la aplicación pueda funcionar de forma correcta.
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Media
Necesidad	Esencial
Estabilidad	Media

Tabla 36. RSNF-09 Permisos

Requisitos de seguridad frente a errores

Identificador	RSNF-10
Título	Control de errores
Descripción	El sistema debe ofrecer opciones cerradas y un control del entorno gráfico para evitar posibles errores.
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 37. RSNF-10 Control de errores

Requisitos de recursos

Identificador	RSNF-11
Título	Formato música
Descripción	El formato de la música debe de ser ogg.
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 38. RSNF-11 Formato música

Identificador	RSNF-12
Título	Formato imágenes
Descripción	El formato de las imágenes debe de ser png.
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 39. RSNF-12 Formato imágenes

Identificador	RSNF-13
Título	SQLite
Descripción	Uso de una base de datos SQLite para el almacenamiento persistente de datos.
Origen o Fuente	CU-01, CU-02, CU-03, CU-04, CU-05
Verificabilidad	Si
Claridad	Si
Prioridad	Alta
Necesidad	Esencial
Estabilidad	Alta

Tabla 40. RSNF-13 SQLite

5. Diseño

En este apartado se presentará el diagrama de los módulos en los que se divide la aplicación y, posteriormente, se realiza la explicación de cuál es la función de cada uno de ellos individualmente así como con los restantes bloques.

5.1 Arquitectura del sistema

El videojuego *Mysterious Galaxy* se divide en un conjunto de módulos relacionados entre sí, de modo que se minimicen las dependencias entre ellos y poder así separarlos con facilidad. Cada uno tiene una funcionalidad específica y la unión de todos ellos producen como resultado la aplicación desarrollada.

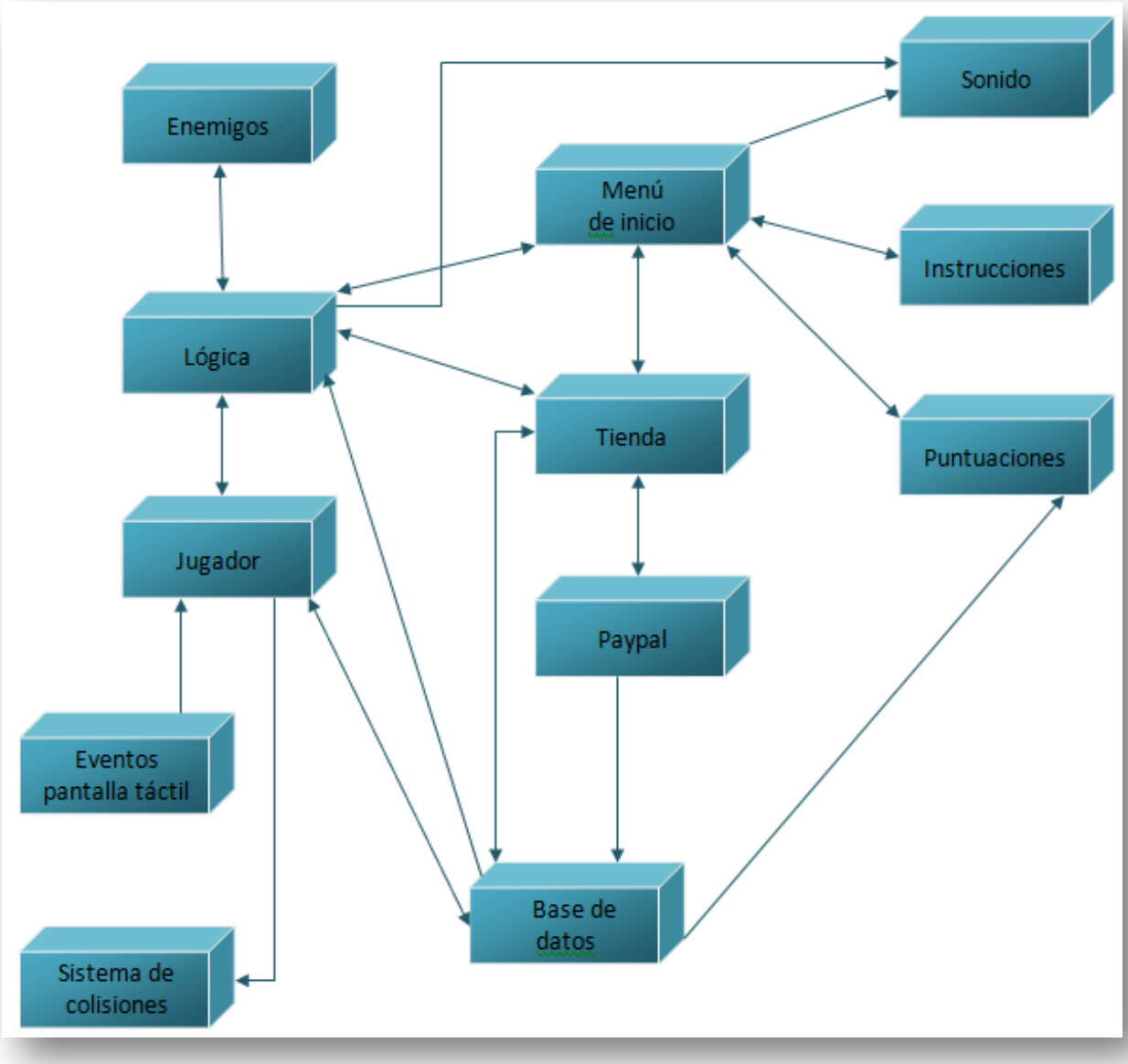


Figura 31. **Arquitectura del sistema**

5.2 Descripción de los módulos

Una vez observada las relaciones de dependencia existente entre los módulos, a continuación se detallará el papel de cada uno de ellos en el conjunto.

5.2.1 Menú de inicio

Es el encargado de mostrar el menú principal cuando se inicia la aplicación. Desde él se puede acceder a todas las funcionalidades del videojuego, como por ejemplo la modificación del estado de la reproducción de música ó el acceso a la tienda con el fin de adquirir un arma. Además, es el punto de salida de la aplicación, por lo que desde cualquier punto se puede llegar al menú de inicio retrocediendo a través de sus pantallas, hasta finalmente cerrarla.

Por lo tanto, el menú de inicio es considerado el núcleo principal de todo el proyecto, dado que es el punto de partida hacia las distintas posibilidades que la aplicación ofrece, retornando a éste una vez finalizadas, pudiendo producir distintos eventos dependiendo de la causa.

5.2.2 Sonido

Se encarga de iniciar o pausar la reproducción de la música. Para la implementación de este módulo es importante tener en cuenta que la melodía sonará independientemente de la ubicación del usuario en la aplicación. Otra funcionalidad de su competencia es la reproducción constante, por lo que cuando finalice la música se iniciará otra vez. El acceso al control del estado del hilo musical está limitado al menú inicial y al estado en pausa del videojuego.

5.2.3 Instrucciones

En él se encuentra toda la información referente a las funcionalidades del juego. El usuario no puede interactuar con este módulo, solo puede acceder para leer las instrucciones del videojuego. Su función dentro de este proyecto es muy importante ya que, es necesario que el usuario entienda correctamente los

controles del juego, sea conocedor de los métodos para lleva a cabo la compra de armas y la adquisición de puntos mediante pago por Paypal.

5.2.4 Puntuaciones

Es el encargado de mostrar al usuario las puntuaciones máximas obtenidas en cada nivel del videojuego. Estas están almacenadas en la base de datos, por lo que se tiene que hacer peticiones a la base de datos para presentar la información. La comunicación con la base de datos es sólo de lectura, el almacenamiento de los valores máximos es llevado a cabo por la parte lógica, que una vez finalizada la pantalla la actualiza si ha sido superada la puntuación.

5.2.5 Tienda

Este es un módulo fundamental dentro de la aplicación porque dota al juego de muchas funcionalidades y posibilidades para que la experiencia le resulte más entretenida al usuario. Su cometido es mostrar de forma inequívoca las armas disponibles y el precio de éstas. Las armas compradas tienen que ser almacenadas en la base de datos para que el jugador pueda utilizar las que haya ido adquiriendo al cargar una nueva pantalla. También es de su competencia el proveer al jugador de la elección entre las armas compradas.

5.2.6 Paypal

Esta es la parte más importante desde el punto de vista comercial ya que es la responsable de los ingresos de la aplicación. Es necesario que sea estable para que no se produzcan problemas en las transacciones monetarias. Debido a lo delicado de esta parte se ha optado por crearlo a partir de las librerías de Paypal, de este modo nos garantizamos su correcto funcionamiento, ya que el reconocimiento y experiencia de dicha empresa proporciona seguridad al cliente.

El cliente podrá elegir entre diferentes cantidades de puntos que puede comprar. Una vez realizada la compra se accede a la base de datos para actualizar los mismos.

El empleo de Paypal como sistema de transacciones conlleva el pago de unas tarifas por cada una que se realice. Si las transacciones son de carácter nacional Paypal obtendrá desde el 3,4 al 1,9 % de los ingresos, este porcentaje depende del total de los pagos recibidos, más 0,35 € por cada transacción. Si la compra es internacional se suma entre un 0,4 y un 1,5 % dependiendo del país de origen.

5.2.7 Lógica

Es el núcleo del videojuego, se encarga de la llamada a los módulos necesarios para su correcto funcionamiento. Es el bloque encargado de presentar por pantalla la interfaz relacionada con el juego: el fondo, la barra de la vida, la puntuación, los botones, la nave protagonista y los enemigos. Además, accede a la base de datos para saber en qué momento y cuál es la nave enemiga que tiene que incorporase a la partida. Finalmente, maneja el ciclo de vida del hilo principal de juego ya que éste puede pausarse para cambiar o adquirir nuevas armas.

5.2.8 Enemigos

Existen tres tipos de enemigos que se diferencian por la imagen de la nave y por la dirección del desplazamiento. Por cada pantalla se encuentra una tabla en la base que indica en qué instante, qué nave y en qué posición va a aparecer. Para el diseño de este módulo hay que tener en cuenta: que el contacto entre ellas no se considera colisión y que cuando una nave es eliminada por el protagonista o desaparece del campo de visión del juego se debe liberar el recurso.

5.2.9 Jugador

Es el elemento del videojuego que controla el usuario por medio de los eventos de la pantalla táctil. Interacciona con la base de datos para saber el tipo de disparo y arma especial que el usuario desea utilizar. También se comunica directamente con el módulo de pantalla táctil, para que la nave reaccione si se ha pulsado alguno de los botones del juego. Por último, está relacionado con el

bloque de colisión para reducir la vida que le queda a la nave en caso de impacto.

5.2.10 Sistema de colisiones

El sistema de colisiones da lugar a que el videojuego tenga un desarrollo lógico y se traten las distintas acciones que se pueden producir durante una partida. En él se determina si se ha producido alguna colisión entre: protagonista-bala enemiga, protagonista-enemigos, bala protagonista-enemigo, escudo enemigo o escudo-bala enemiga. Para ello se comprueba si la ubicación de una de estas imágenes está superpuesta con otra.

5.2.11 Eventos pantalla táctil

Su misión es la de capturar los distintos eventos que se producen sobre la pantalla táctil del dispositivo móvil en el que se ejecute la aplicación. Entra en acción cuando se ha iniciado una partida, para determinar qué evento se ha producido, sobre qué pixeles concretos ha ocurrido la pulsación y si éstos se encuentran sobre la zona delimitada para los botones de movimiento de la nave o sobre los de disparo. Posteriormente a la realización de estas comprobaciones, actuará el módulo jugador en función del resultado que se haya obtenido, produciendo el movimiento o disparo de la nave controlado. Su cometido es esencial e indispensable para el desarrollo normal del videojuego, dado que sin él, el usuario no podría hacer progresar a la nave a lo largo de los distintos escenarios que se presentan, ni tampoco abatir a las naves enemigas.

