INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

“Video juego para celular basado en física elemental”

13-1-0021

Presenta

Israel Murillo Hernández

Directores

Dr. Benjamín López Carrera

M. en C. Rubén Galicia Mejía

México D.F. a 14 de Diciembre de 2012

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
SUBDIRECCION ACADEMICA

ISCCR0 47-13-10021 /2015 13 de Mayo de 2015

Documento Técnico

“Video juego para celular basado en física elemental”

Presenta

Israel Murillo Hernández

Directores

Dr. Benjamín López Carrera M. en C. Rubén Galicia Mejía

RESUMEN

En este reporte se desarrollar una aplicación genérica multiplataforma optimizada para Android, la cual será un juego basado en juegos populares pero a través de un análisis e implementación de la física de modo que el movimiento sea más parecido a la realidad.

Palabras clave: Android, Videojuego, Física, Dispositivos móviles.

---

imurilloh@gmail.com

Advertencia

"Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan."

La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000, extensión 52000.

Agradecimientos

Agradezco especialmente a mis padres ya que sin su apoyo incondicional no hubiera sido posible la conclusión de este trabajo, gracias por sus conocimientos y palabras de apoyo, siempre incitándome a seguir adelante y proponerme nuevos retos.

A mis profesores y amigos con los que supe que siempre podía contar y que a través de esta etapa me brindaron los conocimientos y apoyo para poder seguir adelante. Muchas gracias.

Israel Murillo Hernández

Índice

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

A

R

B

Índice de Figuras

RESUMEN

La situación de la telefonía móvil ha experimentado una gran evolución en los últimos años tanto a nivel funcional como comercial. Uno de los puntos fuertes de esta evolución ha sido la aparición de los smartphones con Android. Android ha acaparado el 43% de las ventas de smartphones, situándose líder en su sector con unos 36 millones de unidades vendidas, frente al 17% que obtuvo en el mismo periodo de tiempo del año 2010. Además se prevé que en 2015 las ventas Android ronden el 50% de la cuota de mercado.

Se cumplió el objetivo de desarrollar una aplicación genérica multiplataforma optimizada para Android, la cual será un juego basado en el popular "angry birds" pero a través de un análisis e implementación de la física de modo que el movimiento sea más parecido a la realidad.

Para este fin se ha creado una aplicación a través de SDK de Corona. En este proyecto también se realiza un estudio para conoce los entornos de desarrollo de los diferentes sistemas operativos móviles y varias opciones de desarrollo multiplataforma que existen en la actualidad.

Al final se ha elegido dadas sus características un software multiplataforma llamado Corona SDK, permite la programación de dispositivos con Android, iOS y recientemente para Amazon/KIndle. Tiene un alto rendimiento y permite la utilización de sus componentes nativos.

Cabe recalcar que varias aplicaciones desarrolladas con Corona SDK han sido líderes en descargas en la "AppStore" y "Android Market" por lo que se demuestra su gran potencial y se puede considerar como una opción alternativa a otros entornos de programación.

1 Introducción

En este capítulo introductorio se muestra un enfoque general de los propósitos y objetivos de este proyecto. En él se tratan apartados tales como la motivación, los objetivos, las fases, los medios utilizados y los contenidos.

En el primer apartado se define la Motivación 1.1 para la realización de este proyecto, el por qué se ha realizado y que aspectos han influido en su elección y no la de otro proyecto.

El segundo apartado muestra los Objetivos 1.2 que se pretenden cumplir en este proyecto, lo que se quiere alcanzar y conseguir con ellos.

Para cumplir con los objetivos del proyecto hay que definir unas Fases 1.3 que nos indicarán su ciclo de vida.

También se han querido detallar los Medios 1.4 utilizados para la elaboración del proyecto tanto a nivel software como a nivel hardware.

Por último, en los Contenidos 1.5 se muestra una visión global del proyecto así como su estructura y un breve resumen de cada apartado.

1.1. Motivación

¿Por qué se ha elegido utilizar Android? ¿Por qué se ha decidido hacer un juego y no cualquier otra aplicación? En este apartado se explican las razones por las cuales se ha elegido este proyecto.

Si analizamos las estadísticas de los smartphone en 2010, solo en Estados Unidos, se vendieron 297 millones de smartphones, lo que supone un 19% del total de teléfonos móviles vendidos (1.6 billones de ventas totales). Estas ventas suponen un aumento del
72% respecto a 2009 y las expectativas esperan un mayor aumento [1].

Según la prestigiosa empresa Gartner (líder mundial en investigaciones de tecnologías de la información), actualmente los terminales con Android suponen el 38.5% del total de smartphones y prevé que en apenas 4 años, Android acapare el 49% de cuota del mercado en el sector móvil [2] como muestra la figura 1.

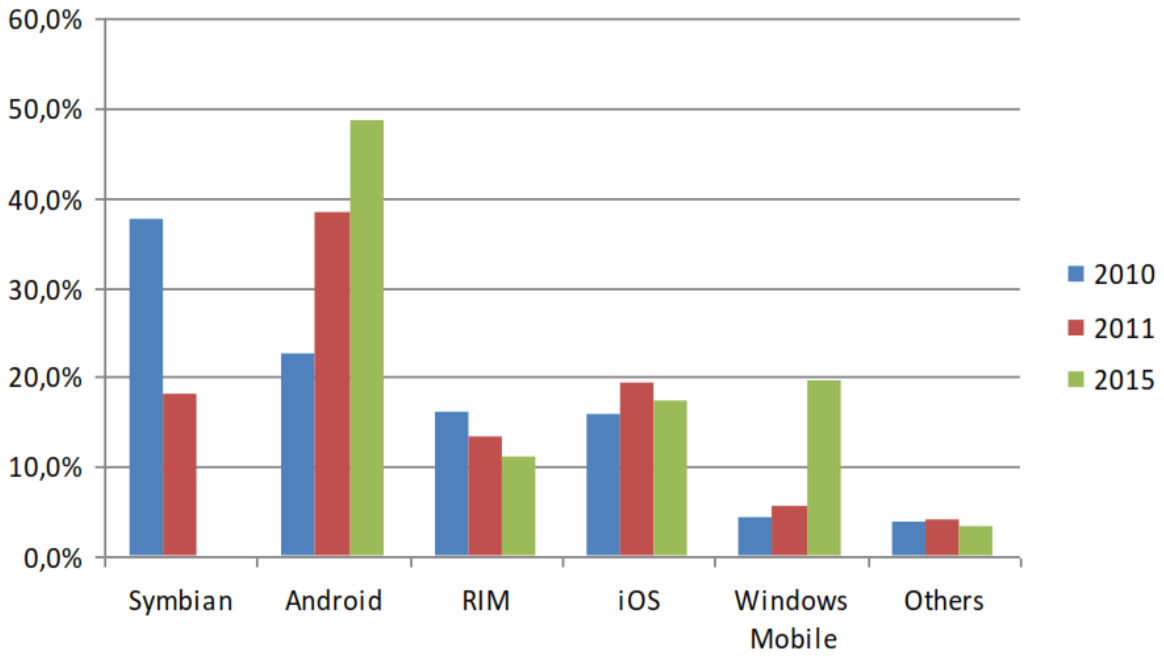

figura 1 Estadísticas anuales y previsión

• Centrándose en las aplicaciones en el Android Market, según AndroidZoom, en Septiembre de 2010 había disponibles 169.092 aplicaciones, de las cuales aproximadamente 26.400 estaban destinadas al entretenimiento [3]. Casi un año después, en Agosto de 2011, hay disponibles 277.252 aplicaciones con las que se han conseguido 6 billones de descargas [4].

En la tabla 1.1 se muestran algunos de los juegos más populares, su precio (en Dólares) y su número de descargas.

Juego	Nº de descargas	Precio
Angry Birds (Rovio Mobile)	10.000.000 - 50.000.000	Gratis
Live Holden Poker Pro (DragonPlay)	5.000.000 - 10.000.000	Gratis
Texas Poker (KamaGames)	1.000.000 - 5.000.000	Gratis
Doodle Jump (GameHouse)	500.000 - 1.000.000	0.9
Worms (Electronic Arts.)	100.000 - 500.000	2
Need For Speed (Electronic Arts)	50.000 - 100.000	3.1
Asphalt 6 HD (Gameloft)	10.000 - 50.000	7.17
Asteroids Defense 2 HD (Deonn Games)	500 - 1.000	2.85

A tenor de los datos, se ha considerado interesante la realización de un videojuego para smartphone sobre la plataforma Android, debido a la popularidad demostrada por este tipo de aplicaciones dentro del mercado Android.

El videojuego escogido es una versión del clásico "Angry Birds" (figura 2). El juego desarrollado consta de dos niveles diferentes y sus objetivos son destruir a los enemigos a través de lanzar pollos con una trayectoria de tiro parabólico hacia los enemigos

Dado que el juego adquirió fama mundial se ha querido diseñar conservando la esencia que le otorgó tanto éxito. La mecánica es tan simple como destruir a los enemigos, pero con las consideraciones técnicas que implica.

figura 2 Angry Birds

1.2. Objetivos

En este apartado se definen los objetivos que se querido tener en cuenta en la elaboración de este proyecto.

El objetivo principal es desarrollar un juego para la plataforma Android, concretamente desarrollar una adaptación móvil de "AngryBirds", ofreciendo una aportación incorporando y desarrollando un análisis físico del movimiento parabólico.

Otro objetivo que se ha tenido en cuenta ha sido elaborar un proyecto que pueda significar un punto de partida para futuros alumnos, de manera que puedan empezar a crear sus aplicaciones o juegos tomando como base este proyecto.

Para finalizar, el último objetivo ha sido crear este proyecto pensando en la posibilidad de futuras expansiones y funcionalidades adicionales de manera que se pueda continuar trabajando con él.

1.3. Fases del desarrollo

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

- **Documentación previa**: Al elegir este proyecto se tenían pocos conocimientos sobre Android, de manera que ha sido necesario documentarse sobre esta plataforma y como desarrollar aplicaciones para ella.

- **Análisis de requisitos**: Ha sido necesario evaluar inicialmente los requisitos necesarios para la elaboración de este proyecto y tener en cuenta las funcionalidades que se querían desarrollar.

- **Diseño**: El diseño del proyecto se ha realizado valorando que se quería lograr en cada una de las pantallas y como se quería mostrar al usuario, de manera que sirviera de guía para la implementación posterior.

- **Implementación**: Tras el análisis y diseño del proyecto, se prosigue con la fase de implementación. Al comienzo de dicha fase se consigue tener un juego ejecutable con la funcionalidad más básica, lo que supone que a medida que se profundice en el desarrollo del juego se irá ampliando la funcionalidad del mismo.

1.5. Contenidos

La presente trabajo se ha diseñado en varios capítulos en los cuales se profundiza más detalladamente sobre los diferentes aspectos que componen este proyecto.

En el primer capítulo, *introducción*, se muestra la motivación para la elección de este proyecto, los objetivos que se ha buscado perseguir, las diferentes fases del proyecto y los medios que se han requerido para la elaboración del mismo.

El segundo capítulo, *estado del arte*, realiza un paso por la historia de los smartphones y se comentan y comparan los sistemas operativos actuales para dichos dispositivos. Se incluye también un repaso de los juegos más populares para Android, se detalla en qué consiste Android y se muestran las diferentes versiones del juego Angry Birds.

El tercer capítulo, *análisis, diseño e implementación*, se trata del capítulo más importante de esta memoria y se puede considerar el núcleo central del proyecto. En él se analiza la estructura del juego, se realiza el diseño y se explica su posterior implementación.

El cuarto capítulo, *conclusiones*, se analiza el resultado final del proyecto. Y *líneas futuras*, se indican las posibles mejoras implementables al proyecto para añadirle funcionalidad o evolucionarlo.

2 Estado del Arte

En este capítulo se ha querido explicar con más detalle los aspectos relacionados con Android y los videojuegos. Relacionados con el primer caso se tratan aspectos como la historia de los smartphones, los diferentes sistemas operativos que pueden utilizar y la historia y evolución de Android. Respecto al uso de juegos, se tratan algunos de los juegos más populares para Android y la historia del juego que nos ocupa, Angry Birds.

2.1. Historia de los smartphones

Un smartphone es un término destinado a dispositivos móviles que aúnan funcionalidades de teléfono móvil (realizar o recibir llamadas y mensajes principalmente) con funcionalidades de una PDA (acceder al correo electrónico, organizador personal y actualmente incluso la instalación de diferente software) controladas por un sistema operativo que además gestiona el resto de recursos software y hardware.

La aparición del concepto smartphone, aunque se pueda considerar un término reciente, data de 1993. Debido a su alto coste, los primeros modelos apenas llegaron al público en general y solamente los altos ejecutivos se lo podían permitir [5].

El primer modelo de smartphone fue creado por IBM y recibió el nombre de Simon (figura 3). Se trató del primer dispositivo móvil en incorporar servicios de voz y datos (incluía calendario, bloc de notas, correo electrónico, juegos e incluso fax). Utilizaba una pantalla táctil con un teclado QWERTY, tal cual se usa en muchos móviles de la actualidad, pero su peso, tamaño y sobretodo su precio (900 dólares) frenaron su éxito. Su SO se llamaba Zauruz.

figura 3 Simon el primer smartphone

En 1997 aparece un prototipo de Ericsson, el modelo GS 88, que nunca se llegó a comercializar y fue el primero en ser calificado de smartphone a pesar de que no llegó al mercado. Un año después apareció el Nokia 9110 Communicator (figura 4), tratándose del primer producto de las series communicator de Nokia. Su pantalla aun no admitía colores y no se podía navegar por Internet. Como novedad poseía un teclado QWERTY deslizable que sirvió de inspiración a modelos más actuales. Su sistema operativo era GEOS.

figura 4 Nokia 9110 Communicator

En 1999, RIM (Research in Motion) sacó a la venta la primera BlackBerry. Se trataba del modelo 850 (figura 5) que poseía un teclado físico completo. Revolucionó la forma de comunicación a través de correo electrónico al hacerlo inalámbricamente. Permitía enviar mensajes, enviar y recibir correos electrónicos y actuar como un organi zador básico. En su contra hay que decir que tenía una pantalla muy pequeña que tan solo mostraba 8 líneas.

