Escuela Politécnica Superior

Ingeniería de Telecomunicación

Proyecto Fin de Carrera

Diseño e implementación de un juego para smartphones con Android: Los Colonos de Catán

Autora: Cristina Fernández Jiménez

Tutor: Juan Peralta Donate

Curso Académico 2010/2011

Proyecto Fin de Carrera

Diseño e implementación de un juego para smartphones con Android: Los Colonos de Catán

Autor

Cristina Fernández Jiménez

Tutor

Juan Peralta Donate

La defensa del presente Proyecto Fin de Carrera se realizó el día 13 de Octubre de 2011, siendo evaluada por el siguiente tribunal:

Presidente:

Vocal:

Secretario:

y habiendo obtenido la siguiente Calificación:

Leganés, a 13 de Octubre de 2011

A los que han estado a mi lado

Agradecimientos

Esta memoria pone punto y final a una etapa muy importante de mi vida y quiero agradecer a todas las personas que me han acompañado durante el camino y me han ayudado a conseguir llegar al final.

En primer lugar quiero dar las gracias a mis padres, MariCruz y Juan. Por estar siempre ahí, darme ánimos cuando ya no tenía ganas ni fuerza, por la metáfora de la montaña y por seguir insistiendo para que terminase el proyecto. Y sobre todo, por lo que he aprendido de ellos en todos los aspectos.

A todos mis amigos, por compartir momentos de ocio y también de estudios, en especial a Miguel, Iñaqui, Lucía, Teresa y Jesús.

A Juan Peralta, por darme la oportunidad de llevar a cabo este proyecto. Por su ayuda incondicional siempre que se la he pedido y por sus consejos.

Para terminar y muy especialmente a Rafa, por darme el último empujón que tanta falta me hacía. Y por estar ahí.

Contenido

Índice de figuras

Índice de tablas

Resumen

En los últimos años la industria de la telefonía móvil, y los diferentes campos de desarrollo y comercialización relacionados con ella, han experimentado una gran evolución. Uno de los puntos claves de esta evolución ha sido y es, el uso de los smartphones con Android. A día de hoy, se producen más de 500.000 activaciones diarias de dispositivos Android, y el crecimiento semanal es del 4,4 %.

El primer terminal Android fue diseñado y fabricado por HTC, que durante 2010 vendió 24,6 millones de smartphones. El incremento de los beneficios de HTC con respecto a 2009, fue de un 134 %.

Con estos datos de crecimiento tanto de Android como de HTC, se ha querido estudiar las posibilidades de ambos implementando una aplicación.

En el presente proyecto, se ha desarrollado un juego bajo la plataforma Android, para el terminal HTC Legend. Es la versión para móvil de "Los Colonos de Catán", juego de tablero en el que el ganador es el jugador que más poblados, ciudades y caminos construye a lo largo de la partida.

Abstract

In the last years, mobile phone industry, and everything related to it, has experienced a great evolution. One of the key point of this evolution has been Android smartphone usage. Up to today, more than 500,000 Android devices are activated every day, and the weekly growth is 4.4 %.

The fist Android device was designed and built by HTC, which during 2010 sold 24.6 millions of Smartphones. The increase of benefits from 2009 to 2010 was of 134 %.

With this data growth, both Android and HTC, an application has been developed to study their possibilities.

The aim of this project is developing a game under the Android platform, for an HTC Legend device. The game will be a mobile version of "The Settlers of Catan", board game in which the winner is the player who builds more cities, settlements and paths during the game.

Glosario

ADT	Android Development Tools
HVGA	Half-size Video Graphics Array
IDE	Integrated Development Environment
Java SE	Java Standard Edition
JDK	Java Development Kit
JRE	Java Runtime Environment
SDK	Software Development Kit
UML	Unified Modeling Language
XML	Extensible Markup Language

Capítulo 1

Introducción

En el capítulo introductorio se presenta un enfoque general de los propósitos y objetivos de este proyecto. Los puntos principales del mismo son la motivación, los objetivos, las fases, los medios utilizados y los contenidos.

En el primer apartado (**Motivación** 1.1), se define la motivación, el por qué se ha realizado y que aspectos son los han movido para la elección de este proyecto y no otro.

En los **Objetivos** 1.2, se muestran los distintos puntos que se persiguen en el proyecto y que se quiere alcanzar y conseguir con ellos.

Para poder llevar a cabo los objetivos, es necesario definir unas **Fases** 1.3 del proyecto que marcarán su ciclo de vida.

También se detallan los **Medios** 1.4 utilizados durante el diseño e implementación: recursos tanto software como hardware que han sido necesarios.

Los **Contenidos** 1.5 presentan una visión general del proyecto en la que se muestra la estructura del proyecto y un breve resumen de que se puede encontrar en cada capítulo.

1.1. Motivación

¿Por qué se ha elegido Android? ¿Por qué un juego? ¿Y por qué HTC? En esta sección se muestran los aspectos que han empujado a hacer esta elección.

Si se revisan estadísticas de smartphones [1], se comprueba que en diciembre de 2010 un 31 % de los móviles de Estados Unidos era un smartphone, y se espera que sean la mayoría a finales de 2011. Según Gartner [2] (líder mundial en investigaciones de tecnologías de la información), los smartphones sumaron 297 millones (19 %) al total de móviles vendidos durante 2011, cifra que representa un 72,1 % más de ventas que en 2009.

El análisis de Gartner de 2010 de ventas de smartphones (figura 1.1) muestra que Symbian dominaba el mercado, sin embargo la venta de terminales que utilizan el sistema de Android, creció casi un 900 % comparado con 2009. Si además se analizan las compras más recientes, Android supone un 43 % de las compras frente al 26 % del iPhone y el 20 % de RIM (BlackBerry).

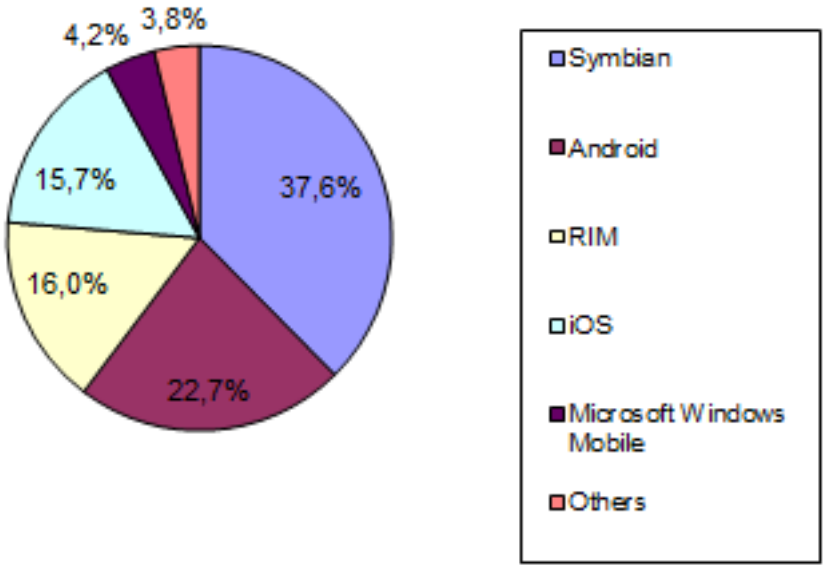


Figura 1.1. Smartphones y sistemas operativos. Estadísticas.

Analizando datos sobre los juegos en Android [3], en Junio de 2011 la cantidad disponible para los usuarios era de 34.639 juegos. En la tabla 1.1 se muestran algunos de los juegos más descargados, con su precio (en euros) y el rango de número de instalaciones.

Juego	Instalaciones	Precio
Robo Defense de Lupis Labs	500.000 - 1.000.000	2,11
X Construction de CrossConstruct	100.000 - 500.000	1
Doodle Jump de GameHouse	100.000 - 500.000	0,70
Game Dev Story de Kairosoft Co.	100.000 - 500.000	2
Order and Chaos Online de Gameloft	5.000 - 10.000	4,92
N.O.V.A. 2 HD de Gameloft	1.000 - 5.000	4,92

Tabla 1.1. Juegos en Android. Precios y número de descargas.

HTC diseñó el primer smartphone para la plataforma Android. Desde entonces ha sido uno de los principales proveedores para la casa Google.

Con todos estos datos en mente, el presente proyecto diseña y desarrolla un juego para smartphone (de la marca HTC) sobre la plataforma Android.