5.2.12 Base de datos

La base de datos es el pilar para la configuración de la aplicación. En ella se encuentran todos los parámetros que el jugador puede variar, permitiendo de este modo que al iniciar otra partida se cargue con la última configuración almacenada.

En esta base de datos se diferencian tres tipos de tablas:

- **Levels**: Contiene una entrada por cada nivel del videojuego en el que se indica el nombre del nivel, la puntuación máxima y si esta desbloqueado o no.
- **Player**: En esta tabla se almacena la configuración del jugador. Se indica el tipo de disparo elegido y desbloqueados. También se almacena el número de armas especiales de cada tipo que se han comprado y cuál de éstas es la que el jugador tiene actualmente seleccionada.
- **Level_1-15**: Estas son 15 tablas en las que se almacena la configuración de cada nivel. Por cada nave del nivel correspondiente existe una entrada en la que se indica la posición relativa en el eje y en la que va a aparecer la nave, el tipo de nave y el instante en el que aparecerá.

6. Implementación

Una vez realizado el análisis y diseño de la aplicación, en este apartado se procede a desglosar la implementación de la lógica del videojuego. No se describe todo el proceso de desarrollo de la aplicación, sino que se profundiza solamente en los aspectos más importantes de su implementación.

En primer lugar se presenta el diagrama de clases del proyecto y, posteriormente, se explicará el funcionamiento interno de: la detección de colisiones, la interfaz gráfica, la compra de puntos, la lógica del juego, la detección de eventos táctiles, la reproducción de audio y la comunicación con la base de datos.

6.1 Diagrama de clases

El diagrama de clases es utilizado para describir las clases en las que está dividido el sistema, las relaciones existentes entre unas y otras, como son las relaciones de herencia o de visibilidad, y define los atributos y métodos de cada una de ellas.

El código de esta aplicación está dividido en tres paquetes, cada uno de ellos con una funcionalidad muy diferenciada:

- **Package *view***: En él se encuentran todas las clases referentes a la interfaz gráfica. Está relacionado con el paquete *game* por medio de la clase *Game* que es la *Activity* responsable de presentar la interfaz del videojuego. También se comunica con el paquete *tools* para completar sus funcionalidades como es el caso de la reproducción de la música o de la compra de puntos mediante Paypal.
- **Package *game***: Encierra de toda la lógica del videojuego: detección de colisiones, enemigos, eventos de la pantalla táctil, nave del jugador… Interactúa con el paquete *tools* entre otras situaciones para la carga de los datos de cada nivel almacenados en la base de datos.
- **Package *tools***: En este paquete se encuentra las herramientas que ayudan a un desarrollo correcto de la aplicación y que no tiene que ver directamente con la interfaz o con la lógica del juego, como por ejemplo el acceso a la base de datos o la reproducción de música.

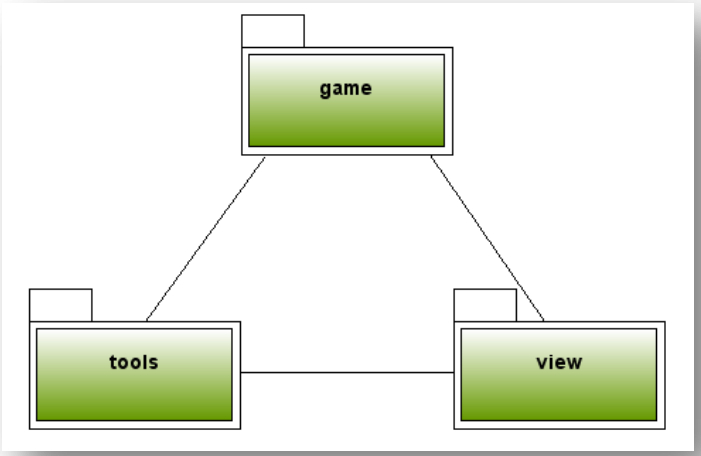


Figura 32. **Relación de los paquetes**

Seguidamente presentaremos el diagrama de clases completo:

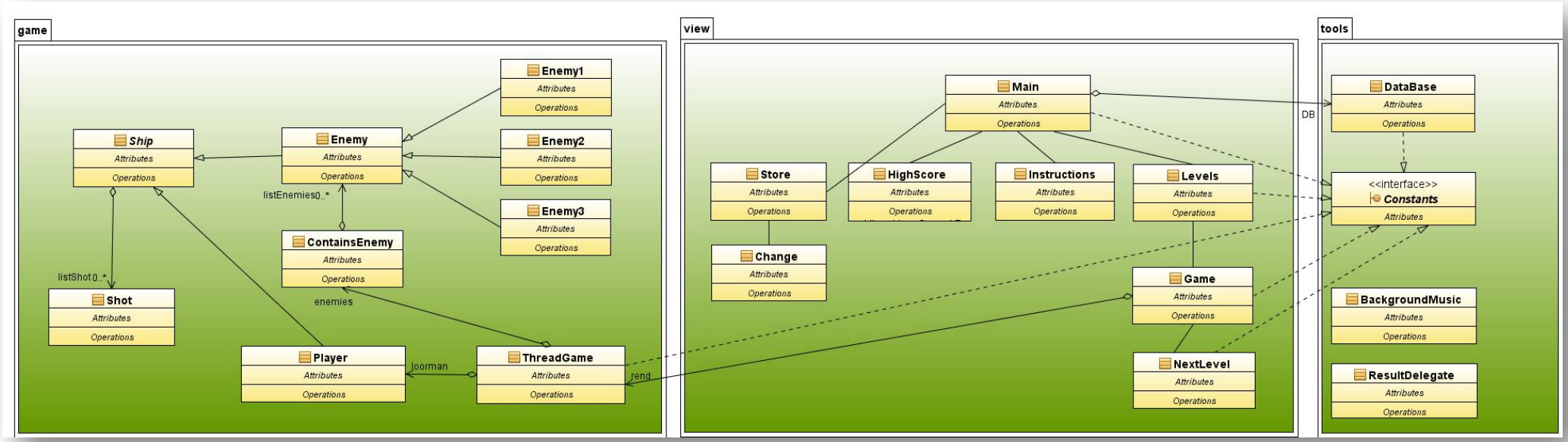


Figura 33. **Diagrama de clase**

6.2 Fichero AndroidManifest.xml

Toda aplicación diseñada para la plataforma Android debe tener un archivo *AndroidManifest.xml* en su directorio raíz. El manifiesto contiene información esencial acerca de la aplicación para el sistema. Entre algunas de sus funcionalidades se encuentra: asignar los permisos que la aplicación debe tener para acceder a partes protegidas del API e interactuar con otras aplicaciones, declarar el nivel mínimo de la API de Android que requiere la aplicación y describir los componentes de la aplicación.

La etiqueta *<uses-sdk>* establece las versiones para las que se puede utilizar esta aplicación. Debido a que los requisitos de software indicaban que la aplicación debía funcionar la para la versión 2.2 la configuración ha sido:

```
<uses-sdk    android:minSdkVersion="8"
             android:targetSdkVersion="8"
             android:maxSdkVersion="8"/>
```

Código 1. **uses-sdk**

Para el correcto funcionamiento del pago a través de Paypal se han incluido los permisos de acceso a internet y de lectura del estado del teléfono.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

Código 2. **uses-permission**

En cuanto a la declaración de los componentes de la aplicación cabe destacar que debido a que el videojuego está diseñado para jugar en sentido horizontal es necesario establecer un atributo de las *activities* para que ésta no cambie su orientación dependiendo de la posición del teléfono móvil.

```
<activity android:name="mysteriusgalaxy.view.Change"
          android:screenOrientation="landscape"/>
```

Código 3. **Activity**

6.3 Interfaz

El primer paso en la realización de un videojuego es el diseño de la interfaz gráfica la cual proporciona al usuario un entorno visual sencillo para poder utilizar todas las funcionalidades del videojuego.

Una característica del sistema Android es que ofrece la posibilidad de crear las interfaces gráficas a través ficheros XML, lo que permite separar la funcionalidad de las clases de la estética. De esta forma, el código queda mucho más ordenado y claro, indicando en los archivos XML la distribución de nuestros objetos gráficos (botones, etiquetas, imágenes…), así como su color, forma, tamaño, colocación etc., mientras que en los archivos java podemos centrarnos en la funcionalidad de éstos y su integración en el sistema.

Estos ficheros se deben almacenar en el directorio */res/layout* de nuestro proyecto, como se puede observar la siguiente imagen:

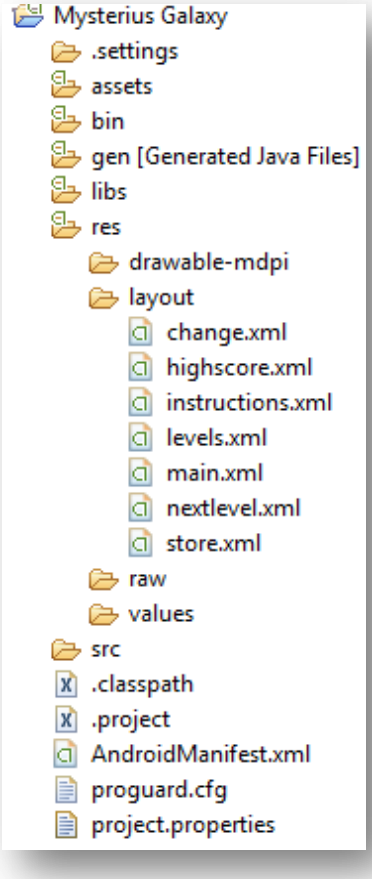


Figura 34. **Directorio *layout***

Para que el diseño creado en el fichero XML se presente como interfaz de una *Activity* hay que establecerlo como vista de dicha clase.

```
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
```

Código 4. **Establecer xml como vista**

Seguidamente se entrará a la explicación detallada de cada una de las clases que forman el paquete *view*, responsable de la parte lógica de la interfaz gráfica.

6.3.1 Clase Main

Es la clase que se ejecuta al iniciar la aplicación. Está compuesta por diferentes botones para acceder a las diferentes opciones que proporciona el programa. Todas las clases de este paquete extienden de la clase *Activity* que es componente que representa una ventana del terminal móvil.

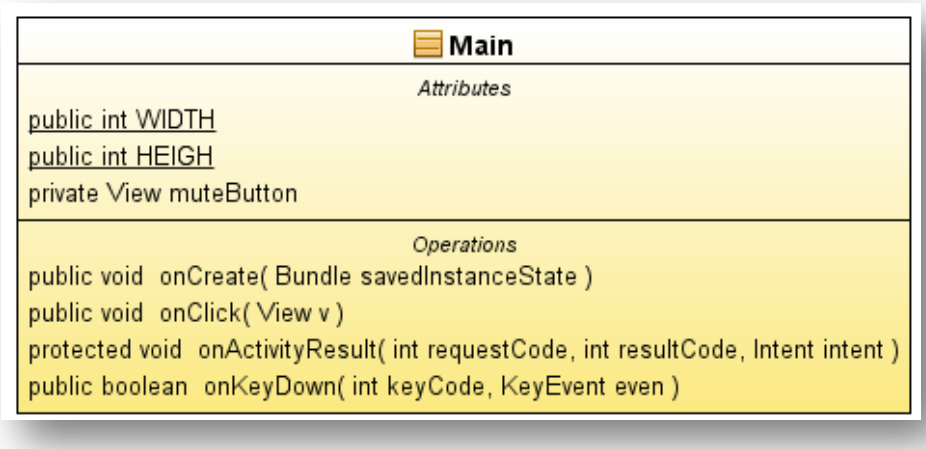


Figura 35. **Clase *Main***

Los métodos definidos para la clase *Main* son:

- **onCreate**: Inicia la reproducción de la música y establece los botones como escuchadores de esta *Activity*.
- **onClick**: Identifica el botón pulsado y ejecuta la acción correspondiente.
- **onKeyDown**: Cierra la aplicación y finaliza la reproducción de la melodía.