Tres años después RIM volvió a revolucionar el mercado sacando a la venta el modelo 5810 con una pantalla más grande y similar a una PDA. Su principal novedad era la posibilidad de navegar por internet. Ambos modelos montaban el sistema operativo propietario BlackBerry OS y tenían un defecto, necesitaban auriculares para hablar por teléfono ya que no tenían altavoces. Esto se solucionó en 2004 con el modelo 6210.

figura 5 BlackBerry 850. Primer modelo BlackBerry

En 2003 apareció el modelo Palm Treo 600 (figura 6), que poseía teclado QWERTY retro-iluminado, cámara integrada y pantalla a color. Otra importante novedad era el soporte de redes GSM y CDMA. Contaba con 32 Mb de RAM y un procesador de 144 MHz Este modelo dominó el mercado estadounidense de smartphones durante varios años y también montaba un sistema operativo propietario, PalmOS.

figura 6 Palm Treo 600

En 2007 se producen dos acontecimientos importantes en el mercado smartphone. Por un lado Apple lanzo su primer iPhone (figura 7), también con sistema operativo propietario, iOS, el cual tuvo un gran éxito gracias a su novedosa pantalla táctil y a una gran experiencia de navegación por Internet. Poseía una cámara de 2 megapíxeles e incluía conectividad WiFi. Supuso un punto de inflexión en la concepción de los teléfonos smartphones

figura 7 iPhone. Primer modelo.

Por el otro Google liberó la primera versión de Android para desarrolladores. Inicialmente no causó un gran revuelo pero hoy en día se trata del SO para móviles con mejor previsión de futuro. Este dato cobra más relevancia si cabe si se tiene en cuenta la cantidad de SO de calidad que existen actualmente (Symbian, Windows Phone, iOS,..).

En 2008 Apple lanzo la evolución del iPhone, el modelo 3G, el cual incluía la posibilidad de acceder a redes 3G y tan solo un año después lanzó el modelo 3GS.

En este mismo año, 2009, Palm anuncia WebOS (la evolución de PalmOS) y apareció en escena el Motorola Droid (figura 8), el cual supuso el primer gran éxito del SO Android en EEUU, vendiéndose 1 millón de unidades en apenas 74 días. Su gran novedad es el uso de la versión 2.0 del sistema operativo.

figura 8 Motorola Droid. Primer gran éxito de Android

En 2010 apareció el HTC EVO 4G (figura 9), un móvil que portaba Android y que logra aprovechar al máximo el potencial de la red WiMax (la red inalámbrica más rápida en EEUU). Además es destacable su amplia pantalla de 4.3" con una resolución 800x400.

figura 9 HTC EVO 4G

Por último, en la actualidad (2012) se ha convertido en uno de los años con mayores novedades. Las principales novedades son el aprovechamiento de las redes 4G, las pantallas en alta resolución y el gran aumento de la potencia de los procesadores, que comienzan a permitir la visualización de videos y juegos en HD. Algunos de estos nuevos terminales son los módelos iPhone 4, Google Nexus S o Samsung Galaxy S3.

2.2. Sistemas operativos para smartphones

Un SO móvil es un sistema operativo que controla un dispositivo móvil análogamente a como Windows, Linux o MAC controlan un PC, es decir, controlan tanto los recursos hardware como software, con la diferencia que estos SO móviles son más simples y están más orientados a la conectividad inalámbrica y los formatos multimedia.

En la actualidad el mercado smartphone se lo reparten 6 compañías con sus respectivos sistemas operativos. Estos sistemas operativos son Android, iOS, BlackBerry OS, Microsoft Windows Phone, Symbian, y Bada.

- **Android**: Se trata de un SO basado en Linux que se ejecuta en Java, desarrollado por Google. En 2007 se lanzó el primer kit para desarrolladores y desde entonces se ha ido actualizando el sistema estando disponible incluso para dispositivos como tablets o netbooks.
- **iOS**: Sistema operativo desarrollado por Apple. Inicialmente se diseñó para el dispositivo iPhone pero su uso se ha extendido a otros dispositivos propiedad de Apple como iPod e iPad.
- **BlackBerry OS**: Desarrollado por Research In Motion (RIM) para sus dispositivos BlackBerry, está basado en Java y emplea MIDP sobre CLDC. Soporta multitarea y diferentes métodos de entrada como trackwheel, trackball, touchpad y pantallas táctiles.
- **Microsoft Windows Phone**: Sistema operativo propietario de Microsoft. Soporta pantallas táctiles de alta resolución, integración con redes sociales y multiescritorio. Por el contrario se olvida la sincronización de datos con el PC, transfe- rencia por Bluetooth e intercambio de tarjetas de memoria externas.
- **Symbian OS**: Desarrollado por Symbian Ltd. que fue fundada por la unión de varias empresas de telefonía móvil como Nokia, Motorola o Ericsson entre otras. Permite un uso eficiente de memoria y energía del dispositivo y soporta en tiempo real los protocolos de comunicación y telefonía. Además soporta múltiples lenguajes de programación como Symbian C++, Java ME, Open C, etc. Su futuro es incierto ya que actualmente es propiedad de Nokia, quien en febrero de 2011 firmó una alianza con Microsoft para usar Windows Phone.

Bada: Sistema operativo desarrollado por Samsung que permite cualquiera de los kernel de Linux. Sus aplicaciones son desarrolladas en C++ y ofrece varios controles de interfaz de usuario a los desarrolladores. También soporta diversos sensores, GPS y detección de rostros. No permite instalar aplicaciones fuera de la tienda ni usar programas tipo VoIP/SIP [6].

Otros: Otros SO que a día de hoy no forman parte de las estadísticas de sistemas operativos más relevantes son WebOS, Maemo/MeeGo o LiMo, aunque estos dos últimos sistemas recientemente se han unido dando lugar a un sistema operativo llamado Tizen [7].

Según el análisis de Gartner del segundo cuatrimestre del año 2011 [8], la venta de dispositivos móviles se ha incrementado un 16.5% mientras que la venta de smartphones ha aumentado un 74% respecto al mismo periodo de 2010, constituyendo el 25% de las ventas totales.

Dicho análisis (tabla 2.1) también muestra el aumento de ventas de teléfonos con Android, iOS o Bada, y un decremento de ventas de Symbian, Windows Phone y RIM. El principal motivo de caída de Symbian se encuentra en la disminución de ventas de Nokia, principal portador de dicho sistema operativo.

Operating System	2011 Units	2011 Market Share (%)	2010 Units	2010 Market Share (%)
Android	46,775.9	43.4	10,652.7	17.2
Symbian	23,853.2	22.1	23,386.8	40.9
iOS	19,622.8	18.2	8,743.0	14.1
BlackBerry OS	12,652.3	11.7	11,628.8	18.7
Bada	2,055.8	1.9	577.0	0.9
Microsoft Windows Phone	1,723.8	1.6	3,058.8	4.9
Others	1,050.6	1.0	2,010.9	3.2
Total	**107,704.4**	**100.0**	**62,058.1**	**100.0**

Tabla 2.1. Sistemas operativos en smartphone. Ventas en el segundo cuatrimestre de 2010 y 2011.

Analizando las ventas a usuarios finales [9], Gartner nos muestra un incremento casi exponencial del sistema operativo Android alcanzando unas ventas de 46.8 millones de unidades, seguido de Symbian con 23.9 millones y de iOS con 19.6 millones (figura 2.8).

Analizando las ventas a usuarios finales [9], Gartner nos muestra un incremento casi exponencial del sistema operativo Android alcanzando unas ventas de 46.8 millones de unidades, seguido de Symbian con 23.9 millones y de iOS con 19.6 millones (figura 10).

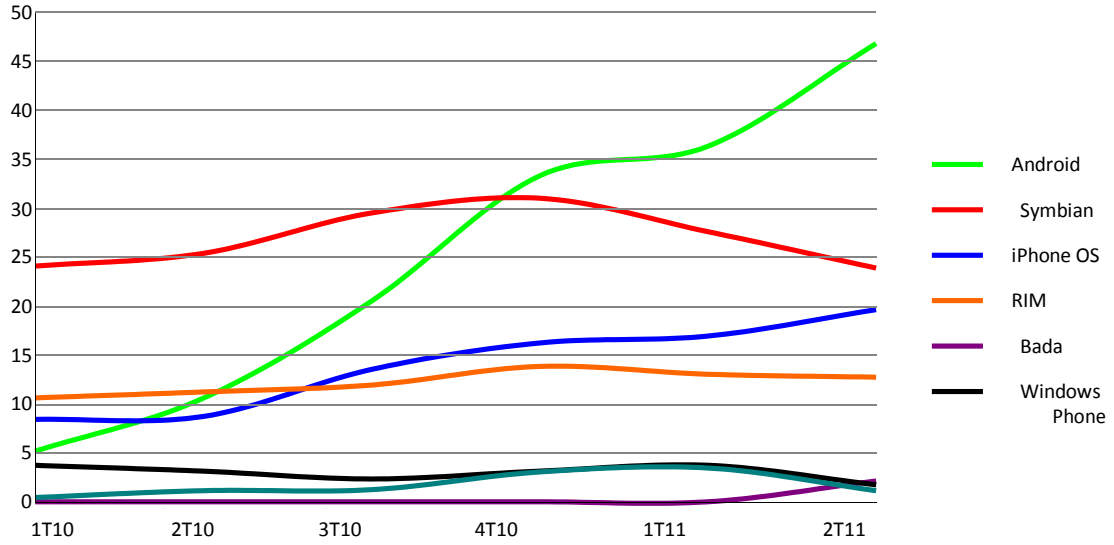

figura 10. Sistemas operativos para smartphone. Evolución trimestral 2010-11.

Técnicamente no existe el sistema operativo ideal, pero comparándolos entre sí se pueden ver las ventajas de unos y otros y se puede evaluar para cada caso que sistema operativo puede resultar más adecuado [10].

Dos de los parámetros más importantes en la elección del sistema operativo son el interfaz de usuario y el conjunto de aplicaciones que tenga disponibles.

Si comparamos el interfaz de usuario de los distintos sistemas operativos que se han analizado tanto iOS como Android y Windows Phone 7 ofrecen un apartado visual bonito, fluido y con buena respuesta, aunque de los tres sistemas Android es el único que ofrece una personalización total por ser un sistema operativo gratuito.

Si evaluamos la cantidad de aplicaciones, tanto por calidad como por cantidad que hay detrás de cada sistema operativo, el mejor sistema seria iOS seguido cada vez más de cerca por Android y un poco más lejos por Symbian.

Otro factor a tener en cuenta es el rendimiento del dispositivo. En este caso Symbian es claramente el sistema que más eficientemente gestiona la energía y batería del dispositivo, aunque Windows Phone parece ser que también funciona bien, mientras que éste es el punto débil de Android e iOS.

Por último se debe analizar la funcionalidad de los sistemas. Si bien inicialmente tanto iPhone como Windows Mobile arrancaron bastante escasos de funcionalidad, con sus nuevas actualizaciones iOS 4.3 y Windows Phone 7 parecen estar bastante completos aunque éste último no soporta aun multitarea ni Flash (iOS tampoco soporta Flash). En éste aspecto Android es el más completo ya que soportan multitarea, multitáctil, copiar y pegar, Flash, widgets, WiFi Tethering, etc.

Actualmente y a tenor de los datos analizados, se ha considerado que Android es el sistema operativo que ofrece una mejor funcionalidad y experiencia de usuario además de ofrecer un buen soporte a los desarrolladores ya que es software libre, lo cual se ha tenido en cuenta a la hora de desarrollar el juego.

2.3. Aplicaciones y juegos para smartphones

Hoy en día las aplicaciones son un aspecto básico de un smartphone y posiblemente un sistema operativo que no posea un buen catálogo de aplicaciones se quede fuera de la lucha por el mercado móvil. La naturaleza de las aplicaciones es muy variada, desde lectores de periódicos hasta juegos pasando por aplicaciones GPS o redes sociales por poner solo un ejemplo.

En los diferentes stores de cada plataforma, el número de aplicaciones disponible aumenta cada día que pasa de manera que se convierten en un punto muy importante de cada sistema operativo. Este crecimiento demuestra que los smartphone son los dispositivos llamados a suceder al teléfono convencional.

2.3.1. Stores de aplicaciones

Actualmente cada uno de los sistemas operativos destinados a ser utilizados en un smartphone posee su propio store de aplicaciones a los cuales se puede acceder a través del propio móvil y en algunos casos incluso desde el PC.

- Android Market: Store del sistema operativo Android. Su lanzamiento se produjo en octubre de 2008 y tres años después el número de aplicaciones activa es 319.161.

- App Store: Store de Apple. Su lanzamiento se produjo en julio de 2008 y en octubre de 2011 el número de aplicaciones activas es algo mayor al de Android Market: 459.589.

- Marketplace: Store de Windows Phone y su lanzamiento se produjo en octubre de 2009. Dos años más tarde, en octubre de 2011, el número de aplicaciones es 35.000.

- App World: Se trata del store de BlackBerry OS. Su salida al mercado se produjo en abril de 2009 y en julio de 2011 disponía de más de 40.000 aplicaciones.

- Nokia Store: Store de Symbian OS que inicialmente se llamó Nokia Ovi Store. Nació en mayo de 2009 y en abril de 2011 ya tenía 50.000 aplicaciones

- Palm App Catalog: Store de WebOS que se lanzó en junio de 2009 con apenas 18 aplicaciones y en septiembre de 2010 ya estaban disponibles 4000 aplicaciones oficiales.