El elegido es una versión simplificada del juego de tablero multijugador "Los Colonos de Catán", inventado por Klaus Teuber. El objetivo es construir sobre un tablero que se modifica en cada nueva partida, poblados, ciudades o caminos. Estas construcciones proporcionan distintas puntuaciones de manera que el jugador que llega antes a los 15 puntos, gana la partida. Aunque la mecánica del juego es muy sencilla, su dinámica es bastante compleja.

Dado que el juego ha tenido mucho éxito en su versión tradicional, éste se ha recreado en forma de software como la de MSN Messenge o Linux, Windows y MAC, Play Station, Xbox 360, Nintendo DS, iPhone [4] 1.2 o iPad .

Figura 1.2. Colonos de Catán. Versión para iPhone.

1.2. Objetivos

La realización de este proyecto busca cumplir varios objetivos que se detallan a continuación.

El objetivo principal es desarrollar un juego utilizando la plataforma Android, en concreto una versión de "Los Colonos de Catán", ofreciendo una alternativa digital y portable al clásico juego de mesa.

Otro de los objetivos es el de explotar las posibilidades que ofrece Android en combinación con un smartphone, destacando el uso de pantalla táctil para facilitar el juego (en contraste con una pantalla más teclado adicional). Si se cumple este objetivo y gracias a esa base adquirida, se logrará aprender a desarrollar cualquier tipo de juegos y aplicaciones.

Un tercer objetivo es de proporcionar un punto de partida a futuros estudiantes, que deseen desarrollar juegos o aplicaciones en la misma plataforma.

Y como último objetivo, crear la base técnica del juego para que futuros desarrolladores sigan añadiendo funcionalidades al mismo 5.

1.3. Fases del desarrollo

En este apartado se detallan las distintas fases que han formado parte del desarrollo del proyecto.

- **Documentación previa**: dado que el proyecto se desarrolla en una plataforma con la que no se estaba familiarizado, es necesario documentarse sobre la plataforma Android y cómo desarrollar aplicaciones en ella.
- **Análisis de requisitos**: se realiza un análisis de requisitos para determinar características tales como las funcionalidades básicas del juego o las opciones disponibles para los jugadores.
- **Diseño**: El diseño del juego ha de incluir bocetos de las diferentes pantallas, componentes que lo forman y todo aquello que sirva como guía para la implementación del mismo.
- **Implementación**: Con el estudio realizado durante las fases de análisis y diseño, se pasa a la fase de implementación. El juego se podrá ejecutar desde el primer momento, de modo que a medida que se avance con el desarrollo, las funcionalidades se irán incrementando.

1.4. Medios utilizados

Para la realización del proyecto han sido necesarios tanto medios hardware como software. Los elementos hardware se han empleado tanto para la fase de desarrollo, como para la de pruebas, mientras que los recursos software se han empleado para la fase de desarrollo y la fase de elaboración de la memoria.

Elementos hardware:

- Ordenador potente para programar y simular el juego. Puesto que se implementa para un smartphone, no es necesario tener una gran tarjeta gráfica. El que se ha utilizado ha sido un Toshiba Intel Core i3 con 2,85 GB de RAM. Además

cumple con los requerimientos del sistema para desarrollar aplicaciones en Android (Windows XP).

- Smartphone HTC Legend para probar el juego durante la fase de desarrollo y pruebas. Aunque se dispone de un simulador en el entorno de desarrollo, probarlo en el dispositivo final permite depurar el juego con mucha más precisión. Tiene una pantalla táctil con una resolución de 320 X 480 HVGA.

Elementos software:

- Eclipse: en su versión Ganymede (versión 3.4 del año 2008)
- JDK: Java SE, no es suficiente con JRE
- Android SDK: para comenzar a programar con Android es necesario su instalación. Para este proyecto se ha empleado la versión android-sdk_r05-windows. No es un entorno de desarrollo completo, sólo incluye las herramientas del núcleo del SDK, que se pueden usar para descargar el resto de los componentes del SDK (como la última versión de la plataforma Android)
- ADT Plugin para Eclipse: Android ofrece un plugin hecho a medida para el IDE de Eclipse llamado ADT (Android Development Tools), y está diseñado para proporcionar un entorno potente e integrado en el que desarrollar aplicaciones de Android. Extiende las capacidades de Eclipse para configurar fácilmente proyectos de Android, crear interfaces de usuario o depurar las aplicaciones utilizando las herramientas del SDK de Android.
- ADV (Android Virtual Device): en el que se simulará el juego. El utilizado es Android 2.1 (API level 7) con una pantalla HVGA, para que simule a la perfección el HTC Legend
- Microsoft Windows XP Profesional y Office 2003: se han usado herramientas como Paint para hacer modificaciones en imágenes y Microsoft Project para hacer la planificación previa al proyecto y los tiempos reales finales
- StarUML: herramienta libre para el diseño de modelos UML utilizados durante la fase de diseño del juego

- Photoshop: para modificaciones más elaboradas o texturas en imágenes
- WinEdt y Latex: para escribir la presente memoria

1.5. Contenidos

La memoria se ha estructurado en varios capítulos, que profundizan en diferentes aspectos del proyecto.

El capítulo 1, **introducción**, muestra una primera visión del proyecto, que ha motivado su realización y por qué se eligió el mismo.

En el capítulo 2, **estado de la cuestión**, se habla de la historia de los smartphones, los distintos sistemas operativos, el por qué de la elección de Android, HTC y un juego y para finalizar una breve historia de los juegos en Android.

El capítulo 3, **análisis, diseño e implementación**, es el núcleo del proyecto: se analiza la arquitectura y el modelo del juego, se diseña y se detalla el proceso de implementación.

En el capitulo 4 de **conclusiones**, se recogen los resultados obtenidos y se reflexiona sobre la consecución de los objetivos marcados al comienzo del proyecto.

El último capítulo, **líneas futuras** se marcan posibles desarrollos que pueden hacer evolucionar el juego inicial.

También se han incluido capítulos con un **glosario** de términos, **anexos** en los que se detallan la planificación y presupuesto, y el sistema de coordenadas del tablero, y una **bibliografía**.

Capítulo 2

Estado de la cuestión

En este primer capítulo se contextualiza el uso de juegos en móviles: se presenta una breve historia del origen y evolución de los smartphones, qué sistemas operativos utilizan, cómo se descargan aplicaciones y juegos y algunos de los más importantes para Android. También se muestra en más detalle el sistema operativo Android y como es su uso en terminales HTC. Para finalizar se detallan algunas de las características del terminal HTC Legend.

2.1. Historia de los smartphones

El origen [5] de los smartphones data del año 1993. Aquellos primeros modelos se usaban como teléfonos de empresa y sus precios eran prohibitivos para la mayoría de los consumidores. Pero gracias al enorme éxito del iPhone, las operadoras han visto que pueden comprometer a sus clientes a tener una permanencia si se subvencionan las compras de los últimos modelos de smartphones.

El Simon (figura 2.1) fue el primer intento real de crear un teléfono que incorporase voz y datos en el mismo dispositivo (actuaba como teléfono móvil , PDA e incluso fax). También disponía de una pantalla táctil que permitía marcar números de teléfono, siendo así el primer precursor del iPhone que nacería 14 años después. Su precio original era de 899 dólares.

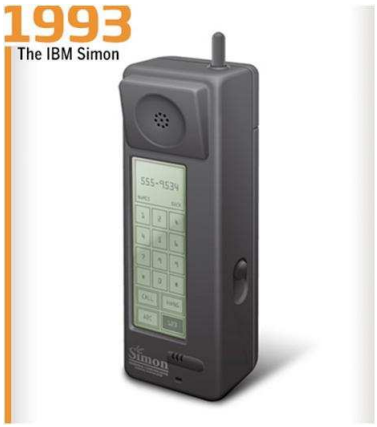


Figura 2.1. El Simon - Primer smartphone

El Nokia 9110 Communicator (figura 2.2) todavía tenía una gran pantalla gris y no se podía navegar realmente por la web, pero sí que tenía un diseño muy creativo que ha servido como modelo para smartphones como el Motorola Droid.

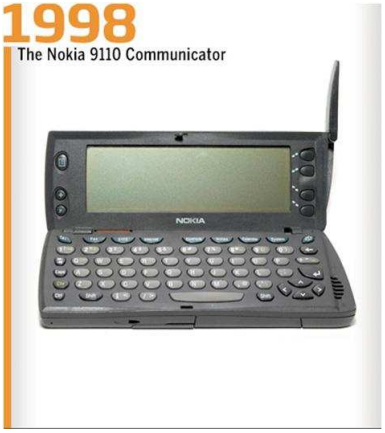


Figura 2.2. Nokia 9110 Communicator. Primeros diseños.