Figura 36. **Interfaz gráfica *Main***

6.3.2 Clase HighScore

Se encarga de presentar la puntuación máxima obtenida en los diferentes niveles.

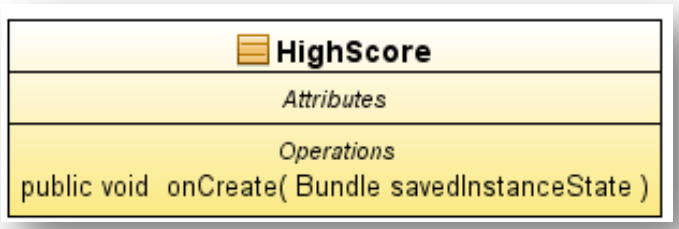


Figura 37. **Clase HighScore**

El método definido para la clase *HighScore* es:

- **onCreate**: Obtiene las puntuaciones de todos los niveles mediante una petición a la base de datos y se la asigna a las etiquetas correspondientes definidas en el fichero XML.

6.3.3 Clase Instructions

Muestra toda la información de interés necesaria para que el jugador conozca el funcionamiento de la aplicación.

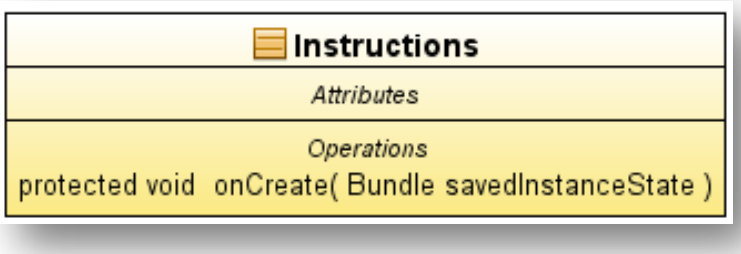


Figura 38. **Clase Instructions**

El método definido para la clase *Instructions* es:

- **onCreate**: Se limita a establecer el fichero *instructions.xml* como vista para la *Activity*, ya que no realiza ninguna otra función que cargar los textos e imágenes de dicho fichero.

6.3.4 Clase Levels

Es la interfaz a la que se accede al pulsar el botón *play* de *Main*. En esta ventana se encuentran representados todos los niveles del juego mediante un botón. En el caso de estar desbloqueado, en el interior de cada botón aparece el número del nivel al cual corresponde y por el contrario si está bloqueado se mostrará un aspa.

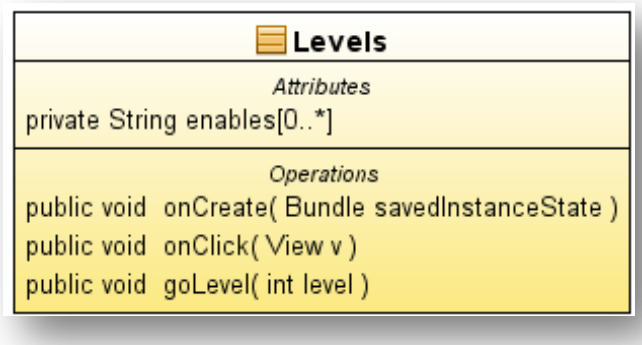


Figura 39. **Clase *Levels***

Los métodos definidos para la clase *Levels* son:

- **onCreate**: Mediante una consulta a la base de datos obtiene la disponibilidad de cada nivel y asigna el valor correspondiente al texto de cada botón.
- **onClick**: Identifica el botón seleccionado y hace una llamada al método *goLevel* indicando el nivel elegido.
- **goLevel**: Comprueba si el nivel esté disponible y en tal caso llama a la *Activity* encargada de la ejecución del juego, *Game*, informando del nivel elegido.

6.3.5 Clase Game

Es la clase que gestiona el ciclo de vida de la interfaz gráfica del videojuego. En ella se vuelca la interfaz pero no realiza ninguna función de la lógica del juego. Es la única clase de este paquete que no tienes un fichero XML relacionado porque la interfaz se representa mediante un *Canvas*.

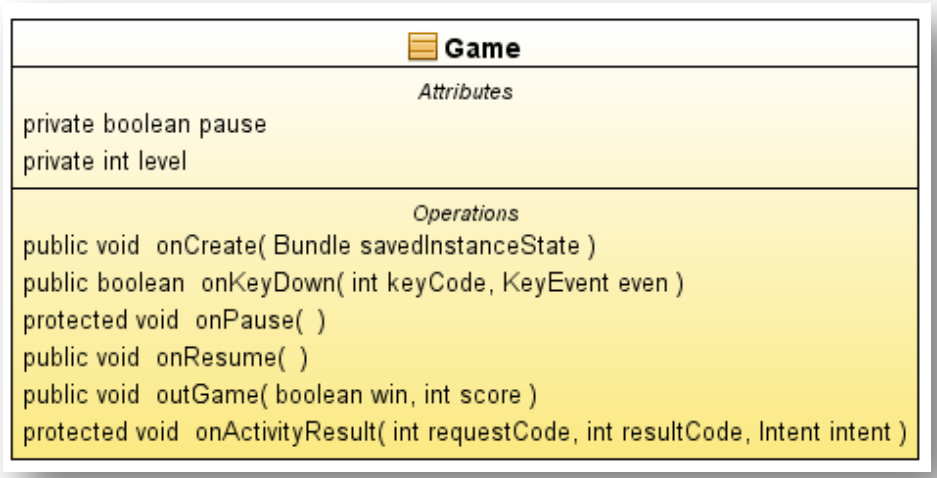


Figura 40. **Clase *Game***

Los métodos definidos para la clase *Game* son:

- **onCreate**: Inicia el hilo principal del videojuego.
- **onKeyDown**: En caso de pulsar el botón retroceder se llama al método *onPause* para que se detenga el videojuego.
- **onPause**: Hace que la ejecución del videojuego se pare y se llame a *outGame* para configurar la *Activity* que debe aparecer en caso de pausa.
- **onResume**: Reanuda el videojuego por el punto en el que se dejó.
- **outGame**: Llama a la *Activity NextLevel* indicando si el juego se ha detenido porque el jugador ha ganado, perdido o pausado la partida. Además le envía la puntuación obtenida para que en los dos primeros casos las muestre por pantalla.

- **onActivityResult**: Finaliza la *Activity* y el hilo del juego. Dependiendo de los parámetros recibidos inicia la clase *Game* o *Main*.

Figura 41. **Interfaz gráfica *Game***

6.3.6 Clase NextLevel

La interfaz gráfica de esta clase puede variar bastante, ya que se ha adaptado para que se pueda utilizar en varias situaciones:

- El jugador gana la partida

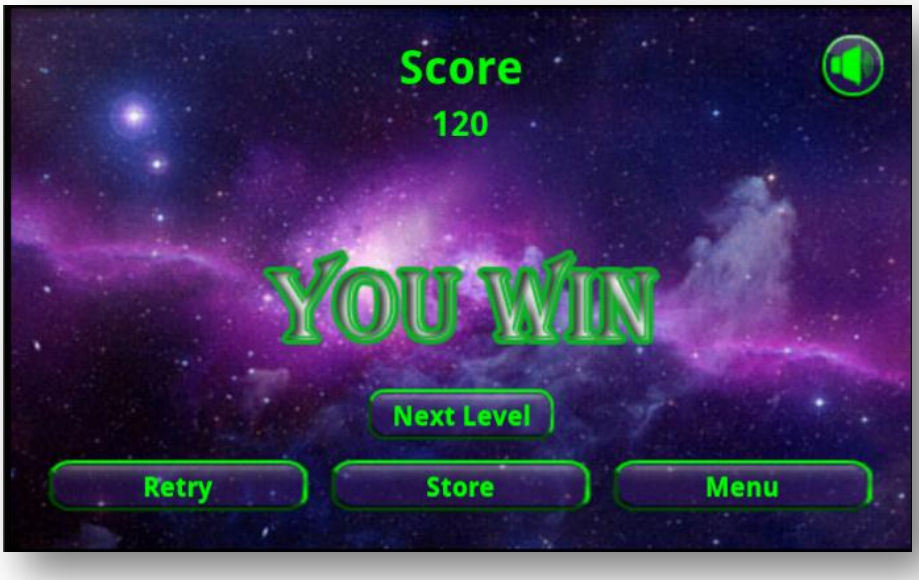


Figura 42. **Interfaz gráfica *NextLevel-win***

- El jugador pierde la partida.

Figura 43. **Interfaz gráfica NextLevel-game over**

- El jugador pause la partida.

Figura 44. **Interfaz gráfica *NextLevel-pause***

Con el fin de conseguir que la interfaz se adapte a las diferentes situaciones lo que se ha hecho es definir todos los elementos en el fichero *nextlevel.xml* pero dependiendo de los parámetros pasados en la llamada de la clase, unos componentes de la interfaz pueden ser ocultados y otros modificados.

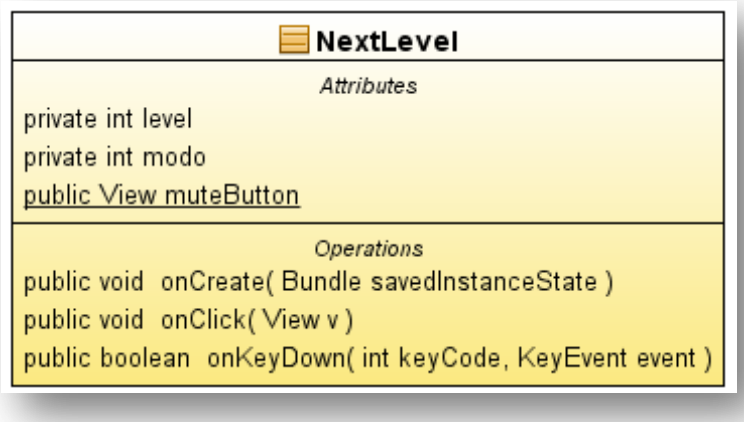


Figura 45. **Clase NextLevel**

Los métodos definidos para la clase *NextLevel* son:

- **onCreate**: Se encarga de modificar la interfaz dependiendo del contexto de la llamada. Además, comprueba si la puntuación obtenida es mayor que la almacenada en la base de datos y en tal caso la actualiza.
- **onClick**: Redirige la aplicación a la *Activity* correspondiente dependiendo del botón pulsado.
- **onKeyDown**: Se inicia la clase *Main* si la tecla pulsada es retroceder.

6.3.7 Clase Store

Esta clase también tiene una doble funcionalidad dependiendo del *radioButton* seleccionado:

- **Buy**: Se pueden comprar los diferentes disparos y armas especiales. Las armas especiales se comprar para un único uso y las armas convencionales para siempre.
- **Select**: EL jugador puede elegir qué disparo y arma especial va a utilizar durante la partida. Mientras se está jugando se puede acceder a esta pantalla para cambiar las opciones.

En ambos casos en la parte inferior se muestran la cantidad de puntos de los que dispone el jugador y el botón que da acceso a la compra de puntos.