2.3.2. Tipos de aplicaciones

Actualmente hay muchos tipos de aplicaciones (negocios, bolsa, meteorológicas, deportivas, etc.) pero centrándose en el market de Android se pueden clasificar en dos grandes grupos: aplicaciones y juegos.

- Juegos: Hay gran variedad de tipos: arcade, carreras, deportes, cartas, puzzle e incluso widgets están incluidos en esta categoría.

- Aplicaciones: Cualquier aplicación que no sea juego ni widget, es decir, aplicaciones

relacionadas con compras, redes sociales, finanzas, medicina, etc.

Por lo general, tanto para el market de Android como para cualquier otra tienda de aplicaciones de otro sistema operativo, las aplicaciones que más éxito suelen tener son aquellas gratuitas.

2.3.3 Juegos más populares para Android

En este apartado se comentan los juegos más populares que posee Android. Muchos de ellos sirven como inspiración para futuros juegos e incluso han marcado tendencia. No todos son exclusivos de Android, algunos como Angry Birds son originales de iOS y han sido adaptados a este sistema operativo.

Evaluando diferentes listas de internet sobre los juegos gratuitos más populares se ha elaborado otra con aquellos que han tenido más éxito:

1. Angry Birds: El más popular en la era smartphone. En este juego se lanza con un tirachinas unos pájaros muy enfadados que quieren acabar con unos cerdos que les roban sus huevos. Tiene más de 10 millones de descargas (figura 11).

2. PaperToss: Tan simple como encestar una bola de papel en una papelera pero adictivo como pocos. Ha sido descargado más de 10 millones de veces.

3. Jewels: Clásico juego en el que hay que juntar 3 figuras iguales para sumar puntos. Este simple pero muy divertido juego ha sido descargado más de 10 millones de veces.

4. Air Control Lite: En este juego deberemos controlar el tráfico aéreo y evitar colisiones entre aviones. También tiene más de 10 millones de descargas.

5. Drag Racing: Los buenos juegos de carreras de coches han sido siempre bien valorados y este juego ya ha sido descargado más de 10 millones de veces.

6. Robo Defense: Clásico juego del estilo “defiende la torre” en el que aparecerán enemigos y tendremos que impedir que entren en la fortaleza. Descargado más de 5 millones de veces (figura 12).

7. World War: Juego que solo posee interfaz gráfica y ofrece interacción con el usuario a través de botones, en el que debemos cumplir misiones y atacar a otros jugadores para ganar dinero y aumentar nuestro ejército. Descargado más de 5 millones de veces.

8. Live Holden Poker Pro: Este juego multijugador en el que hay que acumular dinero jugando al póker ha sido descargado más de 5 millones de veces.

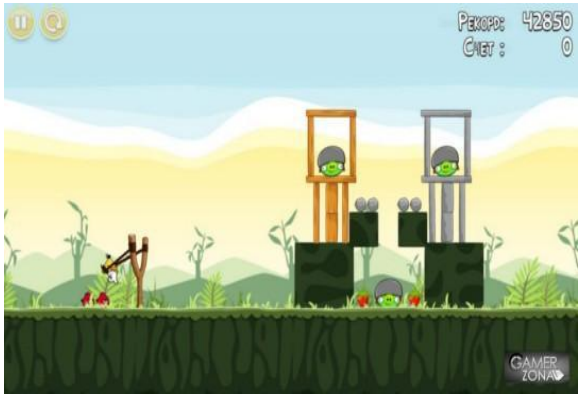

figura 11 Angry Birds version para Android

figura 12 Robo Defense. Juego para Android.

2.4. Historia de Android

En 2003 se creó la empresa Android Inc. en cuyos inicios se gestó la idea de desarrollar un sistema operativo para móviles, al que llamaron Android. Esta compañía fue comprada por Google en 2005 [11].

Dos años después se anunció Android como un sistema abierto basado en GNU/Linux y orientado al mercado de los llamados teléfonos inteligentes. Fue el ingreso de Google en el mercado de la telefonía móvil.

En 2007 se anuncio la creación de la Open Handset Alliance [12], una alianza de varias compañías dedicadas al sector de los dispositivos móviles. En dicha presentación se anunció que Android sería su apuesta como sistema operativo y su intención era que fuera abierto y gratuito gracias, entre otras cosas, a su kernel basado en un kernel Linux, lo que supondría la posibilidad de adaptarlo a casi cualquier terminal móvil.

A los pocos días de la presentación del sistema operativo se anuncio la disponibilidad del SDK de Android junto a un emulador, orientado a los futuros desarrolladores de aplicaciones Android [13].

El primer teléfono móvil que cargaba Android fue el HTC G1 (o HTC Dream) lanzado en octubre de 2009 con la versión 1.1 y desde ese momento multitud de fabricantes (LG, HTC, Sony Ericcson, etc.) han incorporado Android a sus dispositivos móviles en sus diferentes versiones.

Progresivamente y a medida que Android ha ido adaptándose a diferentes operadoras y compañías, se ha podido comprobar un concepto conocido como fragmentación, que supone la aparición de diferentes versiones y la actualización aleatoria por parte de las diferentes compañías y operadoras de sus terminales móviles.

El primer gran terminal que salió a la venta portando Android fue el primer móvil de Google y fabricado por HTC, llamado Nexus One, que además llevaba incorporada la nueva actualización del sistema operativo, la versión 2.1.

En mayo de 2010, Android dio el gran salto con la actualización 2.2 que trajo bastantes mejoras: Adobe Flash Player, optimización de WiFi, Bluetooth, mejora del rendimiento de batería, etc. y aparecieron grandes modelos de terminal como el HTC Des i- re o Samsung Galaxy S. A finales de año apareció la última versión disponible para smartphones, la 2.3, que aun no ha sido aprovechada por la mayoría de terminales y ofrece novedades como soporte nativo para VoIP, SIP o soporte NFC.

Por último, Android no es solo un sistema operativo para smartphones, también ha sido incorporado a las tablets PC y netbooks con una versión exclusiva, la 3.0.

2.4.1. ¿Qué es Android?

Se trata de un conjunto de software para dispositivos móviles formado por sistema operativo, middleware y aplicaciones [14].

Entre sus características se encuentran un marco de aplicaciones que permite reusar y reemplazar componentes, una maquina virtual Dalvik optimizada para dispositivos móviles, un navegador integrado basado en el motor WebKit, gráficos optimizados 2D y 3D basados en OpenGL o SQLite para almacenamiento de datos estructurados entre otras características.

El SDK de Android proporciona las herramientas y APIs necesarias para comenzar a desarrollar aplicaciones utilizando Java como lenguaje de programación.

2.4.2. Arquitectura de Android

La imagen a continuación (figura 13) muestra como está estructurado el sistema operativo y los componentes de cada sección.

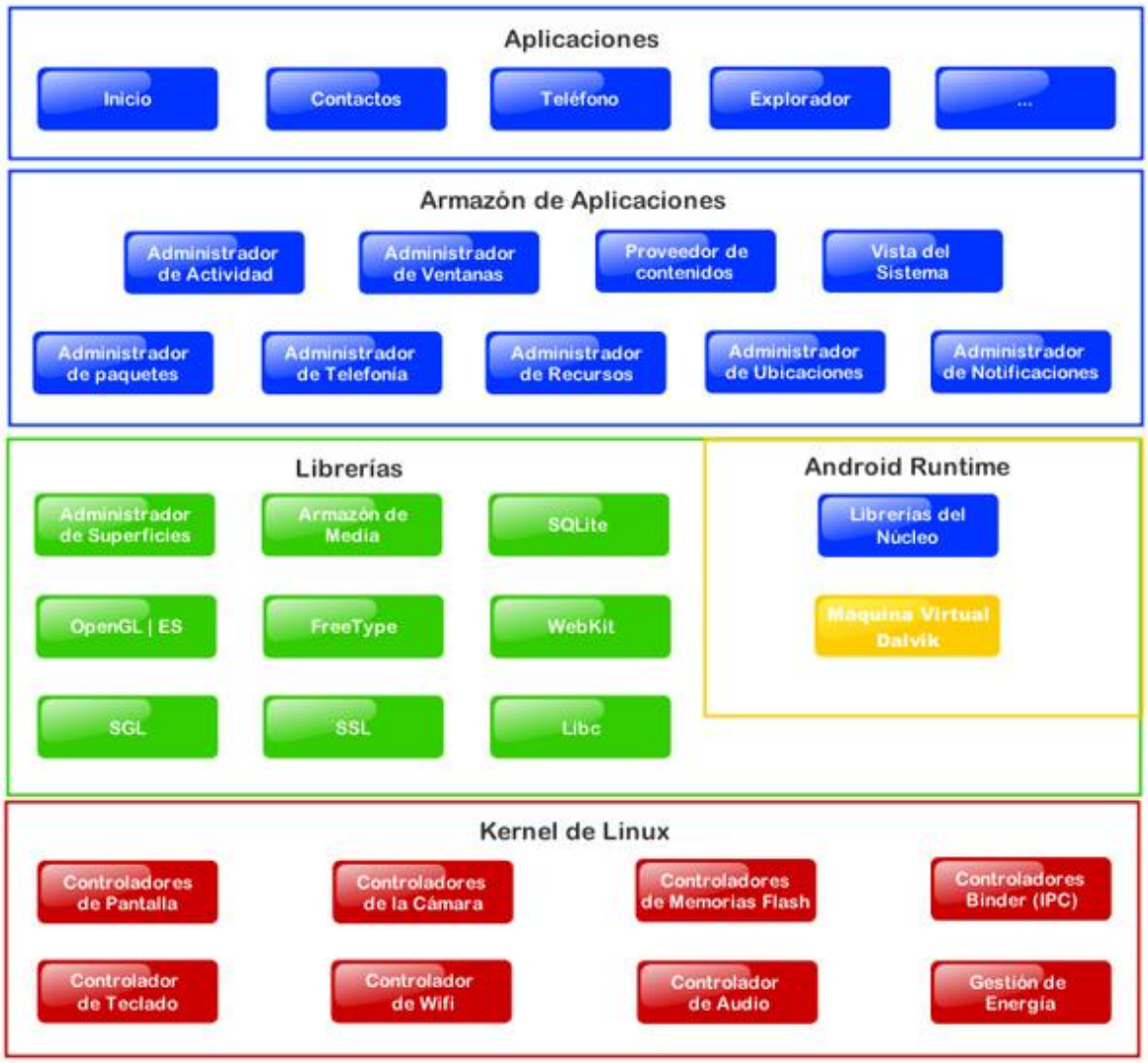

figura 13 Arquitectura del sistema operativo Android

A continuación se explican más detalladamente cada una de las secciones de la arquitectura.

Aplicaciones:

Se trata de un conjunto de aplicaciones básicas escritas en Java como e-mail, calendario, aplicación de SMS, mapas, navegador, etc. que suelen venir pre-instaladas en el dispositivo móvil.

Framework de aplicaciones:

Compuesto por un conjunto de herramientas para el desarrollo de aplicaciones. Al ofrecer una plataforma de desarrollo abierta, Android ofrece a los desarrolladores la capacidad de crear aplicaciones innovadoras y extremadamente ricas visualmente, además de poderse beneficiar del hardware del controlador, acceder a información de localización, correr servicios en background, etc.

Los desarrolladores podrán acceder a las mismas APIs usadas por las aplicaciones básicas. La arquitectura de la aplicación está diseñada para reutilizar componentes, es decir, cualquier aplicación puede publicar sus capacidades y cualquier otra puede hacer uso de las mismas.

Por debajo de todas las aplicaciones se encuentran un conjunto de servicios y sistemas, incluyendo:

- Un amplio conjunto de vistas (*Views*) que se pueden usar para construir una aplicación como listas, grids, cajas de texto, botones, etc. e incluso un navegador embebido.

- *Content Providers* que permiten a las aplicaciones acceder a datos de otras aplicaciones o compartir sus propios datos.

- *Resource Manager* para poder acceder a recursos como Strings, gráficos y archivos XML.

- *Notification Manager* que permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.

- *Activity Manager* para manejar el ciclo de vida de las aplicaciones y ofrecer una navegación común.

Librerías

Compuesto por un conjunto de librerías escritas en C/C++ usadas por varios compone n- tes del sistema, cuyas capacidades son puestas a disposición del desarrollador a través del framework de aplicaciones.

Algunas de las librerías principales son: System C, librerías Media, Surface Manager, LibWebCore, SGL, librerías 3D, FreeType y SQLite.

Android Runtime

Compuesto por las librerías principales y la máquina virtual Dalvik. Las librerías principales permiten que la funcionalidad disponible en las librerías anteriores est é disponible desde las librerías principales de Java. La maquina virtual Dalvik se ha diseñado de forma que cada aplicación pueda correr en su propio proceso con su propia instancia de la maquina virtual, de manera que un dispositivo móvil pueda correr múltiples maquinas virtuales simultáneamente que ejecutan ficheros en el formato .dex optimizado para minimizar el uso de memoria.

Linux Kernel

Android se basa en una versión de Linux 2.6 para servicios del sistema principales como seguridad, manejo de memoria, manejo de procesos, pila de red o modelo de controladores. Además actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

2.4.3. Fundamentos de las aplicaciones

Las aplicaciones Android están escritas en lenguaje de programación Java [15]. Las herramientas del SDK compilan tanto el código como los ficheros de datos y otros recursos en un único fichero con extensión .apk que incluye todo el código de la aplicación y es el fichero que utiliza el dispositivo para instalarlas.

Cuando la aplicación ha sido instalada en el dispositivo móvil, vive en su propia zona de seguridad. Esto se consigue de la siguiente forma:

- Android es un sistema Linux multiusuario y cada aplicación se considera un usuario diferente.

- Android asigna un ID de usuario único a cada aplicación (este identificador es conocido por el sistema y no por la aplicación). El sistema establece unos permisos para todos los ficheros de una aplicación de manera que solo el ID de usuario asignado a esa aplicación puede acceder a ellos.

- Cada proceso corre en su propia maquina virtual de manera que está aislada del resto de aplicaciones.