A finales de los noventa, la compañía canadiense Research in Motion sacó al mercado localizadores que usaron millones de personas por todo el mundo. Y fue en 2002 cuando RIM entró en el mercado de la telefonía móvil con su dispositivo BlackBerry 5810 (figura 2.3) con e-mail y capacidad de navegar por la Web. Requería de cascos para poder hablar. Hasta el año 2004, RIM no saco al mercado un dispositivo que no necesitaba cascos (BlackBerry 6210).

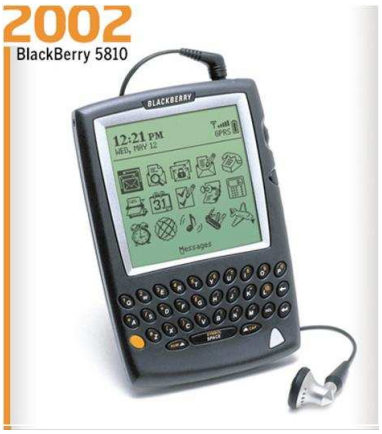


Figura 2.3. BlackBerry 5810. .

El Treo 600 (figura 2.4) fue el primer smartphone lanzado por Palm. Este dispositivo que tenía modelos GSM y CDMA, poseía además una capacidad de 32MB de RAM y procesador de 144MHz.

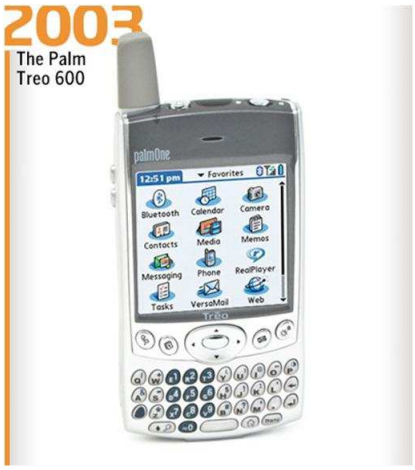


Figura 2.4. Palm Treo 600. .

El iPhone (figura 2.5) fue el primer intento de smartphone de Apple integraba pantalla táctil y la mejor experiencia para navegar por la red ofrecida hasta entonces. A día de hoy, todavía es comparado con el resto de smartphones.

Figura 2.5. iPhone. .

Se ha de destacar el crecimiento de Android (figura 2.6), sistema operativo para móviles que fue lanzado en el otoño de 2007. El éxito del sistema operativo de código abierto, es más impresionante si se tiene en cuenta la cantidad de OS muy poderosos que ya estaban en el mercado (iPhone, BlackBerry, Windows Mobile o Symbian). Ahora el OS de Google se ha convertido en el mayor proveedor dentro de la industria de los smartphones: a finales de 2009, más de un 7 % de los dispositivos en todo USA. Durante una serie de encuestas realizadas en el primer trimestre de 2011, el 31 % de los usuarios que pensaban comprar un smartphone indicaron que preferían el OS de Android en comparación con el 30 % que prefería iPhone.

Figura 2.6. Android. .

El G1 (figura 2.7) de la operadora T-Mobile fue el primer terminal Android [6] y salió al mercado en septiembre de 2008. Fue fabricado y diseñado por HTC con la supervisión

de Google. Este primer terminal tenía un teclado físico además de la pantalla táctil como métodos de control, y se comenzó a comercializar de manera online a través de la web de T-mobile. En su versión para desarrolladores, el G1 fue el responsable de la revolución de Android, ya que los programadores hicieron las primeras aplicaciones para él.

Figura 2.7. G1 de T-mobile - Primer smartphone bajo Android

Aunque Android llegó al mercado un año antes que el lanzamiento del Droid (figura 2.8), éste fue el primer gran hito para la plataforma Android que le permitió su reconocimiento como marca. Este dispositivo, que fue el primer smartphone basado en Android que funcionó por la red de Verizon, vendió más de un millón de unidades durante sus primeros 74 días en el mercado.

Figura 2.8. Motorola Droid

Con el lanzamiento del HTC EVO 4G de Sprint (figura 2.9), han conseguido que sus usuarios puedan sacar provecho de su red de alta velocidad (tienen la red wifi comercial más rápida de los Estados Unidos). Además de su conectividad, el dispositivo tiene un

gran tamaño (800x400 px) y un peso de 170 gr.

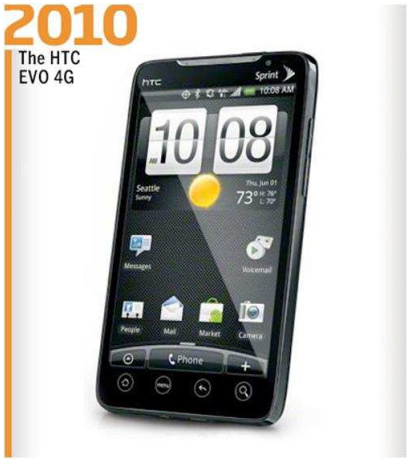


Figura 2.9. HTC Evo 4G. .

2.2. Sistemas operativos para los smartphones

El sistema operativo del smartphone es el que gestiona sus recursos hardware y software. En la actualidad, 7 grandes compañías y sus sistemas operativos, abarcan más del 99 % de los dispositivos. Estos 7 sistemas operativos son: Symbian, Research In Motion (RIM), iPhone OS, Microsoft Windows Mobile, Linux, Android y WebOS.

1. **Symbian**: fue producto de la alianza de varias compañías entre las que se encuentran Nokia, Sony Ericsson, Psion, Samsung, Siemens, Arima, Benq, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic o Sharp. La plataforma de Symbian se diseñó con los smartphones en mente y la mayoría de los terminales son de Nokia [7].

2. **Research In Motion**: compañía de telecomunicaciones y dispositivos inalámbricos canadiense. Es conocida por ser la desarrolladora de la BlackBerry. Su sistema operativo (BlackBerry OS) es propietario y proporciona multitarea. Soporta dispositivos de entrada como pantalla táctil o trackball.

3. **iPhone OS**: es el sistema operativo para móviles de Apple. Se desarrolló originariamente para el iPhone, pero en la actualidad se ha extendido para su uso en otros dispositivos de Apple como iPod touch, iPad y Apple TV.

4. **Microsoft Windows Mobile**: sistema operativo desarrollado por Microsoft que se usa tanto en smartphones como en dispositivos móviles. La última versión (Windows Phone 7) establece los requisitos mínimos para pantallas táctiles de alta resolución.

5. **Linux**: LiMo foundation [8] es un consorcio de la industria cuya finalidad es la de crear un sistema operativo para dispositivos móviles, basado en Linux. Participan en el proyecto empresas como NEC, Panasonic, Vodafone, Telefonica, Orange, Mozilla, Huawei o McAfee, entre otras. Las tecnologías clave de LiMo incluyen una potente y flexible interfaz de usuario, librerías extendidas para widgets, efectos 3D para widgets, multimedia avanzado, redes sociales y localización.

6. **Android**: sistema operativo basado en Linux que utiliza el lenguaje de programación Java. Se diseñó para smartphones, pero actualmente soporta otros dispositivos como tablets o netbooks.

7. **WebOS**: sistema operativo multitarea basado en también en Linux, diseñado para dispositivos de pantalla táctil. Fue desarrollado por Palm Inc, pero actualmente es propiedad de Hewlett-Packard Company.

Según los datos de Gartner [2], Symbian lideraba el mercado de los sistemas operativos en 2009. La presión de sus competidores como RIM o Apple, y la reducción de ventas de los terminales de Nokia han impactado negativamente en la cuota de mercado de Symbian.

Las dos mejores actuaciones en 2009 fueron Android y Apple (figura 2.10). Android incrementó su cuota de mercado 3,9 puntos en 2009, mientras que Apple creció 6,2 puntos desde 2008, desplazando así del tercer puesto a Microsoft Windows Mobile.

En Enero de 2011 (figura 2.11), Canalys (una compañía de desarrollo de mercado), publicó las cifras para el Q4 del mercado de los smartphones: Android con 33,3 millones, sobrepasó a Symbian (con 31 millones) y se convirtió en la plataforma más vendida a nivel mundial para smartphones.