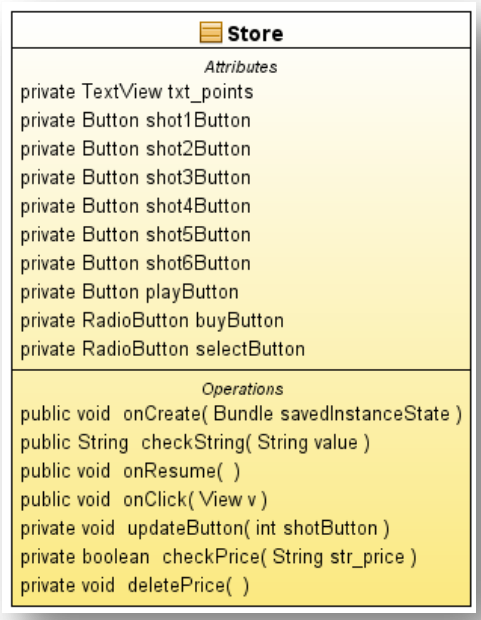


Figura 46. **Clase *Store***

Los métodos definidos para la clase *Store* son:

- **onCreate**: Asigna el fichero *store.xml* y llama al método *deletePrice*.
- **checkString**: Comprueba el valor del *string* pasado.
- **onResume**: Actualiza el valor de los puntos del jugador.
- **onClick**: Trata todas las posibles situaciones que se pueden dar con los diferentes botones.
- **updateButton**: Se utiliza para mantener pulsado el botón del arma seleccionado y desactivar el resto.
- **checkPrice**: Comprueba si se dispone de los puntos suficientes para realizar la compra.
- **deletePrice**: Elimina el precio de los disparos que ya han sido adquiridos.

6.3.8 Clase Change

Clase que se comunica con el paquete de Paypal para realizar las transacciones monetarias. Existen diferentes cantidades entre las que el usuario puede elegir.

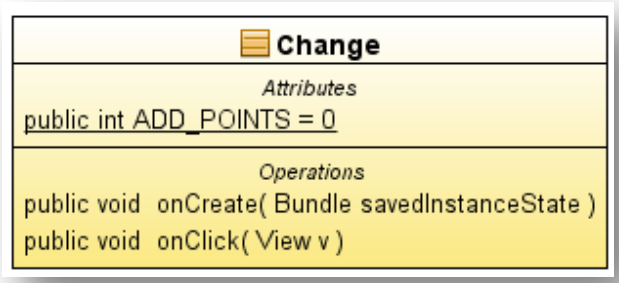


Figura 47. **Clase *Change***

Los métodos definidos para la clase *Change* son:

- **onCreate**: Configura la conexión con Paypal.
- **onClick**: Envía los datos para la transacción: tipo de moneda, cuenta del destinatario, concepto e importe.

6.4 Lógica del videojuego

En este apartado se detallará en la implementación de la parte lógica del videojuego: eventos táctiles, sistema de detección de colisiones, hilo principal, naves, base de datos, reproducción de audio y pago mediante Paypal.

Se ha optado por no utilizar ninguna librería disponible para el desarrollo de juegos fin de tener más control sobre las funcionalidades necesarias, adquirir un conocimiento más detallado de la implementación de videojuegos y poder la posibilidad los recursos de forma óptima.

6.4.1 Naves

Dado que existen diferentes tipos de naves y que comparten muchas características se ha decidido implementarlas utilizando herencia, de este modo se ahorra en código y tiempo de desarrollo.

Clase Ship

Es la clase padre de todas las naves. En ella se definen los atributos y métodos comunes.

Los atributos definidos para la clase *Ship* son:

- **x - y**: Indican la posición de la nave.
- **explosion**: Array con las imágenes que se sucederán en el caso de que una nave sea destruida.
- **nboom**: Índice de la imagen de *explosion* que hay que dibujar en pantalla.
- **imagebullets**: Array con las imágenes de las balas que dispara una nave. En el caso de los enemigos sólo existirá una imagen, ya que por cada disparo sólo lanza una bala pero en el caso del protagonista puede variar el número de balas.
- **listShot**: Lista con los objetos *Shot* que la nave ha disparado.
- **moveYShot y moveXShot**: Velocidad de las balas.
- **life**: Número de impactos que puede sufrir.
- **oneMove**: Velocidad de la nave.

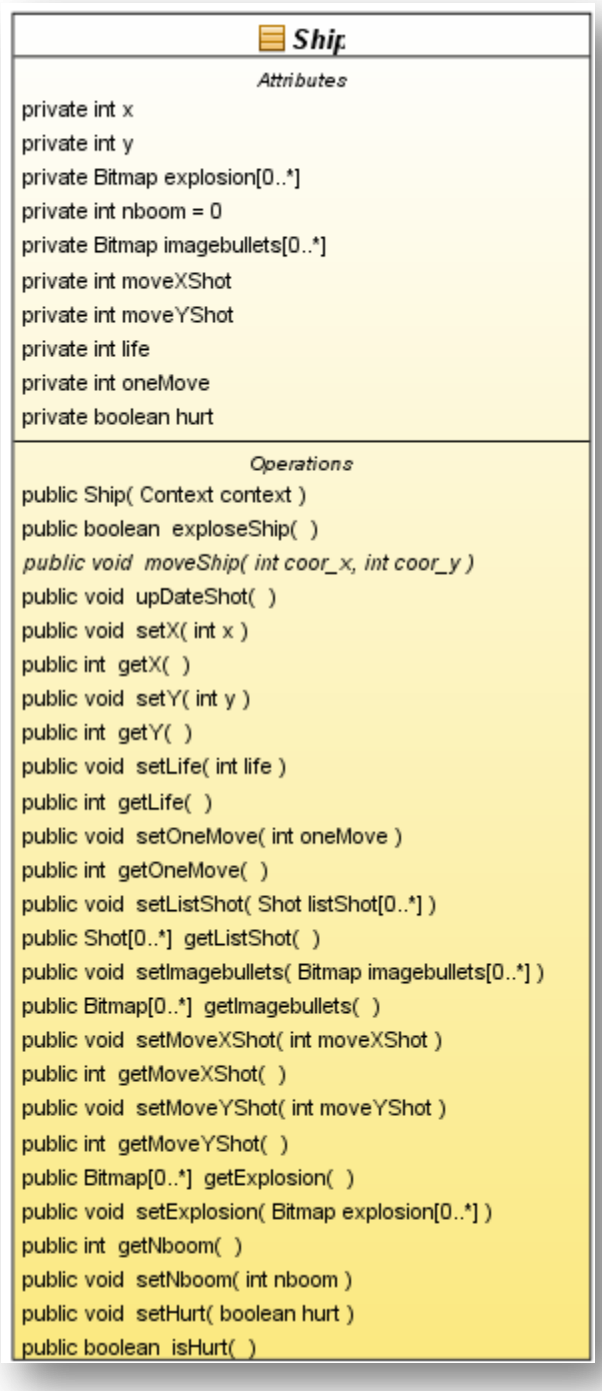


Figura 48. **Clase *Ship***

Los métodos definidos para la clase *Ship* son:

- **exploseShip**: Realiza la secuencia de imágenes de la explosión.
- **moveShip**: Método abstracto que se implementará en cada una de las clases finales.
- **upDateShop**: Se encarga del movimiento de las balas. Cuando una de éstas sale de las coordenadas de la pantalla se llama al *garbage collector* para liberar memoria.
- **Métodos get y set**: Permiten acceder a los atributos privados de la clase.

Clase Enemy

Subclase de *Ship* y a su vez superclase de *Enemy1*, *Enemy2* y *Enemy3*. En esta clase se engloban los métodos y atributos comunes de las naves enemigas pero únicamente los que no se han podido incluir en *Ship*, ya que, no son propias de la nave del jugador.

Los atributos definidos para la clase *Enemy* son:

- **imageShip**: Imagen con la que se representa el enemigo.
- **nShot**: Array con las coordenadas en las que realiza un disparo.
- **moveX y moveY**: Especifica la dirección de desplazamiento de la nave.
- **value**: Puntuación que obtiene el jugador al destruirla.
- **nBullet**: Número de veces que dispara.

Los métodos definidos para la clase *Enemy* son:

- **moveShip**: Realiza el movimiento de los enemigos dependiendo de los valores de los atributos.
- **setnShot**: Calcula las coordenadas en las que se van a realizar los disparos.
- **Métodos get y set**: Permiten acceder a los atributos privados de la clase.

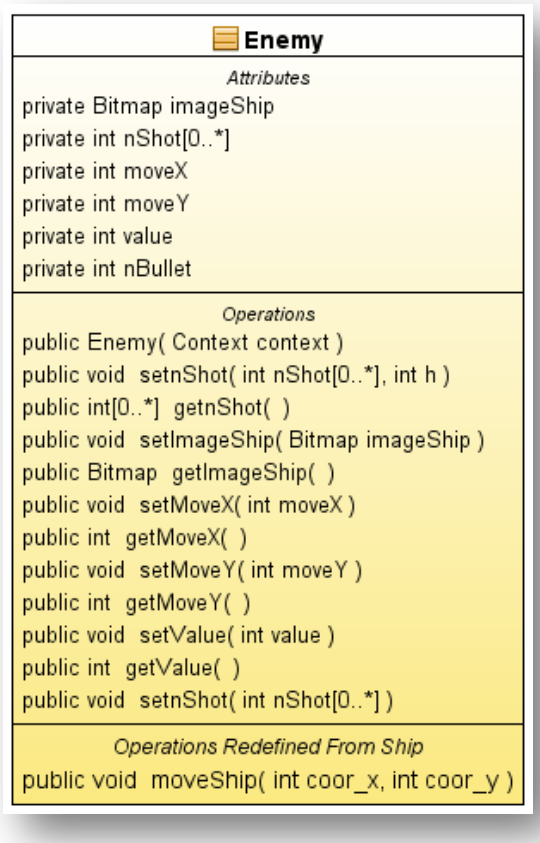


Figura 49. **Clase *Enemy***

Clases Enemy1, Enemy2 y Enemy3

Son las clases finales de las naves enemigas. Debido a la jerarquía de clases en éstas sólo hace falta configurar los valores adecuados para que se diferencien unas de otras. Una de las ventajas que ha proporcionado la implementación empleando herencias es la facilidad con la que se pueden crear nuevos enemigos.

- **Enemy1**: Su desplazamiento es horizontal, realiza 3 disparos y el jugador obtiene 10 puntos por destruirla.

Figura 50. **Ememy1**

- **Enemy2**: Su desplazamiento es diagonal, realiza 2 disparos y el jugador obtiene 20 puntos por destruirla.

Figura 51. **Enemy2**

- **Enemy3**: Su desplazamiento es vertical, realiza 4 disparos y el jugador obtiene 30 puntos por destruirla.