- Cada aplicación corre en su propio proceso Linux que Android arranca cuando cualquier componente de la aplicación necesita ser ejecutado y que Android cierra cuando ya no se necesita o es necesario recuperar memoria para otras aplicaciones.

Android utiliza el principio de privilegios mínimos, haciendo que una aplicación solo pueda acceder a los componentes que requiere para funcionar y ninguno más, de manera que no puede acceder a partes del sistema a las que no tiene permisos.

Sin embargo es posible que varias aplicaciones compartan datos si comparten el mismo ID de usuario, permitiéndolas incluso compartir la misma máquina virtual y el mismo proceso. También es posible que una aplicación acceda a los servicios del sistema, como almacenamiento en la tarjeta SD, contactos de teléfono, etc. Para ello el usuario debe conceder a la aplicación ciertos permisos a la hora de instalarla en el dispositivo.

Los componentes de las aplicaciones son los bloques esenciales para la construcción de una aplicación. Cada componente es un punto diferente a través del cual el sistema es capaz de acceder a tu aplicación, aunque no todos los componentes son puntos de entrada reales para el usuario e incluso pueden depender de otros componentes. A pesar de ello, cada componente tiene entidad propia y juega un rol específico en el comportamiento de la aplicación.

Existen cuatro tipos de componentes, cada uno con un propósito y ciclo de vida distinto:

- Actividades: Representa una pantalla única con interfaz de usuario. Una actividad es implementada como subclase de Activity y es independiente del resto de actividades de una aplicación. El ciclo de vida de una actividad se muestra en la figura 14.
- Servicios: Representa una tarea sin interfaz gráfico que se ejecuta de fondo. Otro componente puede arrancarlo, interactuar con él, dejarlo correr o pararlo. Un servicio es implementado como una subclase de Service.
- Proveedor de contenidos: Se encarga de manejar datos compartidos entre aplicaciones, almacenados en cualquier localización de almacenamiento persistente, que pueden ser consultados o modificados. Se implementa como una subclase de ContentProvider.
- Receptores broadcast: Son componentes que responden a los anuncios broadcast originados por el sistema u otras aplicaciones. No tiene interfaz gráfico pero puede enviar notificaciones a la barra de estado del sistema. Se implementan como una subclase de BroadcastReceiver.

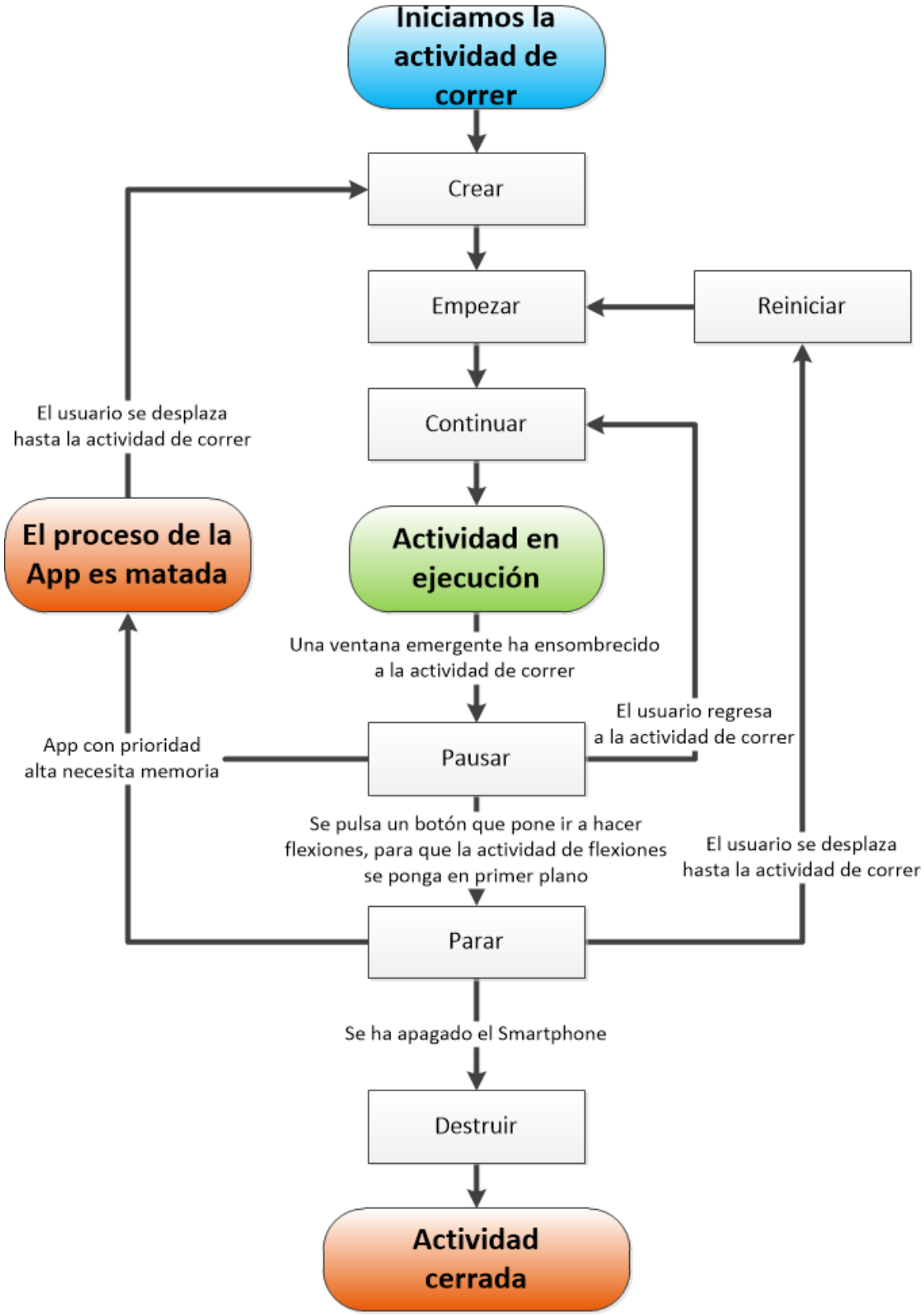

figura 14 Ciclo de vida de una actividad

Un aspecto único de Android es que una aplicación puede arrancar cualquier componente de otra aplicación sin necesidad de implementar el código, basta con acceder a la actividad de la aplicación que pueda realizar la tarea que necesitas.

Estos componentes, excepto ContentProvider, se activan mediante el uso de un mensaje asíncrono llamado Intent, cuya finalidad es enlazar dos componentes en tiempo real. En el caso de Activity o Service este mensaje representa una acción a realizar.

Antes de poder usar estos componentes es necesario declararlos en un archivo llamado AndroidManifest.xml, en el cual se deben incluir también los permisos que necesita la aplicación, el nivel de API mínimo que se requiere, requerimientos hardware y software necesarios, etc.

3 Análisis, diseño e implementación

En este capítulo se detalla el proceso de creación del Proyecto para el Trabajo Terminal, desde la primera idea hasta la instalación de la aplicación en un dispositivo móvil.

Se ha dividido en tres partes para explicar cada una de ellas en detalle por separado. En la primera se muestra un análisis inicial del proyecto y sus primeros pasos. La segunda parte se centra en el diseño realizado y las características del juego. En la tercera y última parte se detalla la implementación del juego.

3.1. Metodología

Previo a la planificación, es necesario decidir qué modelo de ciclo de vida se va a utili- zar en el desarrollo de la aplicación. De entre los diferentes modelos (lineal, cascada, sashimi, etc.) se ha elegido el modelo evolutivo.

Este modelo es ideal para el proyecto ya que ofrece gran flexibilidad para la inclusión de nuevos requerimientos, ya que es muy complicado obtenerlos todos al inicio, permitiendo añadir o mejorar la funcionalidad de los diferentes módulos del proyecto en cualquier momento. Como muestra la figura 15 este modelo se basa en la iteración del ciclo requerimientos-desarrollo-evaluación

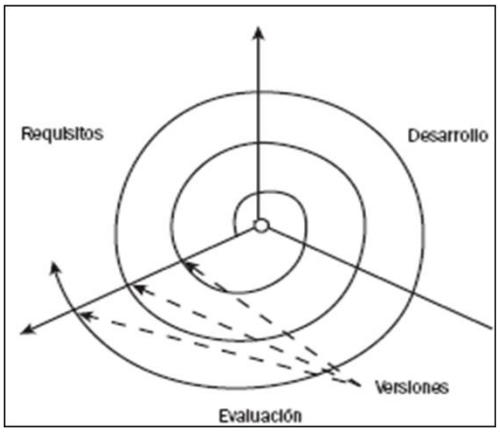

figura 15 Ciclo de vida evolutivo

Finalizada la fase de desarrollo se tiene una versión del producto que es evaluada, tras la cual se pueden añadir requerimientos nuevos o revisar los actuales si la evaluación no es satisfactoria, consiguiendo que el producto se pueda ir puliendo y mejorando continuamente.

En los diferentes módulos o componentes del proyecto también se ha aplicado este modelo de manera que ha sido posible depurarlos por separado hasta obtener la versión final.

Los componentes son:

- **Formación inicial:** Debe familiarizarse con los elementos necesarios para el desarrollo del proyecto y estudiar tanto el entorno de desarrollo como con la plataforma para la que se implementa la aplicación.
- **Memoria y documentación:** A medida que se avanza en el desarrollo del proyecto es necesario recoger toda la documentación necesaria (análisis, diseño, problemas encontrados, variaciones, etc.) para redactar el documento final.
- **Aspectos y estructura básica:** Este módulo aporta la funcionalidad básica para que la aplicación funcione. Incluye inicio y fin de la aplicación, detección e implementación de eventos (pantalla, botones) y el paso entre actividades.
- **Interfaz gráfico:** Se diseñan las apariencias de las diferentes pantallas y se crean los diferentes menús, diálogos e iconos.
- **Núcleo del juego:** Se trata del componente más extenso ya que engloba toda la gestión de movimientos y disparo, colisiones y el ciclo de vida de la aplicación.
- **Otros:** En este componente se incluyen los sonidos y la vibración.
- **Evaluación y pruebas:** Una vez obtenida la versión completa se debe evaluar y comprobar que todo funciona según lo esperado.

3.1.1. Propuesta inicial

La idea planteada inicialmente fue la realización de un juego sobre la plataforma Android que aplicara leyes físicas y que fuera factible su implementación en un desarrollo de un semestre, correspondiente al tiempo disponible para su desarrollo. La elección del juego Angry Birds fue gracias a que cuenta con las características físicas necesarias para la implementación de dichas reglas.

El juego básicamente, trata de destruir a los enemigos impactándolos con un proyectil el cual es lanzado al estilo de un tiraficas, creando así un movimiento parabólico.

El juego inicia cargando una pantalla de bienvenida, en la cual se le puede poner información acerca del juego, una vez termina el delay de la pantalla de bienvenida se carga el menú principal en el cual nos muestra el titulo del juego y el botón de Jugar.

Una vez seleccionada la opción de Jugar, se despliega la pantalla de opción de nivel del juego. La cual cuanta con dos niveles implementados los cuales difieren entre si por la dificultad de los objetivos, ya que el escenario está dotado de obstáculos los cuales dificultan impactar al enemigo directamente.

El modo de juego dispone de cuatro intentos (vidas) las cueles están representadas por un icono en la parte superior izquierda de la pantalla, las cuales nos permiten disponer de un proyectil. La manera de disparar un proyectil es situando el dedo en el personaje y deslizarlo en sentido opuesto a la trayectoria deseada, estilo tirafichas, el cual saldrá con una fuerza proporcional a la distancia con la que se deslizo el dedo y a un ángulo el cual también es seleccionado a la hora de deslizar el dedo en la pantalla touch.

Una vez el jugador logre destruir a los enemigos, se carga el nivel siguiente, o en su defecto siendo el caso de que el jugador no haya logrado destruir los enemigos con los cuatro intentos disponibles, mostrara un mensaje de que ha perdido y se reiniciara el nivel actual.

3.1.2. Requisitos de usuario

Tras el análisis de la propuesta inicial, se realiza la evaluación de los requisitos de usuario. En dichos requisitos, el propio usuario indica un conjunto de restricciones o funcionalidades que el programa, juego o aplicación en cuestión debe cumplir. Se diferencian dos tipos de requisitos: de capacidades (requeridos por el usuario para resolver un problema o determinar un objetivo) y de restricciones (impuestos por los usuarios sobre cómo debe ser resuelto el problema o logrado el objetivo).

Los requisitos de usuario están compuestos por los siguientes campos:

- **Identificador**: Muestra el tipo de requisito RUC (requisito de usuario de capacidad) o RUR (requisito de usuario de restricción) acompañado de un valor numérico. El identificador debe ser un valor único.
- **Necesidad**: Muestra la prioridad del requisito. Su valor puede ser Esencial o Deseable.
- **Título**: Frase breve que indica el nombre del requisito.
- **Descripción**: Breve comentario que describe el requisito.

A continuación se muestran los requisitos de usuario de capacidad:

Identificador:	RUC-01	Necesidad:	Esencial
Título:	Arranque del juego		
Descripción:	El juego se arrancará a través de un ejecutable instalado en el propio terminal.		

Tabla 3.1. RUC-01. Arranque del juego.

Identificador:	RUC-02	Necesidad:	Deseable
Titulo:	Pantalla predeterminada		
Descripción:	Los niveles del juego deben generarse de forma predeterminada.		

Tabla 3.2. RUC-02. Pantalla predeterminada.

Identificador:	RUC-03	Necesidad:	Deseable
Título:	Mensaje de pausa		
Descripción:	Se debe mostrar un mensaje cuando se pausa el juego, permitiendo al jugador seguir jugando o abandonar la partida.		

Tabla 3.3. RUC-03. Mensaje de pausa.

Identificador:	RUC-04	Necesidad:	Deseable
Título:	Mensaje para abandonar la aplicación		
Descripción:	En el menú principal se debe mostrar un mensaje al usuario que permita abandonar correctamente la aplicación.		