Company	2009 Units	2009 Market Share (%)	2008 Units	2008 Market Share (%)
Symbian	80,878.6	46.9	72,933.5	52.4
Research In Motion	34,346.6	19.9	23,149.0	16.6
iPhone OS	24,889.8	14.4	11,417.5	8.2
Microsoft Windows Mobile	15,027.6	8.7	16,498.1	11.8
Linux	8,126.5	4.7	10,622.4	7.6
Android	6,798.4	3.9	640.5	0.5
WebOS	1,193.2	0.7	NA	NA
Other OSs	1,112.4	0.6	4,026.9	2.9
Total	**172,373.1**	**100.0**	**139,287.9**	**100.0**

Figura 2.10. Sistemas Operativos en Smartphones. Ventas mundiales de smartphones por sistema operativo en 2009.

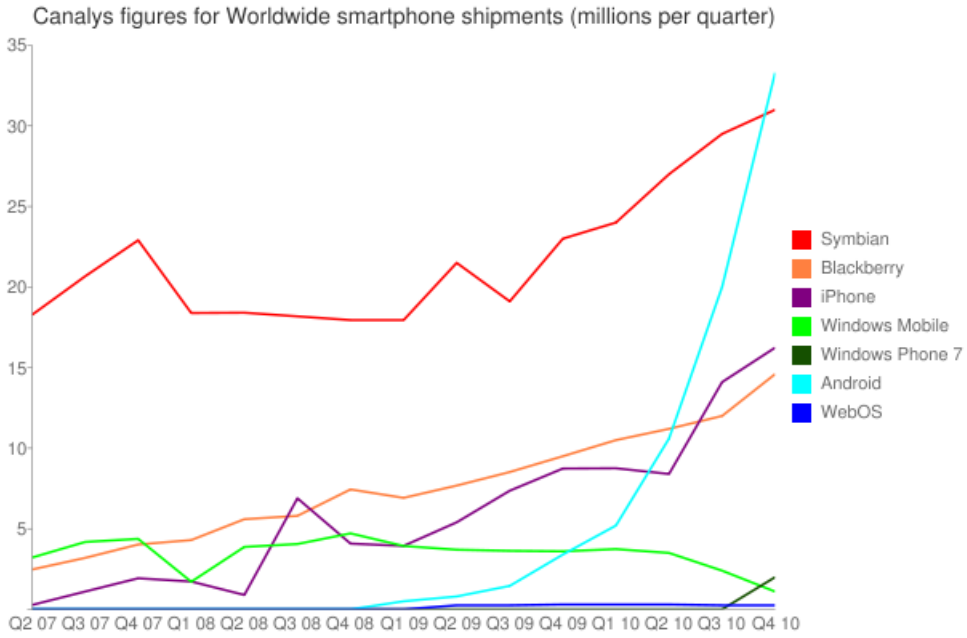


Figura 2.11. Sistemas Operativos en Smartphones. Crecimiento mundial de smartphones por sistema operativo en 2010.

Comparando distintas características de los diferentes sistemas operativos [9], se puede llegar a una conclusión de cual es el más conveniente.

Teniendo en cuenta la estética (diseño de la interfaz de usuario), Windows Phone 7 de Microsoft, sacó al mercado algo realmente original y con buena apariencia. La facilidad de uso destaca en el sistema operativo de Apple seguido por Android.

Con respecto a las funcionalidades, no fueron el punto fuerte de Apple, pero con el paso de los años se han añadido multitasking, copiar y pegar, carpetas, etc. Android sin embargo si que tiene funcionalidades adicionales como multitasking, widgets, Wi-Fi hotspot, etc. ; convirtiéndolo en el más completo en cuanto a funcionalidades se refiere.

El performance de Symbian es uno de los mejores, gracias a lo ligero de su sistema operativo. Windows Phone 7 también tiene un gran comportamiento, mientras que Android no funciona tan bien en dispositivos de gama baja.

Las aplicaciones están lideradas por Apple aunque Android está siguiéndole los pasos.

A día de hoy y teniendo en cuenta el análisis, Android ofrece la mejor combinación de funcionalidades, comportamiento y soporte para la comunidad de desarrolladores en términos de aplicaciones, dato considerado a la hora de desarrollar el juego en la plataforma.

2.3. Aplicaciones y juegos en los smartphones

Las aplicaciones se han convertido en un filtro para nuestro mundo: a través de ellas podemos acceder a información, juegos o mundos virtuales. Los smartphones están desbancando a los teléfonos móviles tradicionales, entre otras cosas por el amplio abanico de aplicaciones que nos pueden llegar a proporcionar.

Las distintas aplicaciones se pueden descargar desde diferentes stores de aplicaciones, dependiendo del sistema operativo que utilice el smartphone. Las aplicaciones se pueden agrupar en distintos tipos, siendo el juego lo estudiado para este proyecto.

2.3.1. Stores de aplicaciones

Actualmente hay diferentes grandes stores donde poder conseguir estas aplicaciones:

- App Store: store para aplicaciones para el iPhone de Apple. Su lanzamiento data del 10 de julio de 2008 y en julio de 2011 contaba con más 425.000 aplicaciones.

- Android Market: store para Android de Google. Se lanzó el 22 de octubre de 2008 y con fecha de julio de 2011, el número total de aplicaciones es de 250.000.

- App World: store de BlackBerry que salió al mercado el 1 de abril de 2009. Tiene más de 40.000 aplicaciones diponibles.

- App Catalog de Palm/HP, nació el 6 de junio de 2009 con alrededor de 4.000 aplicaciones.

- Windows Marketplace para Windows Mobile, se lanzó el 5 de octubre de 2009 y contiene unas 1.000 aplicaciones.

Entre estas stores, hay más de 720.000 aplicaciones disponibles. Más de 4 billones se han descargado de la App Store (Google no proporciona ese número).

2.3.2. Tipos de aplicaciones

Dos grandes grupos engloban los distintos tipos de aplicaciones y son los juegos y las aplicaciones propiamente dichas. Dentro de ellos, diferentes categorías definen las posibilidades que nos pueden ofrecer:

- Juegos: arcade y acción, carreras, casuales, deportes, cartas y casino, puzzles, etc.

- Aplicaciones: compras, comunicación, educación, finanzas, medicina, productividad, etc.

Hay que tener en cuenta que mientras que una aplicación de logística o inventario esta orientada a un público puntual, un juego se dirige al público masivo.

2.3.3. Juegos para Android

En este apartado se detallan algunos de los mejores juegos de Android [10], que sirven como inspiración para el desarrollo de nuevos juegos.

1. **Angry Birds**: juego en el que se lanzan pájaros con un tirachinas. Se irá avanzando de nivel a medida que se golpee con los pájaros a cerdos que han robado sus huevos(figura 2.12).

2. **World War**: juego de acción en el que se construye un ejército a base de completar misiones, conseguir dinero, subir de nivel, comprar tropas, construir edificios o crear alianzas.

3. **Replica Island**: 40 niveles en los que la finalidad es descubrir una fuente secreta de poder en una isla misteriosa.

4. **Gem Miner**: el jugador ha de cavar una mina para encontrar metales, piedras y gemas para hacer una fortuna. Tiene muchas pantallas, y el jugador tiene que mejorar su modo de cavar y comprar mejores herramientas y mapas.

5. **Air Control**: en este juego se ha de pilotar un avión de un modo correcto, y hacerlo aterrizar en la pista.

6. **Bonsai Blast**: puzzle en el que hay que hacer combinaciones de esferas de tres colores lanzándolas desde un punto determinado de la pantalla, intentando evitar que la cadena de esferas llegue al agujero negro (figura 2.13).

7. **Abduction**: la finalidad es la de rescatar a animales que han sido secuestrados por extraterrestres. El jugador que controla a una vaca, tiene que darles caza.

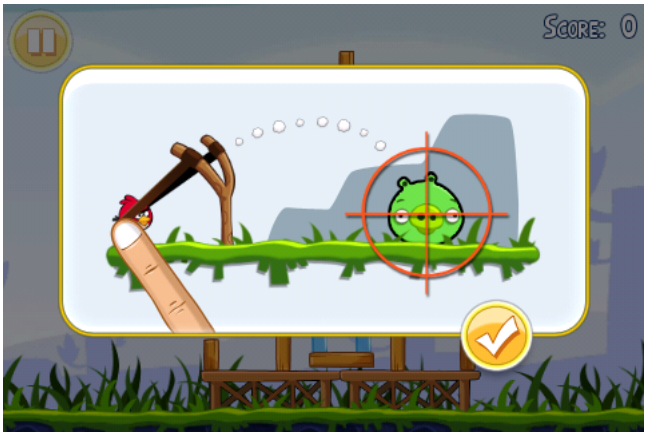


Figura 2.12. Angry Birds. Juego para Android.