Figura 52. **Enemy3**

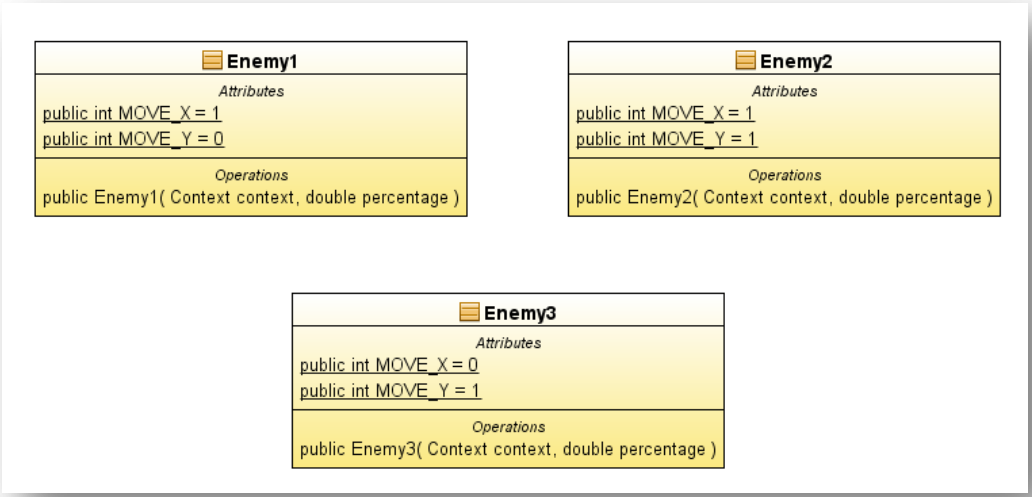


Figura 53. **Clases Enemy1, Enemy2 y Enemy3**

Clase ContainsEnemy

Esta clase facilita el manejo de las naves enemigas en el hilo principal de juego. Crea una lista con todos los enemigos que están en pantalla, para que el hilo principal no trate con cada una de las naves enemigas de forma individual, sino que mediante *ContaisEnemy* indica a todas las naves que tienen que moverse o mostrarse en pantalla.

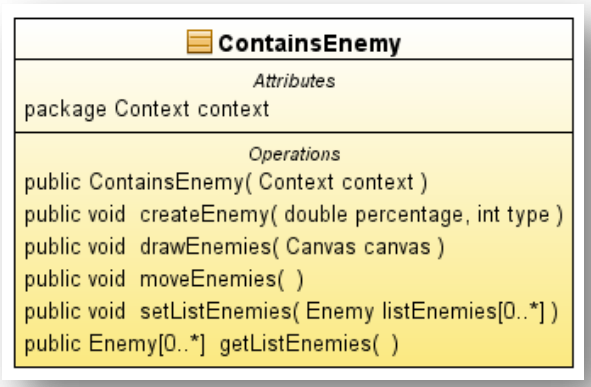


Figura 54. **Clase ContainsEnemy**

Los métodos definidos para la clase *ConstainsEnemy* son:

- **createEnemy**: Crea el tipo de enemigo que se le indica.
- **drawEnemies**: Método utilizado para pintar las naves enemigas en el *canvas* que se utiliza para la visualización del juego.
- **moveEnemies**: Actualiza la posición de toda las naves enemiga y sus disparos. Si alguna de la naves sale del campo de visión del jugador se elimina de la lista y se llama al *garbage collector* para liberar memoria.
- **getListEnemies y setListEnemies**: Permiten acceder a la lista de enemigos.

Clase Player

En cuanto a la lógica del juego, *Player* es la clase más importante junto a *ThreadGame*. Esto es así porque no sólo se encarga del control de la nave del jugador sino que también es responsable del sistema de detección de colisiones.

La implementación de esta nave no es tan simple como la de los enemigos, ya que ésta tiene que responder ante los eventos táctiles, permite usar diferentes disparos, incorpora la posibilidad de emplear armas especiales y está representada por una secuencia de imágenes.

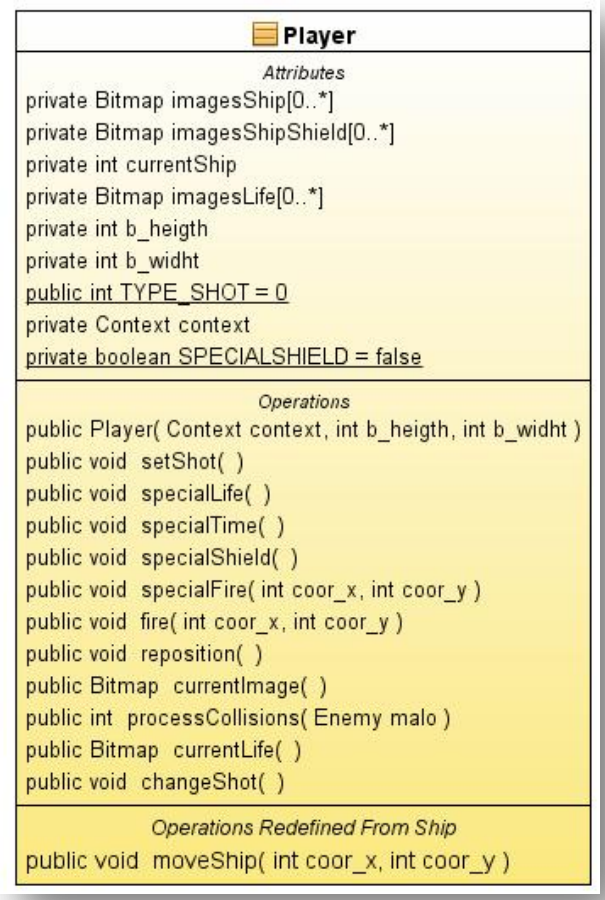


Figura 55. **Clase *Player***

Los atributos definidos para la clase *Player* son:

- **imagenesShip**: Conjunto de imágenes que dota a la nave de sensación de movimiento.

- **imagenesShipShield**: Conjunto de imágenes de la nave con el escudo del arma especial.

Figura 56. **Imágenes de la nave con escudo**

- **currentShip**: Número de la imagen actual de la nave.
- **imagenesLife**: Imágenes de la barra de vida del jugador.
- **b_height y b_width**: Altura y anchura de los botones para identificar si el evento táctil se ha producido en el área de alguno de los botones.
- **context**: Contexto de la aplicación para poder obtener las imágenes de la carpeta correspondiente.
- **TYPE_SHOT**: Indica el tipo de disparo que está utilizando el jugador.
- **SPECIALSHIELD**: Indica si esta activo el escudo para utilizar correctamente la secuencia de imágenes que representa la nave.

Los métodos definidos para la clase *Player* son:

- **setShot**: Establece según el valor de *TYPE_SHOT* el tipo de disparo que ha seleccionado el jugador.
- **specialLife**: Se le llama cuando el jugador utiliza el arma especial *Life*. La función de esta arma es regenerar la vida de la nave aumentando así el número de impactos necesarios para que finalice la partida.
- **specialTime**: Se le llama cuando el jugador utiliza el arma especial *Time*. Esta arma paraliza todas las naves y sus disparos durante 15 segundos, mientras que el jugador puede desplazarse, eliminar enemigos y esquivar balas.
- **specialShield**: Se le llama cuando el jugador utiliza el arma especial *Shield*. Consiste en un escudo protector que impide que los impactos, ya sea de una bala o un enemigo, causen daño a la nave.
- **specialFire**: Identifica el arma especial que el jugador seleccionó, comprueba que tenga unidades disponibles y en tal caso llama al método correspondiente.
- **fire**: La nave realiza un disparo del tipo seleccionado.

- **moveShip**: Identifica que botón de movimiento es la que ha sido pulsada y desplaza la nave en consecuencia.
- **reposition**: Se encarga de volver la nave a la posición inicial después de un desplazamiento lateral.
- **currentImage**: Devuelve la imagen actual teniendo en cuenta si tiene el escudo activo o ha explotado.
- **currentLife**: Accede a la imagen actual de la barra de vida según los impactos recibidos.
- **changeShot**: Permite cambiar el tipo de disparo que utiliza el jugador.
- **processCollisions**: Método responsable de la identificación de colisiones. Éste se tratará más adelante en detalle.

Clase Shot

Esta clase representa los disparos realizados por las naves, por eso el atributo *listShot* es una lista de objetos de esta clase.

Los atributos definidos para la clase *Shot* son:

- **images**: Imágenes que conforman el disparo.
- **p**: coordenadas de cada una de las balas que componen el disparo.
- **nbullet**: Número de balas que forman el disparo.

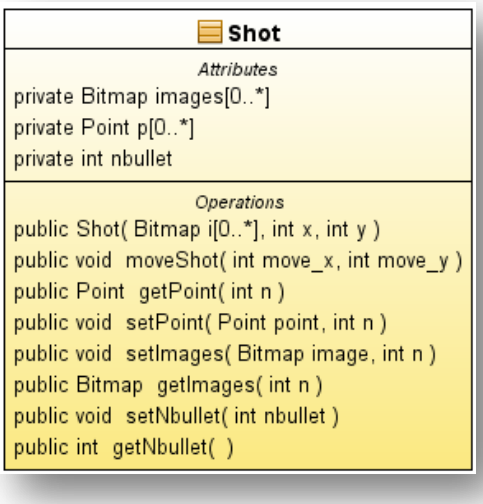


Figura 57. **Clase *Shot***

Los métodos definidos para la clase *Shot* son:

- **moveShot**: Realiza el desplazamiento de las balas.
- **Métodos get y set**: Permiten el acceso a los atributos de la clase.

6.4.2 Sistema de colisiones

Como ya se comentó anteriormente las posibles colisiones son:

- Protagonista - bala enemiga
- Protagonista – enemigos
- Bala protagonista – enemigo
- Escudo - enemigo
- Escudo-bala enemiga.

Se ha considerado que se produce una colisión cuando dos pixeles de las imágenes mencionadas se superponen.

El sistema de colisiones se ha centralizado en un único método, *processCollisions* de la clase *Player,* en el que se comprobará si ha sucedido alguna de las colisiones posibles. A este método se le llama desde el hilo principal pasándole como parámetro un objeto *Enemy* almacenado en *ContainsEnemy*.

Los pasos que realiza el método *processCollisions* son:

1. Comprueba si el enemigo está en fase de explosión, en tal caso no se consideran las colisiones.
2. Colisión de las bala de jugador ha colisionado con el enemigo. En dicha situación, se destruye la nave, se elimina la bala, se obtiene la puntuación y se llama al *garbage collector* para liberar memoria.
3. Comprueba si el protagonista está en fase de explosión, ya que en tal caso no se consideran el resto de colisiones.
4. Colisión de las balas enemigas y el jugador, en cuyo caso se elimina la bala, si el jugador no tiene activado el escudo se reduce su vida y se llama al *garbage collector*.
5. Colisión entre la nave protagonista y la enemiga. Una vez detectada esta situación, se destruye la nave enemiga, la del jugador reduce la vida, se obtiene la puntuación y se llama al *garbage collector*.

6.4.3 Hilo principal

La clase *TheadGame* es el hilo principal de la lógica del juego. Se encarga de pintar la interfaz gráfica, los eventos táctiles, la creación de los enemigos y coordinar la interacción de todas las clases.

Al instanciar la clase se crean dos hilos:

- **loadThread**: Se encarga de leer la tabla de datos del nivel correspondiente y crea los enemigos en los instantes de tiempo establecidos en dicha tabla. Si el protagonista es destruido o si se llega al final de la partida destruye el hilo principal y llama a la *Activity NextLevel* para finalizar la pantalla.
- **Hilo principal**: En primer lugar, comprueba si se ha generado un evento táctil, en cuyo caso llama al método correspondiente con la acción del botón pulsado. Posteriormente, comprueba si se ha producido una colisión y de ser así qué elementos tienen que iniciar la secuencia de explosión. Finalmente, se actualiza el *canvas* donde se dibujan todos los elementos de la interfaz gráfica.

6.4.4 Evento táctil

En la implementación de los eventos táctiles se ha optado por emplear el método *onTounchEvent*. Este método no soporta *multitouch*, pero se ha escogido por su compatibilidad con versiones anteriores de Android que no permiten el reconocimiento de varios eventos táctiles simultáneos. Aunque esto no es un requisito del proyecto se ha considerado adecuado facilitar el uso del videojuego para compatibilidad del sistema operativo hacia atrás.