Tabla 3.4. RUC-04. Mensaje para abandonar la aplicación.

Identificador:	RUC-05	Necesidad:	Esencial
Título:	Personaje		
Descripción:	Se puede direccionar al personaje antes de ser lanzado		

Tabla 3.5. RUC-05. Personaje.

Identificador:	RUC-06	Necesidad:	Esencial
Título:	Indicador de puntuación		
Descripción:	Se debe mostrar la puntuación acumulada durante la partida.		

Tabla 3.6. RUC-6. Indicador de puntuación.

Identificador:	RUC-07	Necesidad:	Esencial
Título:	Indicador de vidas.		
Descripción:	Se debe mostrar el número de vidas restante durante la partida.		

Tabla 3.7. RUC-7. Indicador de vidas.

Identificador:	RUC-08	Necesidad:	Deseable
Título:	Indicador de nivel.		
Descripción:	Se debe poder mostrar el numero de nivel durante la partida en el modo Panic		

Tabla 3.8. RUC-8. Indicador de nivel.

Identificador:	RUC-09	Necesidad:	Deseable
Título:	Sonido de explosión		
Descripción:	Se debe escuchar un sonido cuando se rompe destruye a un objetivo.		

Tabla 3.9. RUC-9. Sonido de explosión.

Identificador:	RUC-10	Necesidad:	Deseable
Título:	Sonido de bloque.		
Descripción:	Se debe escuchar un sonido cuando el proyectil sale disparado.		

Tabla 3.10. RUC-10. Sonido de disparo.

Identificador:	RUC-11	Necesidad:	Deseable
Título:	Sonido fin de partida		
Descripción:	Se debe escuchar un sonido cuando el jugador finaliza un nivel, satisfactoriamente o de manera negativa.		

Tabla 3.18. RUC-11. Sonido fin de partida.

Seguidamente se muestran los requisitos de usuario de restricción:

Identificador:	RUR-01	Necesidad:	Esencial
Título:	Plataforma Android		
Descripción:	El juego debe ser compatible con la plataforma Android		

Tabla 3.22. RUR-01. Plataforma Android.

Identificador:	RUR-02	Necesidad:	Deseable
Título:	Sonidos		
Descripción:	El jugador deberá poder subir/bajar el volumen del juego tocando los botones de volumen de su Smartphone.		

Tabla 3.23. RUR-02. Sonidos.

Identificador:	RUR-03	Necesidad:	Deseable
Título:	Formatos audio – imagen		
Descripción:	El formato para los ficheros de audio es .WAV y para las imágenes es .PNG		

Tabla 3.24. RUR-03. Formatos audio – imagen.

Identificador:	RUR-04	Necesidad:	Esencial
Título:	Pantalla táctil		
Descripción:	El Smartphone debe ser compatible con el uso de la pantalla táctil.		

Tabla 3.25. RUR-04. Pantalla táctil.

3.1.3. Requisitos de software

Los requisitos de software describen el comportamiento del sistema desarrollado (requisitos funcionales) y el tipo de requisito (requisito de rendimiento, interfaz, operación, recursos, mantenimiento, calidad, seguridad, documentación, comprobación y aceptación de pruebas). Para esta aplicación no se han valorado todos los tipos de requisito.

Al igual que los anteriores requisitos de usuario, los requisitos de software vienen definidos por varios atributos:

- **Identificador**: Indica el tipo de requisito RS (Requisito software) añadiéndole una inicial que muestra si se trata de un requisito funcional (F), de rendimiento (R), de interfaz (I) o de recursos (Re) y un valor numérico. Este identificador debe ser un valor único.

- **Necesidad**: Muestra si la prioridad del requisito es Esencial o Deseable.

- **Título**: Frase breve que indica el nombre del requisito.

- **Fuente**: Muestra los requisitos de usuario en los que se basa este requisito software. Si no procede de ningún requisito de usuario y se incluye por consideración del analista, se identifica como ANA.

- **Descripción**: Breve comentario que describe el requisito. A

continuación se muestran los requisitos funcionales:

Identificador:	RSF-01	Necesidad:	Esencial
Título:	Lanzamiento del juego		
Fuente:	RUC- 01		
Descripción:	La aplicación deberá ser lanzada desde un fichero con extensión .apk generado al compilar e instalado previamente en el terminal.		

Tabla 3.28. RSF-01. Lanzamiento del juego.

Identificador:	RSF-02	Necesidad:	Esencial
Título:	Mostrar menú principal.		
Fuente:	RUC-01, RUC-04, RUC-05		
Descripción:	Al arrancar el juego, la aplicación deberá mostrar una pantalla inicial con las opciones Iniciar juego.		

Tabla 3.29. RSF-02. Mostrar menú principal.

Identificador:	RSF-03	Necesidad:	Esencial
Título:	Elección de modo de juego.		
Fuente:	RUC-01		
Descripción:	Cuando el usuario en el menú principal elija la opción Iniciar juego, se deberá mostrar una pantalla para elegir entre nivel 1 y nivel 2		

Tabla 3.30. RSF-03. Elección de modo de juego.

Identificador:	RSF-04	Necesidad:	Deseable
Título:	Mostrar diálogo pausa		
Fuente:	RUC-06		
Descripción:	Al pulsar el botón atrás del smartphone durante la partida, se deberá mostrar un dialogo, pausando el juego, permitiendo al usuario abandonar la partida.		

Tabla 3.31. RSF-04. Mostrar diálogo pausa.

Identificador:	RSF-06	Necesidad:	Esencial
Título:	Interacción con personaje		
Fuente:	RUR-06		
Descripción:	El smartphone debe ser compatible con el uso pantalla táctil para disparar los proyectiles.		

Tabla 3.33. RSF-06. Interacción con personaje.

Los requisitos de interfaz especifican como es la interfaz con la que el usuario deberá interactuar.

Identificador:	RSI-01	Necesidad:	Deseable
Título:	Imágenes del juego		
Fuente:	RUR-03		
Descripción:	Todas las imágenes del juego tienen formato .PNG		

Tabla 3.38. RSI-01. Imágenes del juego

Los requisitos de recursos indican de qué recursos físicos es necesario disponer para el desarrollo y ejecución de la aplicación.

Identificador:	RSRe-01	Necesidad:	Esencial
Título:	Entorno de desarrollo		
Fuente:	ANA		
Descripción:	El ordenador en el que se desarrollará el juego debe poseer el software necesario para el desarrollo de aplicaciones Android. No es obligatorio el uso de un IDE pero si muy recomendable.		

Tabla 3.40. RSRe-01. Entorno de desarrollo.

Identificador:	RSRe-02	Necesidad:	Esencial
Título:	Entorno de ejecución		
Fuente:	ANA		
Descripción:	El juego se debe ejecutar en un dispositivo que posea el sistema operativo Android 2.1 en adelante.		

Tabla 3.41. RSRe-02. Entorno de ejecución.

3.1.4. Casos de uso

Los diferentes casos de uso sirven para identificar las relaciones existentes entre los actores (el jugador del juego en este caso) y el software, como muestra la figura 3.1.

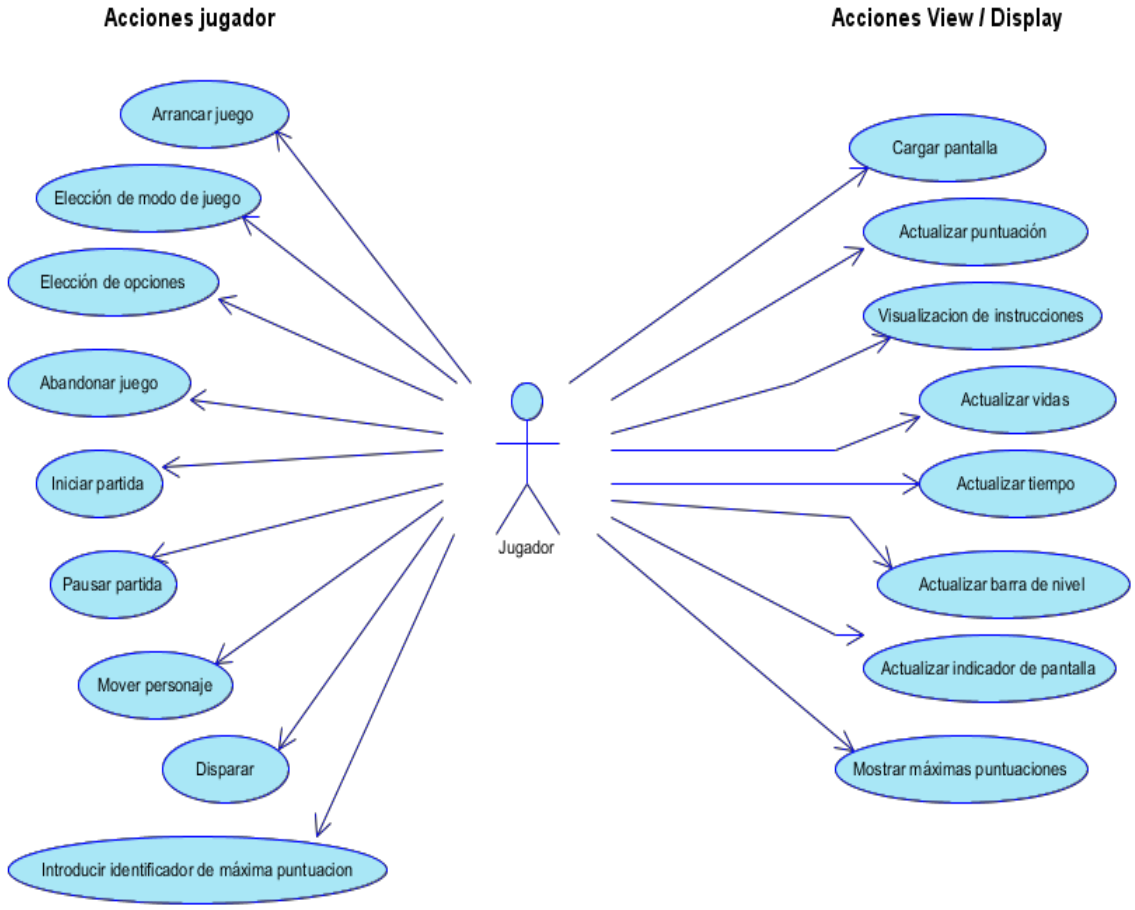

Figura 3.1. Diagrama de casos de uso.

Los casos de uso se representan mediante los siguientes atributos:

- **Identificador**: Muestra el caso de uso (CU) acompañado de un valor numérico. Este identificador debe ser un valor único.
- **Título**: Frase breve que identifica el nombre del caso de uso.
- **Actores**: Principales actores del caso de uso.
- **Objetivo**: Objetivo pretendido por el caso de uso.

- **Pre-condiciones**: Condiciones previas que deben cumplirse para que ocurra el caso de uso.

- **Post-condiciones**: Condiciones producidas a posteriori de que produzca el caso de uso.

A continuación se describen los distintos casos de uso:

Identificador:	CU-01
Título:	Arrancar juego
Actores:	Jugador
Objetivo:	Arrancar la aplicación.
Pre-condiciones:	Haber instalado el juego en el terminal.
Post-condiciones:	Se cargará el menú principal de la aplicación. En él, el usuario puede elegir si iniciar juego, elegir las opciones o leer las instrucciones.

Tabla 3.42. CU-01. Arrancar juego.

Identificador:	CU-02
Título:	Elección de modo de juego
Actores:	Jugador
Objetivo:	Elegir entre nivel 1 y nivel 2.
Pre-condiciones:	Haber seleccionado iniciar juego en el menú principal.
Post-condiciones:	El juego cargará la modalidad elegida y se cargará la pantalla correspondiente esperando a que el jugador pulse la pantalla.

Tabla 3.43. CU-02. Elección de modo de juego.

Identificador:	CU-03
Título:	Abandonar juego.
Actores:	Jugador
Objetivo:	Mostrar al jugador un cuadro de diálogo permitiéndole abandonar el juego.
Pre-condiciones:	Haber pulsado atrás en el menú principal.
Post-condiciones:	Se cerrará la aplicación si el jugador elige Si y se volverá al menú

Tabla 3.44. CU-03. Abandonar juego.

	CU-04
Título:	Cargar pantalla
Actores:	Jugador
Objetivo:	Mostrar al jugador la pantalla inicial de la partida.
Pre-condiciones:	Elegir un modo de juego en la pantalla de selección de modo.
Post-condiciones:	El juego permanecerá parado hasta que el jugador toque la pantalla.

Tabla 3.45. CU-04. Cargar pantalla.

Identificador:	CU-05
Título:	Iniciar partida
Actores:	Jugador
Objetivo:	Comenzar el funcionamiento de la partida, permitiendo la interacción con el jugador.
Pre-condiciones:	Haber pulsado la pantalla una vez cargada la partida.
Post-condiciones:	Se iniciará la partida y el jugador podrá jugar

Tabla 3.46. CU-05. Iniciar partida.

Identificador:	CU-06
Título:	Pausar partida
Actores:	Jugador
Objetivo:	Pausar el juego y mostrar un cuadro de diálogo.
Pre-condiciones:	Haber pulsado atrás en el terminal con la partida arrancada.
Post-condiciones:	El juego permanecerá parado hasta que el jugador seleccione en el cuadro de diálogo la opción No (volver al juego) o la opción Si (abandonar la partida y volver al menú principal).

Tabla 3.47. CU-06. Pausar partida.

Identificador:	CU-07
Título:	Mover personaje
Actores:	Jugador
Objetivo:	Mover el personaje por las zonas de la pantalla habilitadas
Pre-condiciones:	Haber pulsado la pantalla después de mostrarla.
Post-condiciones:	El personaje se desplazará un valor proporcional a la distancia con la que se desplazo en el tirafichas.