Figura 2.13. Bonsai Blast. Juego para Android.

2.4. Historia de Android

[11]En Junio de 2005 Google compró una pequeña empresa que desarrollaba aplicaciones para móviles llamada Android Inc. Andy Rubin, cofundador de la compañía, pasó a ser el director de la división de plataformas móviles de Google. en ese momento se empezó a especular con que Google tenía la idea de lanzar el "Google phone": el dispositivo y el coste del servicio serían sufragados con publicidad en el mismo.

Aparecieron distintos prototipos y demos no oficiales, y fue en Noviembre de 2007 cuando se anunció la creación de la Open Handset Alliance [12] (organización para la difusión de la plataforma móvil de Android). Fabricantes de dispositivos y proveedores de servicios se unieron para crear el primer sistema operativo abierto para móviles y que además no estaría atado a un terminal concreto o marca (gracias a su kernel de Linux se podría adaptar a casi cualquier dispositivo). Cinco días después de este anuncio, Google sacó su Software Development Kit o SDK que además incluía un emulador.

El primer móvil con Android fue el G1 T- Mobile (HTC Dream) lanzado en septiembre de 2008. Desde este lanzamiento, numerosos fabricantes como Sony Ericsson, LG, Motorola, Samsung o Lenovo han utilizado el sistema operativo de Android.

2.4.1. Qué es Android

Android [13] es un conjunto de software para dispositivos móviles que incluye sistema operativo, middleware y aplicaciones. El SDK de Android proporciona las herramientas y APIs necesarios para empezar a desarrollar aplicaciones en su plataforma utilizando el lenguaje de programación de Java.

2.4.2. Arquitectura de Android

El siguiente gráfico (figura 2.14) muestra los distintos componentes del sistema operativo de Android.

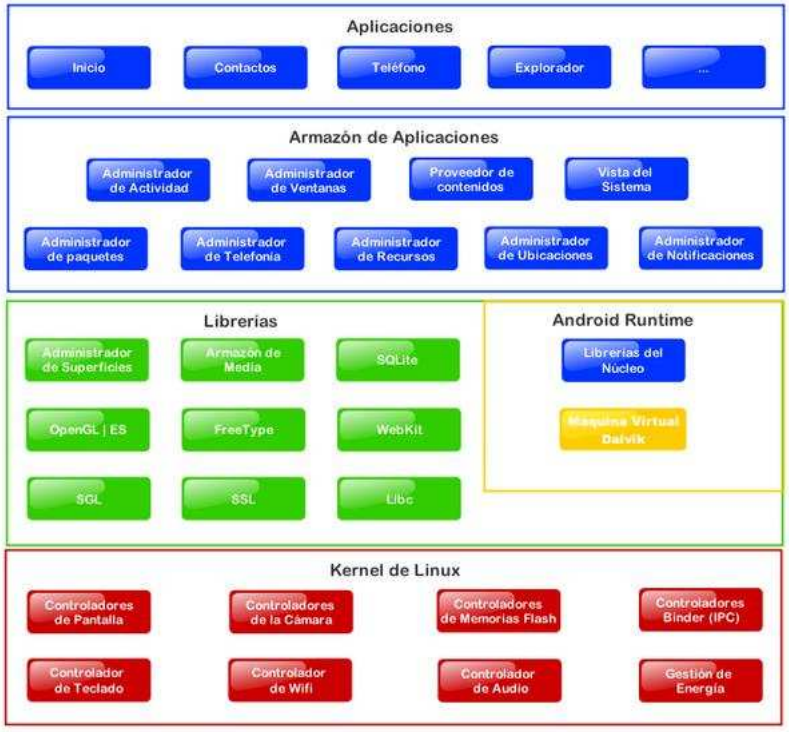


Figura 2.14. Arquitectura de Android. .

Aplicaciones

Android proporciona una serie de aplicaciones de core como cliente de e-mail, programa de SMS, calendario, mapas, explorador, contactos, etc. Todas las aplicaciones están escritas utilizando el lenguaje de programación de Java.

Framework de aplicaciones

Al proporcionar una plataforma de desarrollo abierta, Android ofrece a los desarrolladores la posibilidad de programar aplicaciones muy brillantes e innovadoras. Los desarrolladores son libres para beneficiarse del hardware del dispositivo, acceder a información de localización, correr servicios en segundo plano, activar alarmas, añadir notificaciones a la barra de estado, y mucho, mucho más.

Los desarrolladores tienen acceso completo al mismo API utilizado para las aplicaciones de core. La arquitectura de aplicación está diseñada para simplificar la reutilización de componentes; cualquier aplicacíón puede publicar sus capacidades y cualquier otra puede hacer uso de esas capacidades (sujeto a restricciones de seguridad impuestas por el framework). Este mismo mecanismo permite al usuario reemplazar componentes.

Todas las aplicaciones son un conjunto de servicios y sistemas, incluyendo:

- Un conjunto de **Views** que se usan para desarrollar aplicaciones, incluyendo listas, cuadrículas, cuadros de texto, botones e incluso explorador web embebido
- **Proveedores de contenido** que permiten a las aplicaciones acceder a datos de otras aplicaciones (por ejemplo de los contactos), o compartir sus propios datos.
- Un **gestor de recursos**, proporcionando acceso a recursos como strings, gráficos, y ficheros de layout
- Un **gestor de notificaciones** que permite a todas las aplicaciones mostrar alertas personalizadas en la barra de estado
- Un **gestor de actividad** que gestiona el ciclo de vida de las aplicaciones y proporciona una navegación común

Librerías

Android incluye un conjunto de librerías de C/C++ que usan varios componentes del sistema Android.

Algunas de estas librerías:

- **System C**
- **Media**
- **Surface Manager**
- **LibWebCore**
- **SGL**
- **librerías 3D**
- **FreeType**
- **SQLite**

Android Runtime

Android incluye un conjunto de librerías de core que proporcionan la mayoría de las funciones disponibles de las librerías de core del lenguaje de programacíon de Java.

Cada aplicación de Android corre sobre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik se implementó de forma que un dispositivo puede correr varias máquinas virtuales eficientemente. La Dalvik VM ejecuta ficheros en el formato .dex que está optimizado para un uso mínimo de memoria.

Kernel de Linux

Android utiliza la versión 2.6 para servicios de sistema del core como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de driver. El kernel actúa también como una capa abstracta entre el hardware y el resto de la pila del software.

2.4.3. Fundamentos de las aplicaciones

Las aplicaciones [14] de Android están escritas en lenguaje de programación Java. Las herramientas del SDK de Android compilan el código (junto con otros ficheros de datos y

recursos) en un *paquete de Android*, y genera un fichero con una extensión .apk. Todo el código en un único .apk, se considera como la aplicación y es el fichero que los dispositivos basados en Android utilizan para instalar la aplicación.

Una vez instalado en el dispositivo, cada aplicación de Android tiene su propia zona de seguridad:

- El sistema operativo de Android es un sistema de Linux multiusuario en el que cada aplicación es un usuario diferente.
- Por defecto, el sistema asigna a cada aplicación un identificativo de usuario de Linux único (el identificativo es utilizado solo por el sistema y desconocido por la aplicación). El sistema pone permisos a todos los archivos de una aplicación de manera que solo el ID de usuario asignado a la aplicación puede acceder a él.
- Cada proceso tiene su propia VM, así que un código de aplicación corre de manera aislada a otras aplicaciones.
- Por defecto, cada aplicacíón corre en su propio proceso de Linux. Android comienza el proceso cuando cualquiera de los componentes de la aplicación necesitan ser ejecutados, después finaliza el proceso cuando ya no es necesario o cuando el sistema debe recuperar memoria para otras aplicaciones.

De esta forma, el sistema de Android implementa el *principio del menor privilegio*. Esto es, cada aplicación por defecto, tiene acceso solo a los componentes que requiere para hacer su trabajo y no más. Esto crea un medio muy seguro en el que una aplicación no puede acceder a partes del sistema para las que no tenga permisos.

Los componentes de las aplicaciones son los bloques esenciales para construir una aplicación de Android. Cada componente es un punto diferente a través del cual el sistema puede entrar en la aplicación. No todos los componentes son puntos reales de entrada para el usuario y algunos dependen de otros, pero cada uno existe como una entidad en si mismo y juega un rol específico.