Los eventos táctiles se clasifican en 4 grupos:

- **Fuera de los límites de los botones**: El videojuego no realiza ninguna acción.
- **Botón de disparo**: Se tendrá que espera 0,2 segundos hasta que se vuelva a poder utilizar.
- **Botón de arma especial**: Se tendrá que espera 0,4 segundos hasta que se vuelva a poder utilizar.

- **Botones de dirección**: La nave se desplazará mientras se mantenga el botón pulsado, siempre y cuando no salga de la pantalla.

6.5 Herramientas

Además de las clases ya mencionadas hasta ahora, también se ha utilizado otras para completar todas las funcionalidades de la aplicación.

6.5.1 Reproductor

La reproducción del audio del juego se ha implementado con un componente *Service* ya que de este modo la reproducción no irá vinculada con ninguna *Activity* y se escuchará en cualquier situación. Además, la implementación se ha apoyado en la clase *MediaPayer* que proporciona Android para facilitar el tratamiento de contenido multimedia.

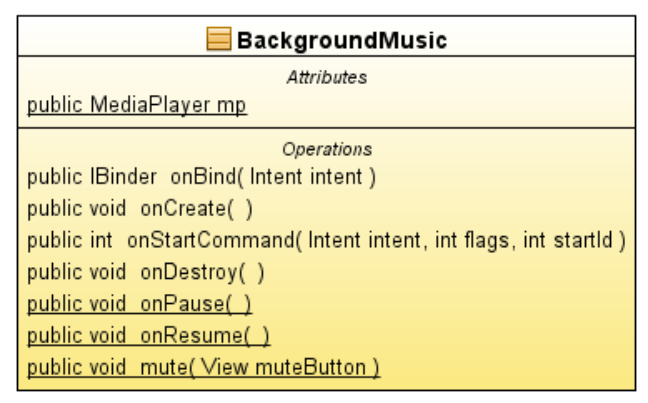


Figura 58. **Clase BackgroundMusic**

Los métodos definidos para la clase *BackgroundMusic* son:

- **onCreate**: Crea el objeto *MediaPlayer* con el fichero de audio y lo configura como reproducción en bucle.
- **onStartCommand**: Inicia la reproducción.
- **onDestroy, onPause y onResume**: Controla el ciclo de vida.
- **mute**: Invierte el estado de reproducción del audio y modifica la imagen del botón mute.

6.5.2 Base de datos

En la gestión de la base de datos se ha utilizado la clase *SQLiteOpenHelper*. Esta clase permite acceder de un modo fácil y seguro a la base de datos.

La clase *Main* tiene como atributo público una instancia de la clase *DataBase* que hereda de *SQLiteOpenHelper*; de este modo desde cualquier parte del programa se puede acceder a la base de datos. Para garantizar la estabilidad de esta parte en este proyecto se ha optado por realizar en *DataBase* un método por cada una de las diferentes peticiones requeridas por el programa, logrando así que todas las peticiones *sql* estén centralizadas.

6.5.3 Paypal

A continuación, se explicarán los pasos que se han llevado a cabo para la implementación del medio de pago mediante Paypal:

1. Declaración la librería *Mobile Payments Liblary* en el manifiesto como una actividad.

```
<activity android:name="com.paypal.android.MEP.PayPalActivity"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"
    android:configChanges="keyboardHidden|orientation"/>
```

Código 5. **Declaración de Mobile Payments Liblary**

2. Declaración de permisos de internet y estado del teléfono.
3. Inclusión del fichero *PayPal_MPL.jar* como librería.
4. Importación de las clases de la librería en la clase *Change*.
5. Creación del objeto Paypal. Para ello hay que indicar el id de la aplicación y en qué modo se quiere usar Paypal: modo real, modo servidor de pruebas o modo desconectado. En este caso se ha utilizado el servidor de pruebas, quedando de la siguiente forma:

```
PayPal.initWithAppID(this, "APP-80W284485P519543T", PayPal.ENV_SANDBOX);
```

Código 6. **Inicialización Paypal**

6. Creación de un objeto *PayPalPayment* con los datos de la transacción.

```
PayPalPayment payment = new PayPalPayment();
payment.setSubtotal(amount);
payment.setCurrencyType("EUR");
payment.setRecipient("oscarm_1345999827_biz@gmail.com");
payment.setMerchantName("Mysterious Galaxy");
```

Código 7. **Datos de la transacción**

7. Llamada a la *Activity* de la librería Paypal para que el usuario introduzca sus datos.

```
Intent checkoutIntent = PayPal.getInstance().checkout(payment, this, new ResultDelegate());
startActivity(checkoutIntent);
```

Código 8. **Inicio Activity Paypal**

8. Creación de la clase *ResultDelegate*, pasada como parámetro en el apartado anterior. Dicha clase, que implementa *PayPalResultDelegate,* recibe la contestación de la transacción por parte de Paypal que puede ser: cancelado, fallido o realizado. En nuestro caso sólo se ha utilizado *onPaymentSucceeded* para incorporar los puntos comprados a los anteriormente obtenidos.

Las siguientes figuras muestran el proceso de compra:

- Introducción de la cuenta del cliente.

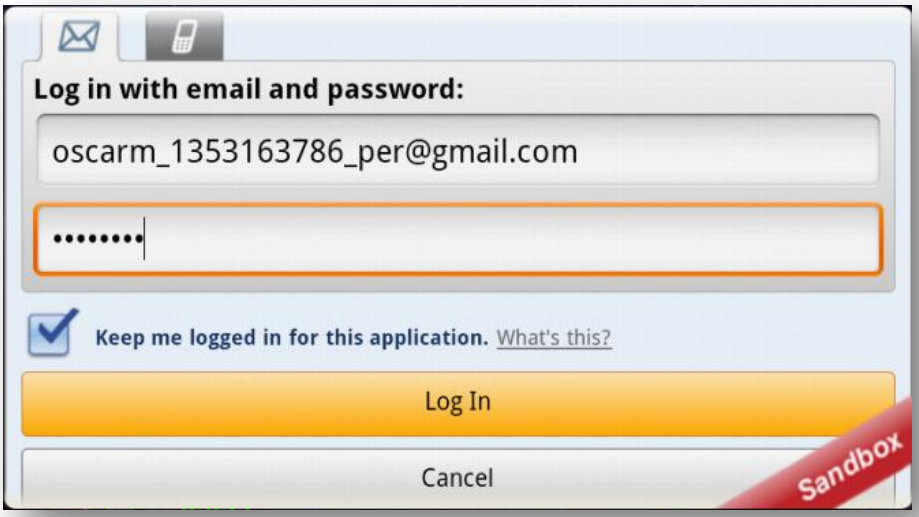


Figura 59. **Login Paypal**

- Comprobación de la cuenta.

Figura 60. **Comprobación del *Login***

- Datos de la transacción.

Figura 61. **Transacción**

- Tramitación de la transacción.

Figura 62. **Tramitación de la transacción**

- Pago aceptado

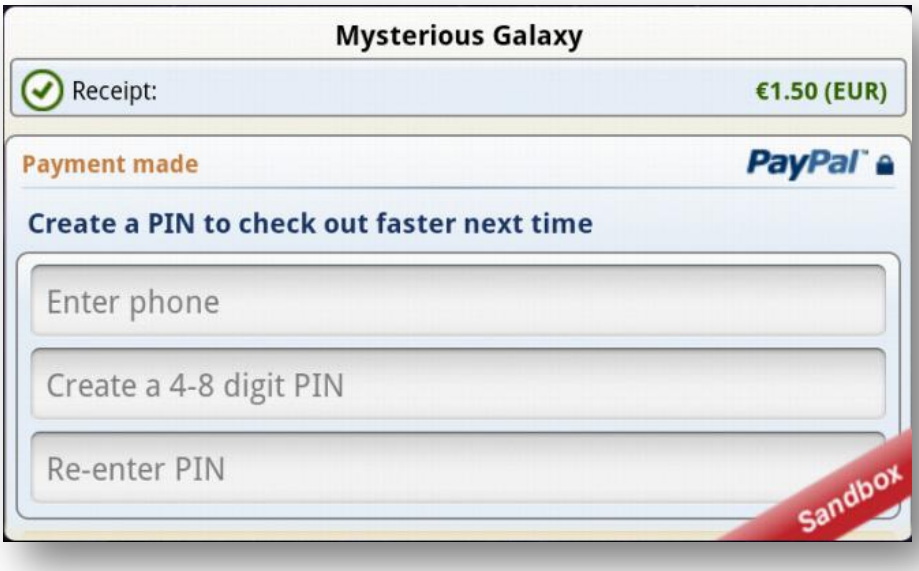


Figura 63. **Pago aceptado**

7. Pruebas

En este capítulo se explican las pruebas a las que se ha sometido el videojuego y que han determinado el correcto funcionamiento de éste. Durante el desarrollo del videojuego se han llevado a cabo pruebas en los diferentes módulos para garantizar que está exento de errores. Además de estas pruebas, una vez finalizado el programa se han realizado las siguientes pruebas de integridad. Estas se han dividido en tres bloques.

Videojuego

Como núcleo principal del programa es indispensable que se compruebe la ausencia de errores para que el jugador disfrute de una buena experiencia.

- **Desplazamiento de la nave**: Comprobación del movimiento de la nave al pulsar los diferentes botones de desplazamiento sin que ésta salga de los límites de la pantalla y realizando correctamente la secuencia del *sprite*.
- **Disparos**: Ejecución correcta de los diferentes disparos, siendo acorde con el seleccionado.
- **Armas especiales**: Se realizan adecuadamente las diferentes armas especiales: se paralizan las naves enemigas y las balas, se recupera la totalidad de la vida y el escudo evita los dañas de la nave del jugador en caso de colisión.

- **Enemigos**: Verificación de la carga de los enemigos, su desplazamiento y sus disparos. También se ha probado que se incremente la puntuación cuando un enemigo es destruido.
- **Sistema de colisiones**: Las colisiones posibles se realizan correctamente llevando a cabo según la situación la explosión de la nave o la reducción de la barra de vida.

Tienda

La tienda es la parte del proyecto en la que se ha desarrollado el aspecto comercial de la aplicación por lo que la ausencia de fallos es muy importante.

- **Compra de armas**: Comprobación de la correcta incorporación a la base de datos las armas compradas y actualización del número de puntos tras realizar el pago por el arma.
- **Elección de armas**: Verificación de la concordancia de las armas seleccionadas en la tienda con las que se emplean durante el juego.
- **Compra de puntos**: Se ha comprobado mediante cuentas de desarrollo de Paypal el correcto funcionamiento de las transacciones que tiene como consecuencia el aumento de los puntos del jugador.

Menú inicial

Batería de pruebas orientadas a comprobar las funciones a las que se puede acceder desde el menú de inicio exceptuando la tienda.

- **Música del juego**: La melodía del juego se inicia correctamente al comienzo del juego. Además, se detiene y reanuda en función del botón de mute.
- **Puntuaciones**: Al acceder a la venta de puntuaciones estas se cargan adecuadamente desde la base de datos.
- **Menú de niveles**: Los niveles del juego permanecen bloqueados hasta haber superado el nivel anterior.

8. Conclusiones y líneas futuras

8.1 Conclusiones

Tras haber concluido el desarrollo de este Proyecto Fin de Carrera, es el momento de efectuar el análisis y balance del resultado final obtenido.

Se ha conseguido el objetivo principal del presente Proyecto Fin de Carrera, desarrollar un videojuego de estética y funcionalidad retro. El estudio del jugador objetivo y los videojuegos de la década de los 80 y 90, ha permitido la creación de un juego en formato 2D con scroll horizontal que permite la estética, funcionalidad y diversión de muchos de los videojuegos presentes en esa época.