Tabla 3.48. CU-07. Mover personaje

Identificador:	CU-08
Título:	Disparar
Actores:	Jugador
Objetivo:	El personaje realizará un disparo.
Pre-condiciones:	Haber pulsado la pantalla táctil durante la partida
Post-condiciones:	Se lanza un proyectil siguiendo la trayectoria calculada con la función desarrollada.

Tabla 3.49. CU-08.Disparar

Identificador:	CU-09
Título:	Actualizar puntuación
Actores:	Jugador
Objetivo:	Actualizar el valor de la puntuación de partida.
Pre-condiciones:	Haber iniciado partida y haber explotado una bola o bloque, o haber conseguido un ítem.
Post-condiciones:	Se actualiza la puntuación de la partida

Tabla 3.50. CU-09. Actualizar puntuación.

Identificador:	CU-10
Título:	Actualizar vidas
Actores:	Jugador
Objetivo:	Actualizar el número de vidas.
Pre-condiciones:	Haber iniciado partida y haber colisionado con una bola o haber conseguido el ítem de vida extra.
Post-condiciones:	Se actualiza el número de vidas de la partida.

Tabla 3.51. CU-10. Actualizar vidas.

Identificador:	CU-11
Título:	Mostrar máximas puntuaciones
Actores:	Jugador
Objetivo:	Mostrar pantalla con máximas puntuaciones.
Pre-condiciones:	Con la partida iniciada, haber perdido todas las vidas o haber llegado hasta el final del modo elegido.
Post-condiciones:	Se muestra una pantalla con las máximas puntuaciones.

Tabla 3.52. CU-11. Mostrar máximas puntuaciones.

3.1.5. Diagrama de actividad del sistema

Los diagramas de actividad muestran la secuencia de actividades que sigue una aplicación mediante diagramas de flujo, desde el punto inicial hasta el punto final indicando muchas de las rutas de decisiones posibles durante la ejecución de la aplicación. Gracias a ellos se comprenden mejor las transiciones y los eventos del juego.

En la figura 15 se muestra el diagrama de actividades respectivo del juego.

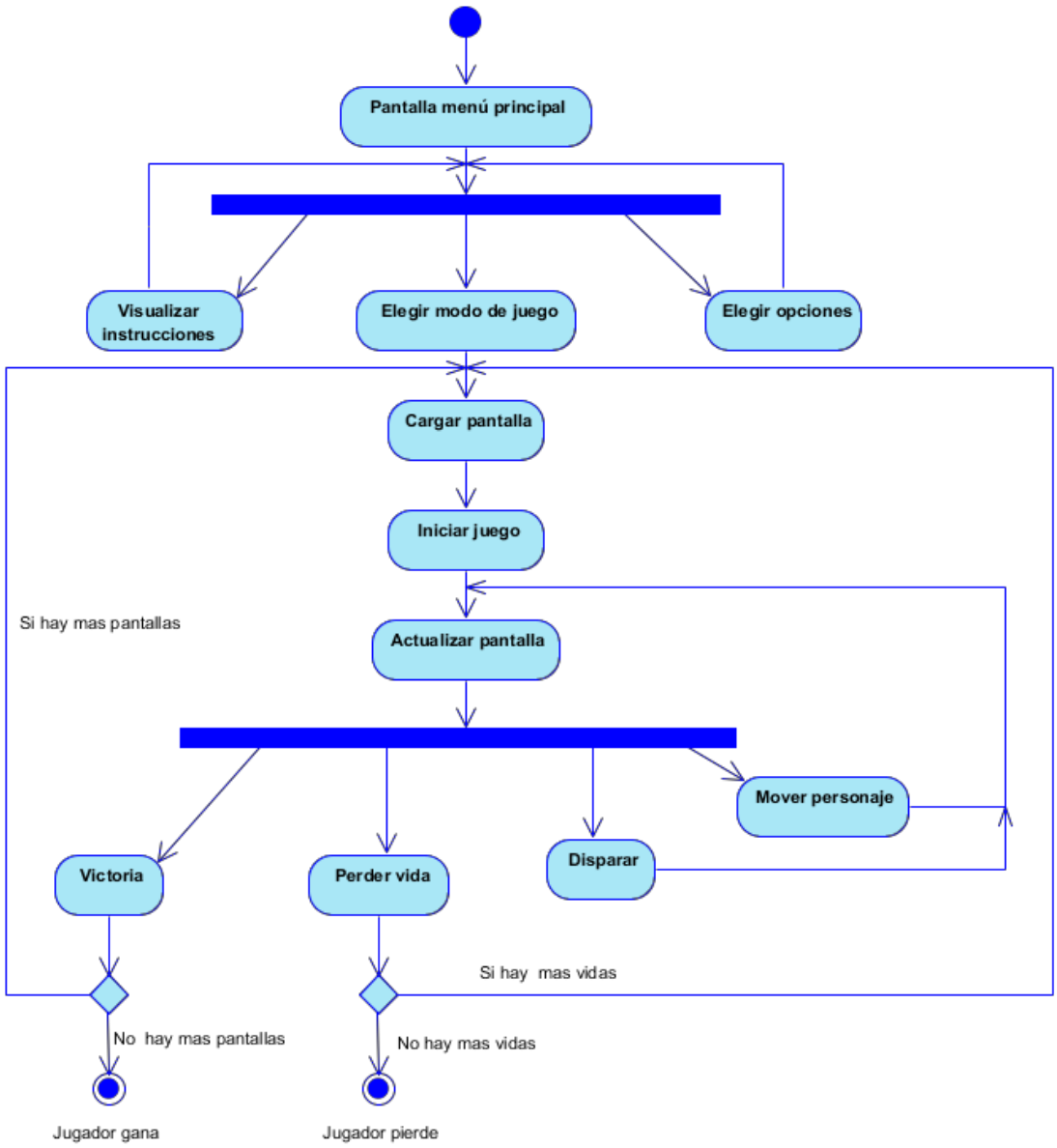

figura 16 Diagrama de Actividades. Flujo del juego.

Al arrancar la aplicación se carga la pantalla del menú principal en la cual el jugador puede tomar tres decisiones: visualizar las instrucciones, ir a la pantalla de opciones o a la de selección de modo de juego. Las dos primeras regresan al menú principal tras cumplir su cometido. La tercera opción carga la partida una vez el jugador ha seleccionado el modo de juego deseado.

3.2. Diseño

Tras realizar el análisis y evaluar qué se quiere hacer, se pasa a la fase de diseño en la que se evalúa cómo se quiere hacer. El diseño debe realizarse sobre todos los componentes de la aplicación y se debe conseguir que todos los aspectos queden completamente definidos de cara a la siguiente fase del proyecto.

El objetivo es realizar un diseño correcto para evitar rectificaciones en la fase de implementación y por lo tanto cumplir los plazos iniciales de entrega del proyecto.

Este apartado se ha dividido en cuatro sub-secciones:

- **Arquitectura**: Definición a alto nivel de la estructura del juego.
- **Interfaces**: Muestra los diseños iniciales realizados para cada pantalla del juego.
- **Carpetas del proyecto**: Muestra el árbol de carpetas de la aplicación.
- **Diagrama de clases**: Muestra las clases y las relaciones entre ellas.

3.2.1. Arquitectura

La arquitectura [17] muestra un diseño a alto nivel de la estructura del juego, en el que se debe definir los módulos principales de la aplicación, las responsabilidades de cada uno, la interacción entre ellos y el control y flujo de datos.

La arquitectura del juego (figura 16) está compuesta por dos módulos principales, el Core de la aplicación y el módulo de Entrada/Salida. El primero debe encargarse de la creación de todos los

componentes y sus ciclos de vida y el segundo se encarga de permitir al usuario interactuar con dichos componentes y mostrar las diferentes vistas de la aplicación.

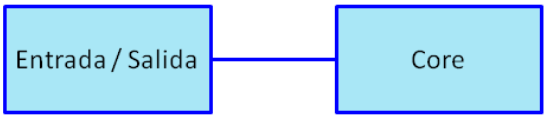

figura 17 Arquitectura del juego. Diagrama de módulos.

3.1.6. Diagramas de secuencia

En los diagramas de secuencia se representa la interacción entre diferentes objetos de una aplicación a lo largo del tiempo para el cumplimiento de los casos de uso, mostran- do los objetos que participan en la interacción y la secuencia de mensajes intercambia- dos entre sí.

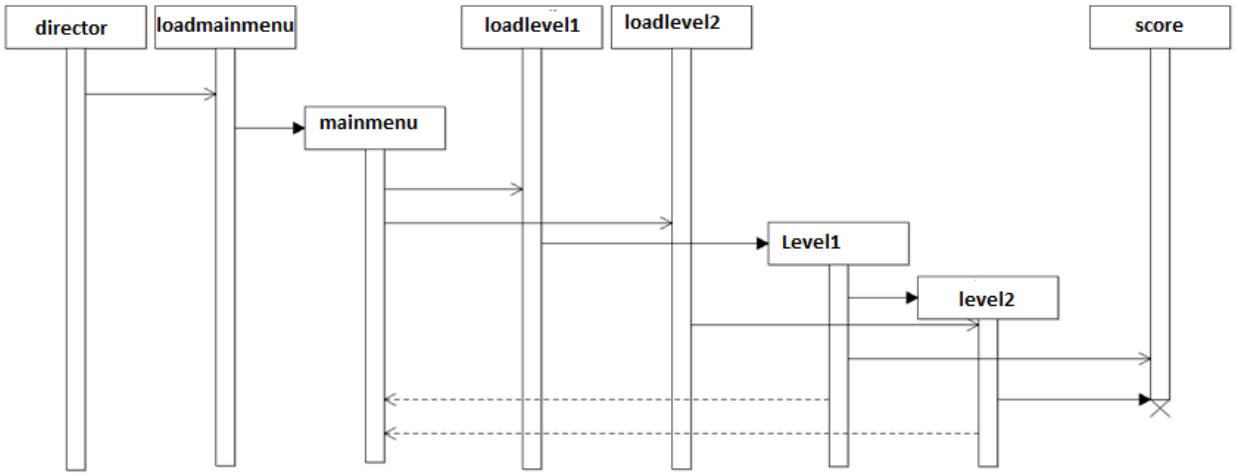

figura 18 Diagrama de secuencia general del juego

3.2.2. Interfaces

En esta sección se muestran los diferentes diseños iníciales que fueron realizados para cada pantalla de la aplicación.

Todas las pantallas de la aplicación se han forzado a mostrarse en posición horizontal por dos motivos. El primero se debe al tamaño limitado de las pantallas. La mayoría de los smartphones poseen una pantalla alargada y eso provoca que este juego se pueda aprovechar mejor horizontalmente. El segundo motivo está relacionado con el primero, ya que nuestro personaje se va a desplazar a lo largo del eje horizontal de la pantalla y para ello es mejor colocar el móvil en posición horizontal para disponer de más recorrido de movimiento.

La figura 18 muestra el diseño inicial de la pantalla del menú principal, mostrado nada más arrancar la aplicación en el terminal. En dicha pantalla aparece la imagen del juego en la zona superior y debajo las diferentes opciones en forma de botón superior y debajo las diferentes opciones en forma de botón.

figura 19 Menú Principal

Si el usuario elige iniciar juego, se muestra el diseño de la pantalla de elección de modo de juego en la que el usuario podrá elegir entre dos opciones de juego representadas por dos imágenes (figura 19).

figura 20 Interfaces. Elección de modo

figura 21 Interface de juego

Una vez iniciado el modo de juego aparece el personaje que cumple las funciones de proyectil y muestra las vidas disponibles en la parte superior izquierda. Como se puede observar en la imagen (figura 18)

Para poder elegir el ángulo y la fuerza con la que se lanzara el proyectil se desliza el personaje utilizando la pantalla touch. (figura 21)

figura 22 Ambiente de Juego

Una vez lanzado el proyectil, este se desplaza dejando un rastro del movimiento parabólico que va dejando su trayectoria (figura 22)

Figura 23 Interacción con el Juego

Después de finalizar la partida, muestra el resultado de los lanzamientos de lograr el objetivo mostrara una pantalla con la leyenda de “Ganaste” en caso contrario mostrará la leyenda de “Perdiste” (figura 23).

figura 24 Marcador

3.3. Implementación

Esta sección explica detalladamente los aspectos relevantes e importantes para la implementación de la aplicación. Sirve como ayuda a este apartado el anterior estudio de análisis y diseño de la aplicación. La implementación se ha realizado siguiendo un incremento funcional progresivo, esto es, comenzar desde la funcionalidad más básica y progresivamente ir incrementando la funcionalidad, permitiendo en todo momento que la aplicación sea operativa.

Este capítulo se estructura de manera similar al desarrollo del juego: comienza con la creación de actividades y como navegar entre ellas o la carga de interfaces de usuario en las distintas actividades y posteriormente se desarrolla el núcleo principal del juego. El último paso es añadir detalles como música, efectos de sonido o diferentes acciones en caso de conseguir una victoria o perder una vida.

3.3.1. Desarrollo móvil multiplataforma

Antes de comenzar con la implementación de la aplicación, se ha considerado necesario una investigación sobre las herramientas existentes para el desarrollo móvil en android y de ser posible algún otro sistema operativo.

3.3.1.2. PhoneGap

PhoneGap es un sistema para crear aplicaciones usando HTML5, CSS3 y Javascript, ejecutadas dentro en un componente WebKit del móvil. Provee una serie de librerías Javascript desarrolladas en el lenguaje específico de cada plataforma (Objetctive---C para IOS, Java para Android, etc) que permiten acceder a las características del móvil como GPS, acelerómetro, cámara, contactos, base de datos, filesystem, etc.

Al ser una página web, se tiene acceso al DOM y se puede usar frameworks web como jQuery o cualquier otro. Requiere diseñar la aplicación web con los componentes visuales típicos del HTML, etc o usar un framework web mobile como jQuery Mobile o Sencha Touch entre otros. Tiene la ventaja de que se puede definir la navegación inicial de la aplicación usando un navegador en un ordenador, sin tener que ejecutarla en el simulador.