Hay cuatro tipos de componentes de aplicaciones:

- **Actividades**: una actividad representa una única pantalla con una interfaz de usuario. Una actividad se implementa como una subclase de Activity.
- **Servicios**: no se corresponde con una interfaz de usuario, es un componente que corre de fondo sin bloquear otros procesos. Implementa una subclase de Service.
- **Proveedores de contenido**: gestiona un conjunto de datos compartidos entre diferentes aplicaciones. Se implementa como una subclase de ContentProvider.
- **Broadcast receivers**: es un componente que responde a anuncios de broadcast del sistema. Se implementa como subclase de BroadcastReceiver.

Las actividades, los servicios y los broadcast receivers se activan por medio de un mensaje asíncrono llamado *intent*.

El archivo del manifiesto (AndroidManifest.xml) muestra todos los componentes que tiene definidos la aplicación. Además de declarar los componentes, identifica los permisos de usuario que requiere la aplicación, declara el nivel de API mínimo, los usos de hardware y software que se necesitan, etc.

2.5. HTC y Android

HTC (High Tech Computer Corporation) es una compañía taiwanesa nacida en 1997 [15] [16], que con el paso de los años ha pasado a ser la fábrica número uno de terminales de Google basados en Android. Gracias al apoyo y respaldo de Google, HTC se ha convertido en una compañía capaz de competir con otras como Samsung o LG. Empezaron diseñando unos de los primeros dispositivos táctiles y wireless del mercado, hasta que en el año 2000 lanzaron el iPAQ (PDA de Compaq figura 2.15) uno de los productos de moda.

El CEO de la compañía prefirió enfocar sus desarrollos en la industria de los móviles que podría dar una mayor cuota de mercado. En este momento comenzaron negociaciones con empresas de telefonía móvil en europa para ofrecer la fabricación terminales: en el año 2002 sacaron al mercado dos teléfonos móviles para O2 en Reino Unido y para Orange en Francia. El XDA (figura 2.16), híbrido entre PDA y teléfono móvil, fue comercializado

Figura 2.15. Compaq iPAQ 3630. Primeros dispositivos de HTC.

bajo distintos fabricantes y fue uno de los primeros con pantalla táctil. Pero no fue hasta la utilización de Android, cuando HTC pasó a ser una gran y poderosa compañía de fabricación de móviles.

Figura 2.16. XDA. Primeros dispositivos de HTC.

El CEO de HTC Peter Chou, empezó a relacionarse con Andy Rubin (creador de Android), hecho que permitió que cuando en 2005 Google comprara Android, después de agregar algunas mejoras, HTC pasó a ser el mejor socio para proveer el hardware: el SO de Google necesitaba un dispositivo bastante sofisticado, y HTC ya sabía como hacerlo. Durante los tres años siguientes al lanzamiento del primer terminal con Android, HTC y Google hicieron grandes esfuerzos para que el nuevo sistema operativo funcionase bien es los dispositivos móviles.

Fue en 2005 cuando utilizaron por primera vez características de pantalla táctil y teclado y además añadieron capacidades de GPS. Durante los tres años siguientes al lanzamiento del primer terminal con Android, HTC y Google hicieron grandes esfuerzos para que el nuevo sistema operativo funcionase bien es los dispositivos móviles.

En 2007 se aprovecharon de los gestos realizados con los dedos para definir diferentes acciones (TouchFLO) y para conseguir una navegación web de mayor calidad. En 2009 incluyeron el HTC Sense, que incluye widgets para agregar nuevas funciones, además de unir contactos desde distintas fuentes, y definir varios perfiles como personal o profesional. También se lanzó el primer móvil 4G con Android (HTC Evo).

2.6. HTC Legend

El HTC Legend[17] (lanzado en Marzo de 2010) se creó con una sola pieza de aluminio pulido para hacerlo robusto y resistente, con dimensiones de 112 x 56,3 x 11,5 mm. El trackball de diseños anteriores se sustituye por un trackpad óptico. Su peso es de 126 gramos con batería y la pantalla es táctil con resolución de 320x480 HVGA y tamaño de 81,2 mm con gran brillo y nitidez.

Figura 2.17. HTC Legend. Smartphone utilizado.

La velocidad de procesamiento de la CPU es de 600 MHz, la plataforma Android 2.1

(Eclair) con HTC Sense y el almacenamiento incluye RAM de 384 MB y ROM de 512 MB (ampliable hasta 32 GB gracias a la admisión de tarjetas MicroSD 2.0).

Dispone de cuatro botones para los controles principales Home, Menú, Volver y Buscar.

Todas estas características hacen del HTC Legend un terminal idóneo para desarrollar el juego y conseguir los objetivos propuestos.

Capítulo 3

Análisis, diseño e implementación

En este tercer capítulo del Proyecto Fin de Carrera se incluye un descripción del problema y un estudio de las alternativas. Es el más importante ya que describe todo el proceso, desde la primera idea hasta su puesta en marcha en el dispositivo del usuario.

Este capítulo se divide en tres partes. La primera de ellas define el análisis, cuáles fueron los primeros pasos del proyecto. La segunda de parte, muestra el diseño y las características del juego que sirven de base para la implementación. La tercera parte muestra la implementación en la que se detallan algunas partes remarcables del código para mejor entendimiento del mismo.

3.1. Análisis

3.1.1. Propuesta inicial

La primera idea planteada para la realización de este proyecto, es la de implementar un juego de mesa pero para una plataforma móvil. El juego elegido es una versión de "Los Colonos de Catán", juego de tablero en el que varios jugadores luchan por tener el mayor número de construcciones. Esto dependerá de la estrategia que sigan a la hora de situar sus recursos para así obtener más materiales y poder seguir construyendo.

El tablero, formado por celdas con materiales o recursos, y números que representan los posibles valores obtenidos con los dados, ha de ser aleatorio en cada partida. De esta forma no es posible repetir estrategias o jugadas.

Al comienzo del juego se dispone de un menú en el que el jugador puede elegir entre comenzar un juego nuevo, leer las instrucciones del mismo, o leer la información relacionada con la autora.

Se debe tener la posibilidad de elegir el número de jugadores con un mínimo de dos, y un máximo de tres (por las limitaciones visuales que implican el uso de la plataforma móvil).

En cada turno del jugador, éste debe lanzar los dados (durante el lanzamiento se emite un sonido) y el sistema muestra el valor obtenido en los dados. Si ese valor corresponde a alguna de las celdas en las que hay poblados o ciudades de algún jugador, el sistema actualiza los materiales disponibles por el jugador/jugadores que tienen esas construcciones.

Si un jugador quiere construir, el sistema verifica si tiene los materiales necesarios, en caso negativo muestra un mensaje de error indicando que no se puede construir, y en caso afirmativo, el jugador podrá poner la construcción elegida en alguno de los lugares habilitados. Cada vez que se construye, se actualizan los puntos del jugador en cuestión, de forma que si éste alcanza 15 puntos, se muestra un mensaje con un sonido en el que se indica que ha ganado el juego.

Si un jugador quiere construir, no tiene los materiales necesarios, pero si cuenta sin embargo con materiales extra, puede comerciar con la banca: cuatro recursos del jugador se pueden cambiar por un recurso de la banca. Gracias a esto, un jugador puede suplir la falta de materiales debida a una mala jugada o a la mala suerte en los dados.

Cuando un jugador saque un 7 en los dados, tendrá la opción de mover el ladrón. La celda en la que le ladrón esté situado, no producirá recursos.

Existe la opción de pasar al siguiente jugador, una vez otro ha finalizado su turno.

La implementación del juego se hace sobre la plataforma Android, programando con librerías basadas en Java. Esto facilita que se haga un análisis orientado a objetos.

3.1.2. Requisitos de usuario

Una vez analizados los objetivos e idea inicial, se realiza la extracción de los requisitos de usuario.

Con los requisitos de usuario, el mismo indica una serie de restricciones o funcionalidades que debe cumplir el programa, juego o aplicación. Los tipos de requisitos de usuario son de capacidades (requeridas por los usuarios para resolver un problema o determinar un objetivo) y de restricciones (impuestas por los usuarios sobre la forma como debe ser resuelto el problema o logrado el objetivo).

Los requisitos de usuario están definidos por los siguientes atributos:

- **Identificador**: muestra el tipo de atributo RUC (Requisito de Usuario de Capacidad) o RUR (Requisito de Usuario de Restricción), seguido del número del requisito. Este identificativo es único.
- **Necesidad**: indica la prioridad del requisito que puede ser *Esencial*, *Deseable* u *Opcional*. Para el análisis del juego, no se han tenido en cuenta los requisitos opcionales.
- **Título**: frase con la que se da a conocer el nombre del requisito.
- **Comentario**: pequeña descripción del requisito.