Además, teniendo en cuenta las características de las actuales tiendas de aplicaciones de smartphones (App Store y Google Play) junto con la posibilidad de hacer el juego comercial se han incorporado características de compra de objetos dentro del videojuego (*App-In-Purchase*) con los que poder, por un lado, aumentar las capacidades del jugador y, por otro, obtener beneficios para el creador.

Asimismo, se ha alcanzado el segundo objetivo, el estudio de la plataforma Android centrándose en la programación de videojuegos. La documentación que Google pone a disposición del desarrollador junto con el elevado número de foros y libros existentes sobre Android me han permitido adquirir los conocimientos para implementar elementos característicos de los videojuegos como: hilos, eventos táctiles, animaciones de *sprites* o el sistema de colisiones.

Finalmente, mediante la estructura de la memoria y la detallada explicación del proceso de creación de Mysterious Galaxy se ha conseguido que este Proyecto Fin de Carrera pueda ayudar a futuros alumnos a crear un videojuego para Android.

8.2 Líneas futuras

Una vez terminado éste proyecto se sabe que existe un margen de mejora en muchos sentidos, así como con amplias posibilidades de ampliación por medio de extensiones de funcionalidad o utilidades complementarias. A continuación, se explicarán las posibles líneas futuras:

- **Nuevos niveles**: Aumento del número de niveles del juego, con esto se incrementaría el tiempo de juego del usuario y en consecuencia los ingresos por la de compra de puntos.
- **Nuevas armas especiales y disparos**: Una mayor oferta de armas espaciales y disparos también fomentaría la compra de puntos, ya que el usuario albergaría la curiosidad del funcionamiento de esas nuevas alternativas.
- **Publicidad**: La presente aplicación se podría dividir en dos, una gratuita que incluiría publicidad y otra por la que se tuviera que abonar 0,89 € para poder adquirirla y que estuviera libre de publicidad. Este sistema favorecería el carácter comercial del producto.
- **Inteligencia artificial de los naves**: Este es un punto que al atenderlo se daría una mayor dificultad al videojuego y que le dotaría además de un aspecto más realista.
- **Efectos sonoros y música de fondo**: Se podría dotar de diferentes melodías para amenizar las partidas y añadir efectos sonoros al destruir una nave, al emplear un arma especial o con los disparos de los enemigos.
- **Resoluciones**: Debido a que Misterioys Galaxy es una prueba de concepto la interfaz gráfica solo esta optimizada para el dispositivo Samsung Galaxy Mini por lo que sería necesario adaptarlo a los diferentes tamaños y resoluciones de pantallas que existen en el mercado.
- **Estandarización de las versiones del S.O.**: La aplicación está desarrollada para funcionar correctamente en la versión de Android 2.2. Habría que estudiar la compatibilidad con otras versiones del sistema operativo y adaptarlo para así poder llegar a un mayor número de usuarios.

- **Nuevas plataformas**: Por último, sería interesante portar este videojuego a otros sistemas operativos como iOS, BlackBerry OS o Windows Phone 7.

9. Planificación

La planificación de un proyecto es fundamental para el buen desarrollo de éste. Cumplir con los plazos previstos, saber qué partes son críticas si no se terminan a tiempo y a qué puede afectar, garantizará el resultado exitoso del mismo.

A Continuación, se presentara el listado de tareas en las que se ha dividido el proyecto. Para cada una de ellas se especifica la duración, la fecha de inicio y la fecha de finalización.

	Nombre de tarea	Duración	Comienzo	Fin
1	Documentacion inicial	10 días	26/03/12	06/04/12
2	Intalación y familalización con el entorno de desa	2 días	09/04/12	10/04/12
3	**Analisis**	**6 días**	**11/04/12**	**18/04/12**
4	Casos de uso y Diagrama de actividad	2 días	11/04/12	12/04/12
5	Requesitos de usuario	2 días	13/04/12	16/04/12
6	Requisitos de software	2 días	17/04/12	18/04/12
7	**Diseño**	**17 días**	**19/04/12**	**11/05/12**
8	Módulos de la aplicación	5 días	19/04/12	25/04/12
9	Elementos de la interfaz	10 días	26/04/12	09/05/12
10	Música	2 días	10/05/12	11/05/12
11	**Implementacion**	**90 días**	**14/05/12**	**17/09/12**
12	**Interfaz**	**33 días**	**14/05/12**	**28/06/12**
13	Menú inicial e instrucciones	4 días	14/05/12	17/05/12
14	Puntuaciones	2 días	08/06/12	11/06/12
15	Tienda y cambio	6 días	12/06/12	19/06/12
16	Niveles	2 días	20/06/12	21/06/12
17	Siguiente nivel	4 días	22/06/12	27/06/12
18	HITO 1: Interfaz	0 días	28/06/12	28/06/12
19	**Herramientas**	**15 días**	**18/05/12**	**07/06/12**
20	Reproducción de audio	3 días	18/05/12	22/05/12
21	Base de datos	12 días	23/05/12	07/06/12
22	**Lógica del videojuego**	**57 días**	**28/06/12**	**17/09/12**
23	Creación del hilo principal y presentacion	16 días	28/06/12	19/07/12
24	Naves	6 días	20/07/12	27/07/12
25	Eventos táctiles	5 días	30/07/12	03/08/12
26	Sistema de colisiones	5 días	06/08/12	10/08/12
27	HITO 2: Finalización de la primera pantalla	0 días	13/08/12	13/08/12
28	Generación automatica de pantallas	5 días	13/08/12	17/08/12
29	Carga de enemigos	5 días	20/08/12	24/08/12
30	HITO 3: Finalización de todas las pantallas	0 días	27/08/12	27/08/12
31	Ajustes de la interfaz	4 días	27/08/12	30/08/12
32	Armas especiales	7 días	31/08/12	10/09/12
33	Compra de puntos mediante Paypal	4 días	11/09/12	14/09/12
34	HITO 4: Incorporacion de las formas de pa	0 días	17/09/12	17/09/12
35	Fase de pruebas y mejoras	10 días	17/09/12	28/09/12
36	HITO 5: Vesión final	0 días	01/10/12	01/10/12

Figura 64. **Tareas del proyecto**

9.1 Diagrama de Gantt

El diagrama de *Gantt* sitúa de forma secuencial las diferentes etapas del proyecto a lo largo de una línea temporal que representa el tiempo total para la consecución del videojuego. En la siguiente gráfica refleja la planificación de éste, así como la fecha estimada para cada una de las tareas.

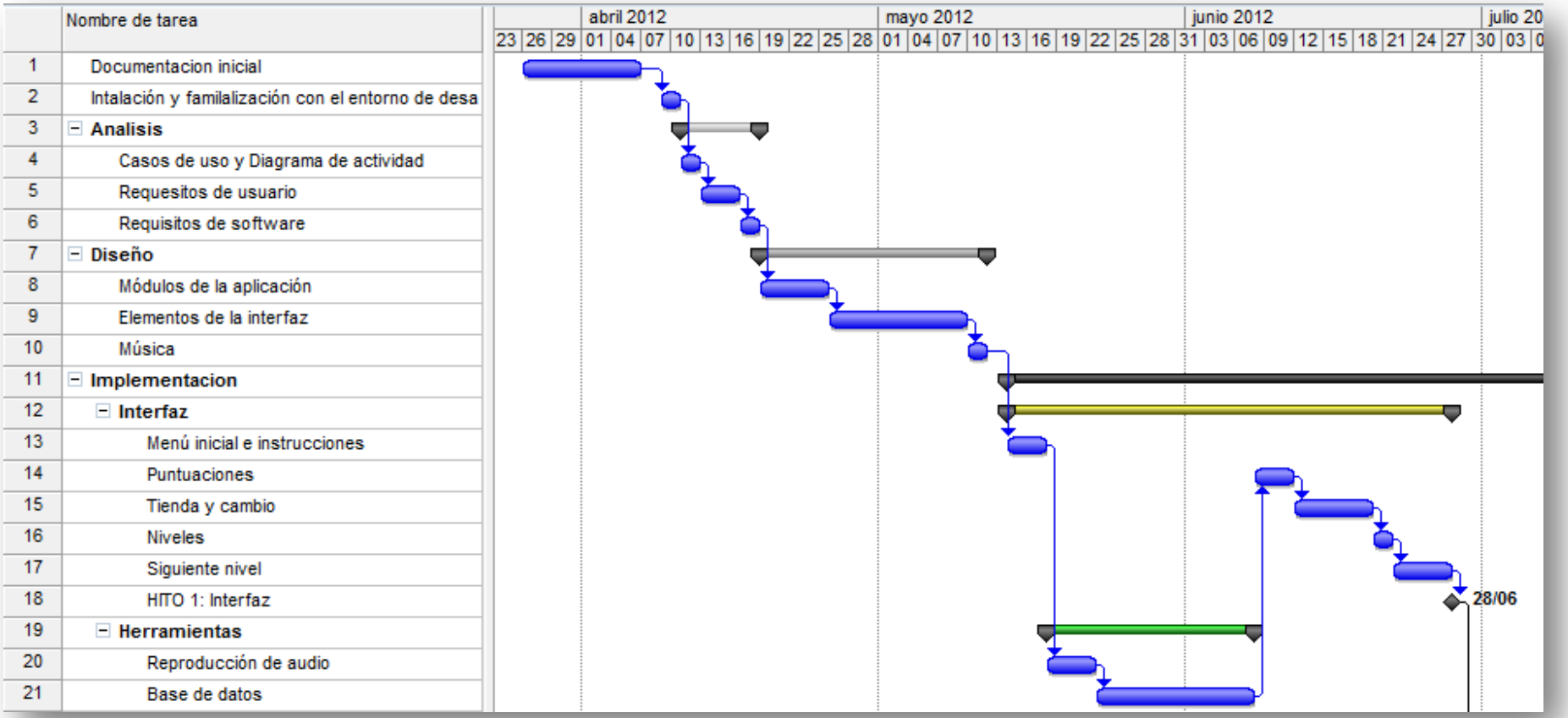


Figura 65. **Diagrama de *Gantt* 1**

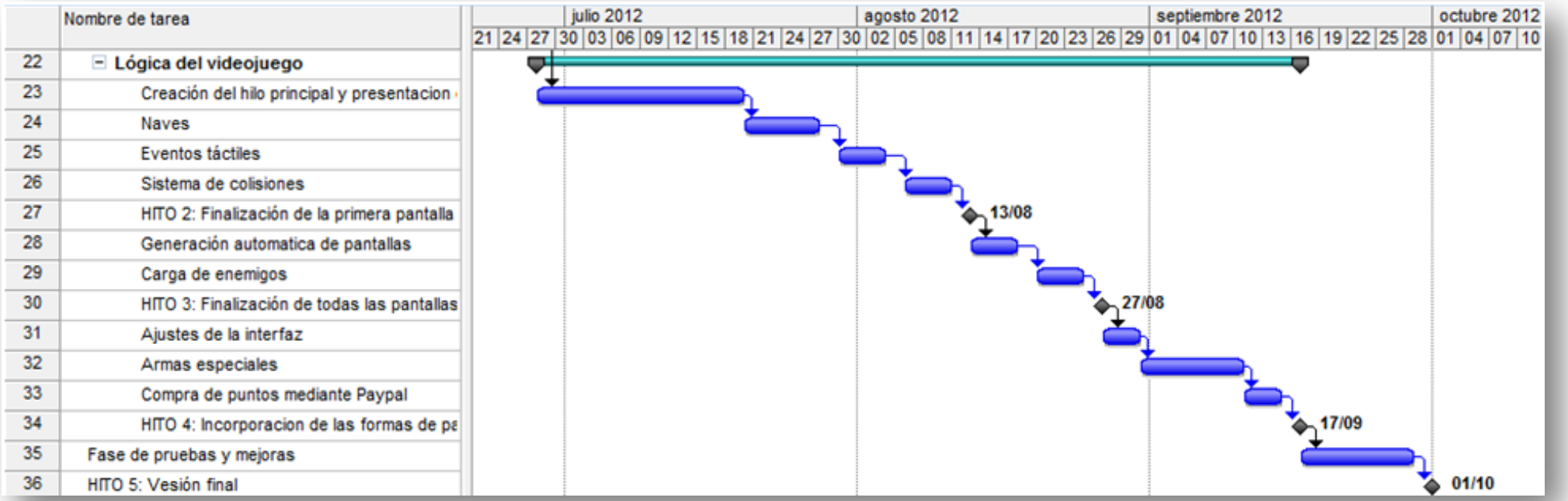


Figura 66. **Diagrama de *Gantt* 2**

10. Presupuesto

Atendiendo a la planificación se puede calcular a cuánto asciende el presupuesto del videojuego. El presupuesto se desglosa en costes de material y costes de personal según se detalla en los siguientes apartados.