Se puede ver una aplicación PhoneGap como una serie de páginas web que están almacenadas y empaquetadas dentro de una aplicación móvil, visualizadas con un navegador web, con acceso a la mayoría de APIs del móvil, lo cual lo convierte en una alternativa muy sencilla para crear aplicaciones.

Para trabajar con cada plataforma hay que usar un sistema distinto: para Iphone/Ipad es necesario usar Xcode (solo disponible en Mac) y una plantilla de proyecto que proporciona PhoneGap. Para Android se debe usar Eclipse (Windows, Mac y Linux) y otra plantilla de proyecto específica. Y para Blackberry no hay entorno: solo Java SDK, BlackBerry SDK y Apache Ant.

Ventajas:

- Es la solución que más plataformas móviles soporta, ya que corre dentro de un navegador web. Además de Iphone/Ipad y Android, funciona también en Palm, Symbian, WebOS, Windows mobile 7 y BlackBerry,
- Es muy fácil de desarrollar y proporciona una gran libertad a los que tienen conocimientos de HTML y Javascript.
- Hay buena documentación y bastantes ejemplos.
- Es gratis, soporte de pago. Licencia BSD.

Inconvenientes:

- Requiere Mac con Xcode para empaquetar aplicaciones IOS.
- La aplicación no es más que una página web, por lo que el aspecto dependerá del framework web utilizado. Necesitaremos el uso de frameworks HTML móviles si queremos que parezca una aplicación nativa.
- No llega al rendimiento de una aplicación nativa, pues el HTML, CSS y Javascript debe ser leído e interpretado por el motor del navegador cada vez arranca.

3.3.1.3. Titanium Appcelerator

Con Appcelerator es posible crear aplicaciones para Android, Iphone y de escritorio, usando exclusivamente Javascript (el soporte para Blackberry está en fase beta).

Para programar proporciona Titanium Studio, un IDE basado en Eclipse con el que crear los proyectos y editar los ficheros Javascript y el resto de recursos y lanzar los scripts de creación.

Las aplicaciones se programan íntegramente con Javascript, creando y colocando "a mano" todos los controles, usando para ello una librería que hace de puente entre la aplicación Javascript y los controles del sistema. Esto significa que las ventanas y demás controles visuales (botones, listas, menús, etc) son nativos: cuando se añade un botón, se crea un botón del sistema y se añade a la vista, lo que lo hace más rápido de renderizar y la respuesta del usuario es también rápida.

Una de las características más interesantes de Appcelerator es que al empaquetar la aplicación, el Javascript es transformado y compilado. Después, cuando se arranca la aplicación en el móvil, el código se ejecuta dentro de un engine Javascript, tal y como dice la documentación oficial, que será JavaScriptCore en IOS (el intérprete de Webkit, el motor de Safari y Chrome) y Mozilla Rhino en Android/Blackberry.

El hecho de que el Javascript esté compilado y que los controles creados sean nativos, le hace tener mejor rendimiento posible en comparación con PhoneGap o Adobe Air para móviles y similar a Corona SDK.

Con Appcelerator es complicado maquetar, pues no existe un HTML inicial donde añadir los controles, sino que hay que crear las ventanas y componentes "a mano" con Javascript.

Los desarrollos de las librerías Javascript para cada sistema operativo evolucionan por separado por lo que es posible que no funcionen de la misma manera. A diferencia de PhoneGap, que solo tiene una librería Javascript para acceder a las características especiales del sistema, Appcelerator necesita además librerías para manejar los controles nativos y su disposición en la pantalla, por lo que el desarrollo en general es más costoso.

Para iOS, Titanium Studio genera un proyecto Xcode con el Javascript transformado junto con todas las librerías necesarias. Después es posible lanzar el simulador con la aplicación en Xcode sin salir de Titanium Studio. Una vez generado el proyecto, éste se puede abrir con Xcode y continuar empaquetándolo y configurándolo para su distribución (certificados, provisioning, logos, splash screen, etc). Desde Xcode no se puede editar el JavaScript, se debe volver a editar en Titanium Studio y regenerar el proyecto Xcode otra vez.

Sobre el soporte Android, tanto para probar en el simulador como para empaquetar la aplicación, solo hay que tener el SDK de Android instalado.

Ventajas:

- Multiplataforma móvil y también de escritorio. * Aspecto y controles nativos. Buen rendimiento. * Buenos ejemplos
- Gratis, soporte de pago. Licencia Apache.

Desventajas:

- Definición de componentes visuales y ubicación de controles compleja
- Mucha documentación pero poco actualizada
- Requiere Mac y Xcode para empaquetar aplicaciones IOS.

3.3.1.4. Adobe Air Mobile

Adobe Air mobile funciona con Flex 4 y soporta las plataformas IOS, Android y BlackBerry Tablet, además de los sistemas de escritorio Windows, Mac y Linux (a través de un runtime). Flex 4 utiliza el lenguaje de programación ActionScript, de tipado fuerte y con clases, interfaces, herencia y paquetes muy parecido a Java con el que poder hacer complejos desarrollos.

El IDE oficial, Flash Builder 4.5 es un IDE muy potente y es de pago. Es posible compilar y empaquetar las aplicaciones con el Flex SDK opensource y gratuito pero es más complejo. Los controles visuales usados durante el desarrollo y ejecución no son los originales de cada plataforma, sino que son específicos de Flex 4. Esto garantiza que todas las aplicaciones tendrán exactamente el mismo aspecto y comportamiento.

Se pueden depurar aplicaciones en remoto. Una de las peculiaridades es que es la única herramienta que no requiere ni el Android SDK ni el Xcode para Mac para ejecutar y crear las aplicaciones.

Ventajas:

- Multiplataforma móvil y también de escritorio.
- ActionScript es un lenguaje muy potente que permite el uso de patrones y estructuras complejas en los desarrollos.
- Desarrollo y definición de las vistas con el editor visual de MXML con Flash Builder. El IDE y Flex 4 son muy potentes, y la documentación buena.
- Flash Builder 4.5 no requiere el uso de Xcode ni Mac.
- Depuración remota.

Desventajas:

- El precio de Flash Builder 4.5. Aunque hay otras herramientas y se puede usar el SDK gratuito.
- No funciona en todos los Android, solo en los de gama alta que tengan arquitectura Arm7.
- Rendimiento es regular y la renderización no es suave en IOS.
- Aspecto no nativo (aunque homogéneo entre todas las plataformas).

3.3.1.5. Corona SDK

Corona es un framework para el desarrollo de aplicaciones gráficas para iOS/Android de la compañía Ansca Mobile. Se desarrolla en Lua y no tiene IDE.

Contiene varios simuladores para cada uno de los dispositivos para los que se puede desarrollar y un depurador por línea de comandos. El emulador dispone así mismo de un terminal que permite mostrar mensajes de trazas durante la ejecución.

No son necesarios conocimientos de Objective---C/Cocoa, C + + o Java. Lua es un lenguaje de programación imperativo, estructurado y bastante ligero que fue diseñado como un lenguaje interpretado con una semántica extendible.

Contiene un conjunto limitado de librerías propias de Lua así como otras propias del SDK que aportan la funcionalidad y apariencia propia de estos dispositivos móviles, en especial de los de Apple.

Hay que pagar una licencia para cada plataforma o una conjunta para IOS/Android.

Ventajas:

- Alto rendimiento en la ejecución de aplicaciones.
- Motor gráfico y físico ideal para juegos.
- Lua es un lenguaje bastante sencillo y potente.

Desventajas:

- El precio de la licencia anual.
- Aunque se puede usar para cualquier tipo de aplicación, realmente es ideal para aplicaciones gráficas y juegos.
- Está en evolución.

3.3.2 Plataforma seleccionada

Después del estudio de las características de los distintos entornos multiplataforma, estás son las conclusiones.

En todos los casos, los lenguajes de las plataformas, Lua, JavaScript y ActionScipt, son suficientemente potentes y además, sencillos de implementar.

Para el diseño de la aplicación, el más avanzado es el de adobe que tiene su propio editor. El resto necesitan de una colocación un tanto manual, aunque en Corona SDK se pueden crear grupos que facilitan el diseño.

Appcelerator y Corona son los únicos que permiten crear controles nativos de cada plataforma aunque la utilización y ubicación gráfica de estos en Appcelerator es más compleja. Adobe permite usar sus propios componentes con resultado homogéneo en todas las plataformas.

PhoneGap es el que más sistemas operativos soporta. El resto cubren las más importantes Android e iOS.

El rendimiento de la aplicación es muy bueno en Corona SDK así como también en Appcelerator. Las otras dos opciones son menos rápidas aunque con un rendimiento aceptable.

Con respecto a la documentación disponible, Corona SDK es la que tiene más disponible y con una amplia comunidad en la página web de la compañía. Es una herramienta en evolución pero la documentación está siempre actualizada. La documentación en Adobe también es bastante completa. Las otras dos son herramientas jóvenes que todavía les falta para estar acabadas: sus APIs cambian, están incompletas y a veces fallan y la documentación es regular.

En cuanto a la distribución, la plataforma de Adobe es la única que no necesita tener un Mac para desarrollar para iOS.

Appcelerator y PhoneGap son gratuitos, solo hay que pagar para el soporte. Flash Builder 4.5 Premium tiene un precio muy elevado, aunque se puede usar la versión de prueba durante 30 días y hay SDK libres. Corona SDK es gratis para desarrollo, pero requiere pagar una licencia anual de $199 si quieres subir tus aplicaciones al App Store o Market de Android o $349 para ambos.

Con estos criterios, las dos mejores opciones por rendimiento y por posibilidad de utilizar componentes nativos son Corona SDK y Appcelerator. Eliminando la restricción económica asociada a las licencias de Corona SDK, y dado que la gestión gráfica es mejor y la documentación más completa en esta plataforma, se selecciona Corona SDK como base para el desarrollo de la aplicación móvil de este proyecto.

3.3.3. Plataforma Corona SDK. Características

3.3.3.1 Introducción a la Plataforma Corona SDK

La plataforma Corona SDK es un entorno de desarrollo de aplicaciones móviles para la creación de aplicaciones de altas prestaciones, aplicaciones multimedia y juegos para dispositivos iOS, Android y Kindle.

Corona SDK contiene un simulador para cada uno de los dipositivos con sistema operativo móvil de Apple, y para varios modelos de dispositivos con sistema operativo Android. Tambien contiene un depurador por línea de comandos así como aplicaciones de ejemplo y documentación.

Permite a los desarrolladores usar Lua, un lenguaje de scrip de alto rendimiento construido sobre un motor de Objective---C/C++. No son necesarios conocimientos de Objective---C/Cocoa, C + + o Java. Lua es un lenguaje de programación imperativo, estructurado y bastante ligero que fue diseñado como un lenguaje interpretado con una semántica extendible.

Contiene un conjunto limitado de librerías propias de Lua así como otras propias del SDK

3.3.3.2. Gestión de proyectos en Corona SDK

Para crear aplicaciones en Corona SDK se necesita instalar la aplicación Corona SDK en entorno Mac o Windows y utilizar un editor de texto para generar los archivos con el código. La versión de prueba de Corona SDK nos permite realizar pruebas con el simulador en dispositivos Android o iOS (solo en equipos Mac). Para construir aplicaciones y distribuirlas en la AppStore o el android Market es necesario comprar una licencia.

Para crear un proyecto de Corona es necesario como minimo una carpeta que contenga un archivo de texto llamado "main.lua". Este archivo, "main.lua", es el primer archivo que lee Corona SDK por lo que si no esta disponible la aplicación no puede iniciarse. Este archivo principal, a su vez, puede cargar otros archivos de código externo, o recursos de otros programas, tales como sonidos,imágenes o videos. La extensión de archivo ". lua" indica que el archivo está escrito en lenguaje en ese lenguaje, que es el que se usa para crear aplicaciones en Corona SDK.

De cara a la construcción de la aplicación, en la carpeta del proyecto existen 2 posibilidades de configuración mediante ficheros. El primero es el fichero config.lua con las dimensiones del contenido visible y el modo de escalado de la pantalla. Este archivo se comenta más adelante en la implementación del proyecto al tratar la adaptación

dinámica de contenido en la pantalla. También se debe añadir un archivo Icon.png que

será el icono de la aplicación al instalarse en el dispositivo final . El segundo fichero que podemos incluir es el denominado build.settings que describe las propiedades en tiempo de construcción. Más adelante, en las tareas de administración se comentarán las opciones de este archivo.

3.3.4. Lenguaje de Corona SDK: Lua

3.3.4.1. Generalidades

Lua es un lenguaje de programación imperativo, estructurado y bastante ligero que fue diseñado como un lenguaje interpretado con una semántica extendible.. Fue creado en 1993 en la universidad católica de Rio de Janeiro y cuyo nombre significa "Luna" en portugués. Es muy utilizado en programación de videojuegos asi como en aplicaciones para videoconsolas como PSP y Wii.

3.3.4.2 Características

Lua es un lenguaje de extensión, suficientemente compacto para usarse en diferentes plataformas. En lua las variables no tienen tipo, sólo los datos y pueden ser lógicos, enteros, números de coma flotante o cadenas.

Estructuras de datos como vectores, conjuntos, tablas hash, listas y registros pueden ser representadas utilizando la única estructura de datos de Lua: la tabla.

La semántica de Lua puede ser extendida y modificada redefiniendo funciones de las estructuras de datos utilizando metatablas, casi como en Perl. Lua ofrece soporte para funciones de orden superior, recolector de basura. Combinando todo lo anterior, es posible utilizar Lua en programación orientada a objetos.

3.3.4.5 Funcionamiento interno

Los programas en Lua no son interpretados directamente, sino compilados a código bytecode, que es ejecutado en la máquina virtual de Lua. El proceso de compilación es normalmente transparente al usuario y se realiza en tiempo de ejecución, pero puede hacerse con anticipación para aumentar el rendimiento y reducir el uso de la memoria al prescindir del compilador.