A continuación se muestran los requisitos de usuario de capacidad:

Identificador:	RUC-01	Necesidad:	Esencial
Título:	Lanzamiento del juego		
Descripción:	El juego se lanzará a través de un ejecutable instalado en el móvil		

Tabla 3.1. RUC-01. Lanzamiento del juego.

Identificador:	RUC-02	Necesidad:	Esencial
Título:	Multijugador		
Descripción:	El juego debe permitir que varios jugadores participen		

Tabla 3.2. RUC-02. Multijugador.

Identificador:	RUC-03	Necesidad:	Esencial
Título:	Tablero aleatorio		
Descripción:	El tablero se debe generar de forma aleatoria (materiales y números distribuidos de manera distinta en cada partida).		

Tabla 3.3. RUC-03. Tablero aleatorio.

Identificador:	RUC-04	Necesidad:	Esencial
Título:	Instrucciones		
Descripción:	Se deben mostrar las instrucciones del juego.		

Tabla 3.4. RUC-04. Instrucciones.

Identificador:	RUC-05	Necesidad:	Esencial
Título:	Ladrón		
Descripción:	Se tiene que poder mover el ladrón cuando se saque un 7.		

Tabla 3.5. RUC-05. Ladrón.

Identificador:	RUC-06	Necesidad:	Deseable
Título:	Mensajes de información		
Descripción:	Se deben mostrar mensajes de ayuda/información (cuando no se tienen los materiales suficientes, cuando hay que cambiar el ladrón, cuando se gana el juego, cuando se inicia el juego y se informa de lo que hay que construir).		

Tabla 3.6. RUC-06. Mensajes de información.

Identificador:	RUC-07	Necesidad:	Esencial
Título:	Turno de jugador		
Descripción:	Se debe tener una vista del turno del jugador que corresponda.		

Tabla 3.7. RUC-07. Turno de jugador.

Identificador:	RUC-08	Necesidad:	Esencial
Título:	Puntuación		
Descripción:	Se debe tener una vista de los puntos acumulados por cada jugador.		

Tabla 3.8. RUC-08. Puntuación.

Identificador:	RUC-09	Necesidad:	Esencial
Título:	Cartas		
Descripción:	Se debe tener una vista de las cartas de las que dispone cada jugador.		

Tabla 3.9. RUC-09. Cartas.

Identificador:	RUC-10	Necesidad:	Esencial
Título:	Botón dados		
Descripción:	Se debe tener una vista de un botón para lanzar los dados		

Tabla 3.10. RUC-10. Botón dados.

Identificador:	RUC-11	Necesidad:	Esencial
Título:	Dados		
Descripción:	Se debe tener una vista del valor obtenido tras lanzar los dados.		

Tabla 3.11. RUC-11. Dados.

Identificador:	RUC-12	Necesidad:	Esencial
Título:	Construir		
Descripción:	Se debe tener un menú para construir o poblado, ciudad o camino. Se debe indicar que materiales son necesarios para cada construcción.		

Tabla 3.12. RUC-12. Construir.

Identificador:	RUC-13	Necesidad:	Esencial
Título:	Comerciar		
Descripción:	Se debe tener un menú para comerciar en el que se indique qué 4 materiales que ya tiene el jugador se quieren cambiar, y por cuál.		

Tabla 3.13. RUC-13. Comerciar.

Identificador:	RUC-14	Necesidad:	Esencial
Título:	Siguiente jugador		
Descripción:	Se debe tener una opción en el menú para pasar al siguiente jugador.		

Tabla 3.14. RUC-14. Siguiente jugador.

Identificador:	RUC-15	Necesidad:	Deseable
Título:	Sonido ganador		
Descripción:	Se debe tener un sonido cuando un jugador gana la partida.		

Tabla 3.15. RUC-15. Sonido ganador.

Identificador:	RUC-16	Necesidad:	Deseable
Título:	Sonido dados		
Descripción:	Se debe tener un sonido cuando se lanzan los dados.		

Tabla 3.16. RUC-16. Sonido dados.

Identificador:	RUC-17	Necesidad:	Deseable
Título:	Sonido error		
Descripción:	Se debe tener un sonido de error cuando no es posible construir o comerciar.		

Tabla 3.17. RUC-17. Sonido error.

Identificador:	RUC-18	Necesidad:	Deseable
Título:	Icono en ejecutable		
Descripción:	Para lanzar el juego, éste llevará asociado un icono.		

Tabla 3.18. RUC-18. Icono ejecutable.

A continuación se muestran los requisitos de usuario de restricción:

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

Tabla 3.19. RUR-01. Plataforma Android.

Identificador:	RUR-02	Necesidad:	Esencial
Título:	Dispositivo HTC Legend		
Descripción:	El juego deberá funcionar en el smartphone HTC Legend		

Tabla 3.20. RUR-02. Dispositivo HTC Legend.

Identificador:	RUR-03	Necesidad:	Deseable
Título:	Sonidos		
Descripción:	El jugador podrá subir/bajar el volumen de los sonidos del juego, con los botones del smartphone		

Tabla 3.21. RUR-03. Sonidos.

Identificador:	RUR-04	Necesidad:	Esencial
Título:	Formatos		
Descripción:	El formato para los ficheros de audio es .MP3 y para los ficheros de imagen .PNG		

Tabla 3.22. RUR-04. Formatos.

Identificador:	RUR-05	Necesidad:	Esencial
Título:	Posición de las construcciones		
Descripción:	Las construcciones de poblados y ciudades han de ir en los vértices de las celdas, y las de caminos en los lados de las celdas.		

Tabla 3.23. RUR-05. Posición de las construcciones.

3.1.3. Requisitos software

Los requisitos software indican lo que debe hacer el sistema (requisitos funcionales) y la forma en que se debe llevar a cabo (requisitos de rendimiento, interfaz, opera-

ción, recursos, comprobación, documentación, seguridad, calidad, mantenimiento, daño y aceptación de pruebas).

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

- **Identificador**: muestra el tipo de atributo RS (Requisito Software), más una letra adicional que indica si es funcional (F), de rendimiento (R), de interfaz (I) o de recursos (Re), seguido del número del requisito. Este identificativo es único.
- **Necesidad**: indica la prioridad del requisito que puede ser *Esencial*, *Deseable* u *Opcional*. Para el análisis del juego, no se han tenido en cuenta los requisitos opcionales.
- **Título**: frase con la que se da a conocer el nombre del requisito.
- **Fuente**: indica los requisitos de usuario de los que proviene este requisito software. Si el requisito aparece por consideración del analista, se identifica con la palabra ANA.
- **Descripción**: pequeña descripción del requisito.

Para el análisis del juego, no se han definido todos los tipos de requisitos software. A continuación se muestran los requisitos software funcionales:

Identificador:	RSF-01	Necesidad:	Esencial
Título:	Lanzar el juego		
Fuente:	RUC-01		
Descripción:	el juego se lanzará mediante la ejecución de un fichero .apk que se genera al compilar		

Tabla 3.24. RSF-01. Lanzar el juego.

Identificador:	RSF-02	Necesidad:	Esencial
Título:	Mostrar menú principal		
Fuente:	RUC-01		
Descripción:	cuando se lanza el juego, se debe mostrar una pantalla con las distintas opciones del menú: Nuevo juego, Instrucciones y Acerca de		

Tabla 3.25. RSF-02. Mostrar menú principal.

Identificador:	RSF-03	Necesidad:	Esencial
Título:	Elección de número de jugadores		
Fuente:	RUC-02		
Descripción:	al pulsar nuevo juego, se podrá elegir entre dos y tres jugadores		

Tabla 3.26. RSF-03. Elección de número de jugadores.

Identificador:	RSF-04	Necesidad:	Esencial
Título:	Pantalla principal		
Fuente:	RUC-07, RUC-08, RUC-09, RUC-10, RUC-11		
Descripción:	la pantalla principal del juego mostrará el tablero, el turno del jugador, los materiales acumulados, los puntos acumulados, un botón para lanzar los dados, y la puntuación de los dados una vez de pulse el botón correspondiente		

Tabla 3.27. RSF-04. Pantalla principal.