10.1 Costes de equipamiento informático

Los materiales empleados para la realización del proyecto fueron los siguientes:

Concepto	Características
Ordenador de mesa	Fabricante: Packard Bell Procesador: Intel Core Quad Q6600 2.40 GHz Memoria RAM: 4,00 GB Sistema Operativo: Windows 7
Teléfono móvil	Fabricante: Samsung Modelo: Galaxy Mini Procesador: ARM v6 600 MHz Memoria RAM: 384 MB Sistema Operativo: Android 2.2 (Froyo)

Tabla 41. Materiales empleados

En la siguiente tabla se detalla el coste de equipamiento teniendo en cuenta el precio, el periodo de dedicación y el periodo de amortización de los mismos.

Concepto	Precio	Periodo dedicación	Periodo amortización	Coste amortización
Ordenador de mesa	1000,00 €	6 Meses	60 Meses	100 €
Teléfono móvil	170,00 €	5 Meses	12 Meses	71 €
Total	**1170,00 €**			**171 €**

Tabla 42. Costes de equipamiento

10.2 Costes de personal

La duración del desarrollo completo del videojuego Mysterious Galaxy asciende a un total de 139 días, lo que supone un total de 1112 horas de trabajo de un programador.

Cargo Profesional	Retribución (€/hora)	Horas trabajadas	Total
Programador	15 €	1080	16.200 €

Tabla 43. Costes de personal

10.3 Coste total

Finalmente en la siguiente tabla se establece el precio total de la aplicación.

Concepto	Precio
Costes materiales	171 €
Costes de personal	16.200 €
Subtotal	16.371 €
I.V.A (21%)	3.438 €
Total	19.809 €

Tabla 44. Coste final

11. Referencias

[1] Zorita Leza J., *El consumo en la industria del videojuego para el año 2011 crecerá un 10,4%,* Julio de 2011, disponible en http://www.vadejuegos.com/noticias/2011/07/06/el-consumo-en-la-industria-del-videojuego-para-el-ano-2011-ascendera-hasta-los-74000-millones-de-euros-093304.html (Consulta: 18/12/2012)

[2] Alonso D., *PlayStation Mobile aterriza en Android y PS Vita,* Octubre de 2012, disponible en http://www.hobbyconsolas.com/node/44005 (Consulta: 10/02/2013)

[3]Gartner, *Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012*; Egham, Reino Unido; Abril del 2011, disponible en http://www.gartner.com/it/page.jsp?id=1622614 (Consulta: 16/09/2012)

[4]Gartner, *Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012; Previous Year-over-Year Decline Occurred in Second Quarter of 2009*; Egham, Reino Unido; Mayo del 2012, disponible en http://www.gartner.com/it/page.jsp?id=2017015 (Consulta: 16/09/2012)

[5] Wikipedia, *Retrogaming,* Diciembre 2012, disponible en http://en.wikipedia.org/wiki/Retrogaming (Consulta: 22/12/2012)

[6] McFerran, D., *Convierte tu móvil Android en la máquina retro definitiva*, Diciembre 2011, disponible en http://www.eurogamer.es/articles/convierte-tu-movil-android-en-la-maquina-retro-definitiva-articulo (Consulta: 22/12/2012)

[7] Ramker, R., *Historia de los Videojuegos: El Origen y los Inicios, 2011,* disponible en http://www.otakufreaks.com/historia-de-los-videojuegos-el-origen-y-los-inicios, (Consulta: 23/09/2012)

[8] Velasco, J. J., *Historia de la tecnología: OXO, un videojuego para uno de los primeros computadores de la historia*, 2011, disponible en http://alt1040.com/2011/07/oxo-un-videojuego-para-uno-de-los-primeros-computadores-de-la-historia (Consulta: 23/09/2012)

[9] Uxio P. R.*, Videogame Culture: Magnavox Odyssey, la primera consola de la historia*, 2008, disponible en http://blogocio.net/videogame-culture-magnavox-odyssey-la-primera-consola-de-la-historia-sp-614/ (Consulta: 23/09/2012)

[10] Rodríguez S., *30 años de evolución en las consolas portátiles de Nintendo (Vol.1)*, 2012, disponible en http://juegos.es/analisis/30-anos-de-evolucion-en-las-consolas-portatiles-de-nintendo-vol-1 (Consulta: 23/09/2012)

[11] Rodríguez S., *30 años de evolución en las consolas portátiles de Nintendo (Vol.2)*, 2012, disponible en http://juegos.es/analisis/30-anos-de-evolucion-en-las-consolas-portatiles-de-nintendo-vol-2 (Consulta: 23/09/2012)

[12] Miltonshows, *Historia de las consolas portátiles*, disponible en http://www.retrogames.cl/portatiles.html (Consulta: 25/09/2012)

[13] Castillo J. C., *Los juegos móviles reportarán $10 mil millones en 2014*, Octubre de 2010, disponibles en http://alt1040.com/2010/10/los-juegos-moviles-reportaran-10000m-en-2014 (Consulta: 25/09/2012)

[14] Baz Alonso, A., Ferreira Artime, I., Álvarez Rodríguez, M., García Baniello, R., *Dispositivos Móviles*, Universidad de Oviedo, 2009, disponible en http://es.scribd.com/doc/72739868/Memoria (Consulta: 06/09/2012)

[15] Observatorio Nacional de las Telecomunicaciones y de la SI, *Evolución del número de clientes de telefonía móvil en España,* disponible en http://www.ontsi.red.es/ontsi/es/indicador/evoluci%C3%B3n-del-n%C3%BAmero-de-clientes-de-telefon%C3%ADa-m%C3%B3vil-en-espa%C3%B1a (Consulta: 06/09/2012)

[16] Colombia Digital, *Sistemas operativos móviles,* Enero de 2012, disponible en http://www.colombiadigital.net/entorno-tic/especial-del-mes/dispositivos-moviles/item/1341-sistemas-operativos-m%C3%B3viles.html (Consulta: 06/09/2012)

[17] iOS Delveloper Library, *The iOS Architecture,* Octubre de 2011, disponible en http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/IPhoneOSOverview/IPhoneOSOverview.html (Consulta: 07/09/2012)

[18] W-Bot. *Symbian OS, Definición e Historia*, Abril de 2010, disponible en http://comunidad.wilkinsonpc.com.co/symbian-os-265/symbian-os-definicion-e-historia-15296.html (consulta 07/09/2012)

[19] Johnston C. J., Evers, R., *Professional BlackBerry,* editorial Wrox, Julio de 2005

[20] Plain Concepts, Geeks ms, *Programar en Silverlight para Windows Phone*, 2011, disponible en http://geeks.ms/blogs/jyeray/archive/2011/04/18/wp7-libro-por-cap-237-tulos-programar-en-silverlight-para-windows-phone-7-segundo-cap-237-tulo.aspx (Consulta: 09/09/2012)

[21] Open Handset Alliance. http://www.openhandsetalliance.com/oha_faq.html (Consulta: 25/09/2012)

[22] Meier, R., *Professional Android application development*, editorial Wrox, 2009.

[23] Android developers, disponible en http://developer.android.com (Consulta: 25/09/2012)

[24] Gargenta, M., *Learning Android*, editorial O'reilly, 2011.

12. Glosario

API: Acrónimo de *Application Programming Interface*. Consiste en un conjunto de llamadas que ofrecen acceso a funciones y procedimientos representando una capa de abstracción para el desarrollador.

Bytecode: Código intermedio, más abstracto que el código máquina, y que necesita de un mediador o máquina virtual para poder ser transformado en código nativo y ejecutado por el hardware de destino.

E-commerce: Conjunto de transacciones comerciales que se realizan por medio de una red como Internet. Los pagos para esas transacciones pueden realizarse con tarjetas de créditos, cheques online, servicios de pagos, etc.

Framework: Termino con el que se define un amplio conjunto de elementos que permite el desarrollo y organización de software utilizando un determinado lenguaje o tecnología.

GSM: Acrónimo de Group Special Mobile os Sistema Global para las Comunicaciones Móviles. Es el estándar más extendido para la comunicación mediante telefonía móvil, permite realizar llamadas, envío de SMS y acceso a internet.

GPRS: Siglas de *General Packet Radio Service* o Servicio General de Paquetes vía Radio. Es una extensión de GSM que mejora las prestaciones originales de dicha tecnología aumentando la velocidad en la transmisión de datos.

Hilo: En sistemas operativos, un hilo o *thread* constituye cada uno de los flujos de ejecución en el que puede ser divido un proceso. Todos los hilos de un proceso comparten espacio en memoria o variables globales. Permiten la ejecución concurrente de varias tareas.

Interfaz: Se refiere a la abstracción que un determinado elemento o conjunto de elementos realiza sobre sí mismo, facilitando el uso y acceso por otros elementos externos.

JAR: Acrónimo de *Java Archive* o Archivo Java. Representa un conjunto de ficheros Java y se usa para la distribución de clases.

JVM: *Java Virtual Machine* o Máquina Virtual de Java. Es el elemento que se encarga de transformar el *bytecode* en código nativo de la plataforma de destino. Para cada plataforma, existe una JVM distinta.

Kernel: Núcleo de un sistema operativo y por lo tanto, pieza clave para el funcionamiento del mismo. Responsable de dar acceso al hardware, gestionar recursos y hacer llamadas al sistema.

KVM: Siglas de *Kilobyte Virtual Machine*. Se refiere a la máquina virtual disponible en Java ME cuyo está enfocado a dispositivos con capacidades de cómputo y memoria restringidas.

Librería: Agrupación de código que proporcionan servicios a programas independientes pasando a formar parte de éstos. Permiten la distribución de funcionalidades y la construcción modular.

Máquina virtual: Software que emula el comportamiento de una determinada arquitectura o en el ámbito de Java, aquella que permite convertir el *bytecode* en código nativo del hardware donde se encuentra instalada.

OHA: Siglas de la organización Open Handset Alliance en la que convergen multitud de empresas relacionadas con los dispositivos móviles y sus aplicaciones.

PayPal: Empresa *e-commerce* que permite pagos y transferencias de dinero por internet.

Plug-in: Pieza de software que se relaciona y ejecuta con otro para aportarle una función nueva y especifica.

Retrogaming: Es la afición de jugar y coleccionar viejos PC, consolas y videojuegos arcade.

Sprite: Los *sprites* son mapas de bits en 2D que se dibujan directamente en un destino de representación sin usar la canalización de transformaciones, iluminación o efectos.

XML: Siglas de *eXtensible Markup Language* o Lenguaje de Marcas Extensible. Es un metalenguaje que permite definir la gramática de otros lenguajes específicos.