4 Conclusiones

El primer objetivo, realización de un videojuego para la plataforma Android, ha sido conseguido satisfactoriamente. El juego desarrollado funciona en terminales con sistema operativo Android. Otro de los objetivos planteados fue aprovechar las ventajas que ofrece Android También ha sido cumplido este objetivo ya que se ha conseguido distintas funcionalidades:

- **Pantalla táctil**: Se ha aprendido a utilizar la pantalla táctil y a interactuar con los diferentes objetos de la pantalla, además de detectar las pulsaciones y realizar alguna acción una vez pulsada la pantalla.

- **Botones**: Se ha aprendido a mostrar botones en pantalla, detectar su pulsación y ejecutar alguna acción tras detectar su pulsación. También se ha podido modificar su aspecto visual.
- **Cuadros de diálogo**: Se ha conseguido diseñar diferentes cuadros de diálogo para permitir al usuario abandonar la partida o aplicación.
- **Sonidos**: Se han implementado diferentes tipos de sonido para diferentes situaciones.
- **Lua**: Se aprendió a utilizar este lenguaje de programación imperativo.
- **Corona SDK**: Se utilizaron las herramientas proporcionadas por este entorno de desarrollo, con las cuales se desarrollo la totalidad del proyecto.

4.1 Futuros desarrollos y ampliaciones

Se considera que este proyecto puede servir como base y ayudar a aquellos estudiantes que deseen iniciarse en el desarrollo de aplicaciones o juegos en Android.

También se ha querido dotar al juego de una funcionalidad básica que lo hicieran completamente jugable. Este objetivo se ha cumplido y se permite que futuros desarrolladores mejoren o amplíen su funcionalidad.

APENDICE

Ejemplificación del una aplicación en LUA

Aplicación generada para utilizar las propiedades graficas aplicando un tiro parabólico que resulta de un rebote en un objeto elástico.
El objetivo es derivar el escenario con la finalidad de estrellar los huevos que están dentro de el, a través de una bola que rebota en un trampolín, el jugador puede soltar la bola a la altura y posición que el desee, afectando así su fuerza y el Angulo con el que sale expedida, como se muestra en la figura A1

figura A1 evolución del juego

Código Fuente

```
local widget = require( "widget" )

-- Se active el Motor grafíco
local physics = require("physics")
physics.start()

display.setStatusBar( display.HiddenStatusBar )

-- Se llaman algunas bibliotecas externas
local movieclip = require( "movieclip" )

local ballInPlay = false

-- Nuevo grupo maestro (Para la camara global)
local game = display.newGroup();
game.x = 0

------------------------------------------------------------
-- Definir efectos de audio

local boingSound = audio.loadSound("boing_wav.wav")
local knockSound = audio.loadSound("knock_wav.wav")
local squishSound = audio.loadSound("squish_wav.wav")

------------------------------------------------------------
-- Gráficos de suelo y cielo

local sky = display.newImage( "sky.png", true )
game:insert( sky )
sky.x = 160; sky.y = 160

local sky2 = display.newImage( "sky.png", true )
game:insert( sky2 )
sky2.x = 1120; sky2.y = 160

local grass = display.newImage( "grass.png", true )
game:insert( grass )
grass.x = 160
grass.y = 440
physics.addBody( grass, "static", { friction=0.5, bounce=0.3 } )

local grass2 = display.newImage( "grass.png", true )
game:insert( grass2 )
grass2.x = 1120
grass2.y = 440
physics.addBody( grass2, "static", { friction=0.5, bounce=0.3 } )
```

```
local arrow = display.newImage( "arrow.png" )
game:insert( arrow ); arrow.x = 50; arrow.y = 120

local instructionLabel = display.newText( "Toca Para empezar", 70, 10 + display.screenOriginY,
native.systemFontBold, 18 )
instructionLabel:setTextColor( 235, 235, 255, 255 )
game:insert( instructionLabel )


------------------------------------------------------------
-- Se agrega el Trampolin

trampoline = display.newImage( "trampoline.png" )
game:insert( trampoline )
physics.addBody( trampoline, "static", { friction=0.5, bounce=1.2 } )
trampoline.x = 50
trampoline.y = 355
trampoline.rotation = 45



-------------------------------------------------------------------------
-- Construcción del escenario

castleBody = { density=4.0, friction=1, bounce=0.2 }
castleBodyHeavy = { density=12.0, friction=0.3, bounce=0.4 }

wall1 = display.newImage( "wall.png" )
game:insert( wall1 ); wall1.x = 632; wall1.y = 350
physics.addBody( wall1, castleBody )

wall2 = display.newImage( "wall.png" )
game:insert( wall2 ); wall2.x = 744; wall2.y = 350
physics.addBody( wall2, castleBody )

wall3 = display.newImage( "wall.png" )
game:insert( wall3 ); wall3.x = 856; wall3.y = 350
physics.addBody( wall3, castleBody )

roof = display.newImage( "roof.png" )
game:insert( roof ); roof.x = 684; roof.y = 292
physics.addBody( roof, castleBody )

roof2 = display.newImage( "roof.png" )
game:insert( roof2 ); roof2.x = 804; roof2.y = 292
physics.addBody( roof2, castleBody )
```

```
beam = display.newImage( "beam.png" )
game:insert( beam ); beam.x = 694; beam.y = 250
physics.addBody( beam, castleBodyHeavy )

beam2 = display.newImage( "beam.png" )
game:insert( beam2 ); beam2.x = 794; beam2.y = 250
physics.addBody( beam2, castleBodyHeavy )

roof3 = display.newImage( "roof.png" )
game:insert( roof3 ); roof3.x = 744; roof3.y = 210
physics.addBody( roof3, castleBody )

beam3 = display.newImage( "beam.png" )
game:insert( beam3 ); beam3.x = 744; beam3.y = 168
physics.addBody( beam3, castleBodyHeavy )


-------------------------------------------------------------
-- Construccion de los huevos

eggBody = { density=1.0, friction=0.1, bounce=0.5, radius=25 }

-- Uses "movieclip" library for simple 2-frame animation; could also use sprite sheets for more complex animations
egg1 = movieclip.newAnim{ "egg.png", "egg_cracked.png" }
game:insert( egg1 ); egg1.x = 684; egg1.y = 374; egg1.id = "egg1"
physics.addBody( egg1, eggBody )

egg2 = movieclip.newAnim{ "egg.png", "egg_cracked.png" }
game:insert( egg2 ); egg2.x = 802; egg2.y = 374; egg2.id = "egg2"
physics.addBody( egg2, eggBody )

egg3 = movieclip.newAnim{ "egg.png", "egg_cracked.png" }
game:insert( egg3 ); egg3.x = 744; egg3.y = 258; egg3.id = "egg3"
physics.addBody( egg3, eggBody )


-------------------------------------------------------------
-- Display de marcador simple
local scoreDisplay = display.newText( "0", 0, 0, native.systemFontBold, 32 )
scoreDisplay:setReferencePoint( display.CenterRightReferencePoint )
scoreDisplay.x = display.contentWidth - 25
scoreDisplay.y = 40

score = 0

-------------------------------------------------------------
-- Lanzador de proyectil
```

```
local boulder = display.newImage( "boulder.png" )
game:insert( boulder )

-- initial body type is "kinematic" so it doesn't fall under gravity
physics.addBody( boulder, { density=15.0, friction=0.5, bounce=0.2, radius=36 } )

local function resetBoulder()
	boulder.bodyType = "kinematic"
	boulder.x = 30
	boulder.y = -140
	boulder:setLinearVelocity( 0, 0 ) -- stop boulder moving
	boulder.angularVelocity = 0 -- stop boulder rotating
end

resetBoulder()

-- Camera follows bolder automatically
local function moveCamera()
	if (boulder.x > 80 and boulder.x < 1100) then
		game.x = -boulder.x + 80
	end
end


Runtime:addEventListener( "enterFrame", moveCamera )


------------------------------------------------------------
-- Se agrega un escucha par alas coliciones

function startListening()
	-- if egg1 has a postCollision property then we've already started listening
	-- so return immediately
	if egg1.postCollision then
		return
	end

	local function onEggCollision ( self, event )

		-- uses "postSolve" event to get collision force

		print( "force: " .. event.force )

		-- Crack this egg if collision force is high enough
		if ( event.force > 6.0 ) then
			self:stopAtFrame(2)
```

```
                audio.play( squishSound )
                score = score + 150
                scoreDisplay.text = score

                -- After this egg cracks, it can ignore further collisions
                self:removeEventListener( "postCollision", self )
            end
        end

        -- Set table listeners in each egg to check for collisions
        egg1.postCollision = onEggCollision
        egg1:addEventListener( "postCollision", egg1 )

        egg2.postCollision = onEggCollision
        egg2:addEventListener( "postCollision", egg2 )

        egg3.postCollision = onEggCollision
        egg3:addEventListener( "postCollision", egg3 )

end


-------------------------------------------------------------
-- Agrega un escucha para la colisión con el trampolín

local function onBounce ( self, event )
        audio.play( boingSound )
end

trampoline.collision = onBounce
trampoline:addEventListener( "collision", trampoline )


-------------------------------------------------------------
-- Agrega un escucha para la colisión con el trampolín

local function dropBoulder ( event )
        if ( not ballInPlay ) and ( event.phase == "began" ) then
            ballInPlay = true
            audio.play( knockSound )
            boulder.x = event.x - game.x
            boulder.y = event.y
            -- change body type to dynamic, so gravity affects it
            boulder.bodyType = "dynamic"

            startListening()
        end
end
```

```
local function newRound( event )
	resetBoulder()
	game.x = 0
	ballInPlay = false
	return true
end

local resetButton = widget.newButton{
	default = "buttonRed.png",
	over = "buttonRedOver.png",
	onPress = newRound,
	label = "Bola Extra",
	labelColor = { default = { 255 } },
	emboss = true
}

resetButton.x = 160
resetButton.y = 450

-------------------------------------------------------------
-- tiempo de espera antes de empezar el juego
timer.performWithDelay( 3000, startListening )

-- finalmente se agrega un escucha en el cielo, para obtener nuevos misiles
sky:addEventListener( "touch", dropBoulder )
```

Bibliografía

[1] M. Brownlow. (2012) Smartphone sales and statistics. [Online] Available: http://www.email-marketing-reports.com/wireless-mobile/smartphone-statistics.htm

[2] Gartner. (2012) Gartner says android to command nearly half of worldwide smartphone operating system market by year-end 2012. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1622614

[3] A. Wells (2012) Android app statistics summary for 2010. [Online]. Available: http://www.androidtapp.com/android-apps-statistics-summary-for-2010/

[4] Research2Guidance. (2012) Android market insights august 2011. [Online]. Available: http://www.research2guidance.com/shop/index.php/downloadable/download/sample/sample_id/148/

[5] J. Rodríguez. (2012) Breve historia de los smartphones. [Online]. Available: http://somosgeeks.wordpress.com/2011/01/17/breve-historia-de-los-smartphones/

[6] Samsung Electronics Co, Ltd. (2012) Samsung‟s smartphone platform. [Online]. Available: http://www.bada.com/index.html

[7] The Linux Foundation. (2012) An open source, standards-based software platform for multiple devices categories. [Online]. Available: https://www.tizen.org/

[8] Gartner. (2012) Gartner Says Sales of Mobile Devices in Second Quarter of 2011 Grew 16.5 Percent Year-on-Year; Smartphone Sales Grew 74 Percent. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1764714

[9] teleco.com.br. (2012) Ventas mundiales para usuarios finales y market share. [Online]. Available: http://www.teleco.com.br/es/es_smartphone.asp

[10] phonecurry.com. (2012) The Ultimate Smartphone Shootout. [Online]. Available: http://www.phonecurry.com/knowledge-series/smartphone-comparison/

[11] Danny. (2012) La historia y los comienzos de Android, el sistema operativo de Google. [Online]. Available: http://www.elandroidelibre.com/2011/08/la-historia-y-los- comienzos-de-android-el-sistema-operativo-de-google.html

[12] O. H. Alliance (2012) Open Handset Alliance. [Online]. Available: http://www.openhandsetalliance.com/

[13] Awe. (2012) Historia de la plataforma Android. [Online] Available: http://android.scenebeta.com/tutorial/historia-de-la-plataforma-android

[14] Android developers. (2012) What is Android?. [Online]. Available: http://developer.android.com/guide/basics/what-is-android.html

[15] Android developers (2012) Application Fundamentals. [Online]. Available: http://developer.android.com/guide/topics/fundamentals.html

[16] T. G. Piedrabuena. (2012) Especial: Super Pang. [Online]. Available: http://metodologic.com/index.php?option=com_content&view=article&id=2529:especi al-pang&catid=32:metodolog

[17] J. Casanovas. (2004) Usabilidad y arquitectura de software. [Online]. Available: http://www.desarrolloweb.com/articulos/1622.php

[18] sgoliver.net (2012) Estructura de un proyecto Android. [Online]. Available: http://www.sgoliver.net/blog/?p=1278

[19] Android developers (2012) Technical Resources. [Online]. Available: http://developer.android.com/resources/browser.html?tag=tutorial

[20] Android developers (2012) LunarLander. [Online]. Available: http://developer.android.com/resources/samples/LunarLander/index.html

[21] Android developers (2012) XML Layouts. [Online]. Available: http://developer.android.com/guide/topics/ui/declaring-layout.html

[22] tecnoPLOF.com (2012) Jugabilidad. [Online]. Available: http://www.teknoplof.com/2010/09/17/jugabilidad-ese-concepto-que-no-aparece-en-el-diccionario/

[23] J. L. Gonzalez. Sánchez. (2012). Jugabilidad. Caracterización de la experiencia del jugador de videojuegos. [Online]. Available: digibug.ugr.es/bitstream/10481/5671/1/18931200.pdf

[24] Android Market (2012) Aplicaciones. [Online]. Available: https://play.google.com/store