Identificador:	RSF-05	Necesidad:	Esencial
Título:	Mostrar menú del juego		
Fuente:	RUC-11, RUC-12, RUC-13		
Descripción:	al pulsar el botón menú del smartphone, se tendrá la posibilidad de construir, comerciar o pasar al siguiente jugador		

Tabla 3.28. RSF-05. Mostrar menú del juego.

Identificador:	RSF-06	Necesidad:	Esencial
Título:	Construir		
Fuente:	RUC-12		
Descripción:	al pulsar el botón construir dentro del menú del smartphone, se mostrará un cuadro de diálogo en el que el usuario podrá elegir qué tipo de construcción desea y se le indicará qué materiales y en qué cantidad son necesarios. Si tiene los materiales necesarios, se volverá a la pantalla principal y el jugador podrá construir. Si no tiene los materiales necesarios, se mostrará un mensaje de pop-up indicándoselo		

Tabla 3.29. RSF-06. Construir.

Identificador:	RSF-07	Necesidad:	Esencial
Título:	Comerciar		
Fuente:	RUC-13		
Descripción:	al pulsar el botón comerciar dentro del menú del smartphone, se mostrará un cuadro de diálogo en el que el usuario podrá elegir el tipo de material que desea. Una vez seleccionado, se mostrará otro cuadro de diálogo con los cuatro materiales que el jugador va a ofrecer a cambio. Si se selecciona algún material que el jugador no tiene a su disposición, se mostrará en un mensaje de pop-up, que no hay materiales suficientes. Si se puede hacer el cambio, se volverá a la pantalla principal y el jugador tendrá la cantidad de materiales actualizados		

Tabla 3.30. RSF-07. Comerciar.

Identificador:	RSF-08	Necesidad:	Esencial
Título:	Siguiente jugador		
Fuente:	RUC-14		
Descripción:	al pulsar el botón de siguiente jugador dentro del menú del Smartphone, se refresca la pantalla principal, mostrando los datos del jugador al que se le ha pasado el turno		

Tabla 3.31. RSF-08. Siguiente jugador.

Los requisitos software de Rendimiento definen valores relacionados con el rendimiento (velocidad de procesado o ejecución del sistema software):

Identificador:	RSR-01	Necesidad:	Deseable
Título:	Cargar el juego		
Fuente:	ANA		
Descripción:	el juego debe tardar poco tiempo en cargar		

Tabla 3.32. RSR-01. Cargar el juego.

Identificador:	RSR-02	Necesidad:	Deseable
Título:	Actualización de pantallas		
Fuente:	ANA		
Descripción:	el refresco de las pantallas para la actualización de datos durante el juego, debe ser imperceptible a la vista y rápido		

Tabla 3.33. RSR-02. Actualización de pantallas.

Los requisitos de interfaz especifican con qué hardware y software el que el sistema tendrá que interactuar:

Identificador:	RSI-01	Necesidad:	Esencial
Título:	Formato de imágenes		
Fuente:	ANA		
Descripción:	El formato requerido de las imágenes es de .PNG		

Tabla 3.34. RSI-01. Formato de imágenes.

Identificador:	RSI-02	Necesidad:	Esencial
Título:	Layouts xml		
Fuente:	ANA		
Descripción:	Los layouts de las pantallas se definen en archivos xml		

Tabla 3.35. RSI-02. Layouts xml.

Los requisitos de recursos definen qué recursos físicos son necesarios para la ejecución de la aplicación.

Identificador:	RSRe-01	Necesidad:	Deseable
Título:	Entorno de desarrollo		
Fuente:	ANA		
Descripción:	El ordenador con el que se desarrollará el juego ha de tener un emulador del dispositivo móvil		

Tabla 3.36. RSRe-01. Entorno de desarrollo.

Identificador:	RSRe-02	Necesidad:	Esencial
Título:	Entorno de producción		
Fuente:	ANA		
Descripción:	El juego se debe ejecutar en un dispositivo móvil HTC Legend con sistema Android		

Tabla 3.37. RSRe-02. Entorno de producción.

3.1.4. Casos de uso

Los casos de uso ayudan a definir a grandes rasgos, las relaciones que hay entre los actores (jugadores en este caso), y el sistema.

Los casos de uso vienen definidos por:

- **Identificador**: muestra el tipo de atributo CU (Caso de Uso) más un número secuencial que representará de forma unívoca el caso.
- **Título**: con el nombre del caso de uso.

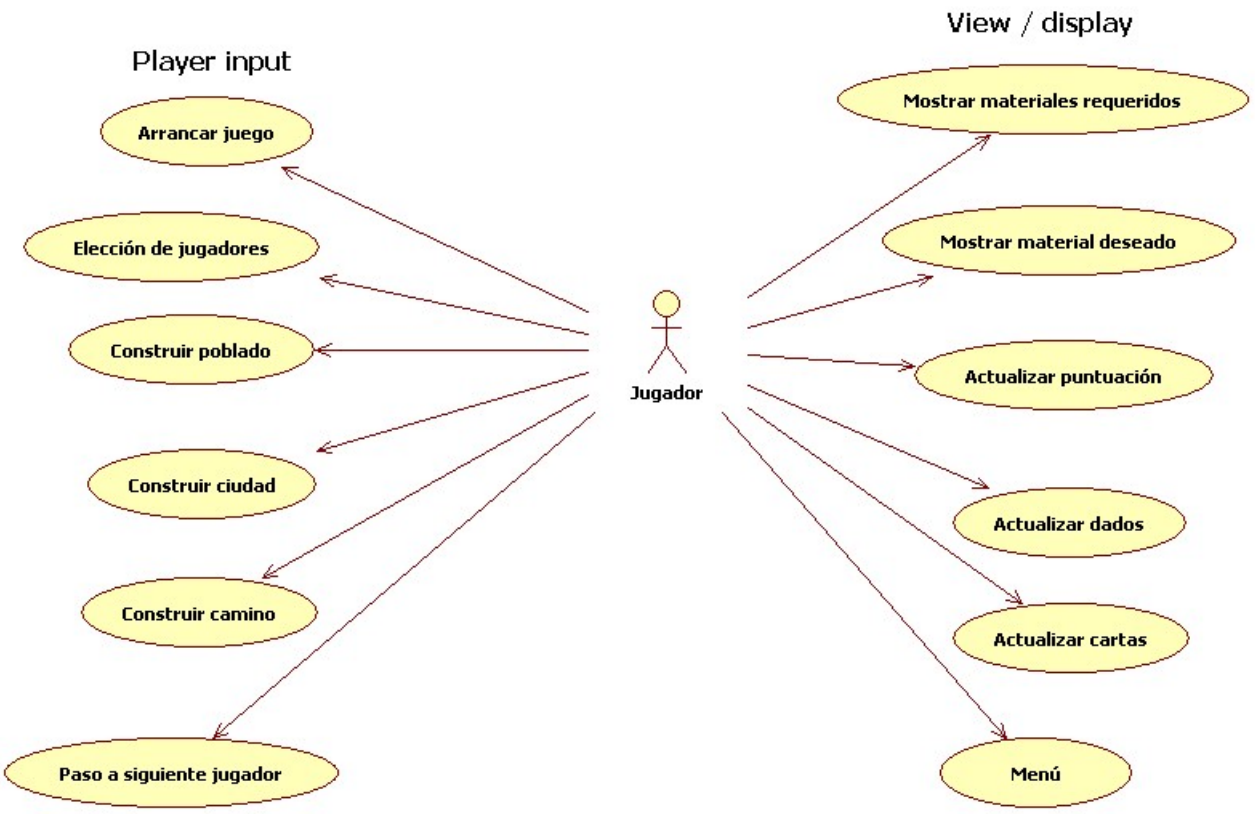


Figura 3.1. Diagrama de Casos de Uso. ..

- **Actores**: principales actores del caso de uso
- **Objetivo**: objetivo a cumplir por el caso de uso.
- **Precondiciones**: condiciones previas que se han de cumplir para que se pueda dar el caso de uso.
- **Post-condiciones**: condiciones que se producirán una vez se haya dado el caso de uso.

A continuación se definen de forma más detallada los distintos casos de uso:

Identificador:	CU-01
Título:	Arrancar el juego
Actores:	Jugador
Objetivo:	comenzar el juego
Precondiciones:	tener instalado el juego en el terminal
Post-condiciones:	se carga el menú principal del juego, el usuario puede elegir entre las opciones del menu

Tabla 3.38. CU-01. Arrancar el juego.

Identificador:	CU-02
Título:	Elección de jugadores
Actores:	Jugador
Objetivo:	Seleccionar en el cuadro de diálogo el número de jugadores de la partida
Precondiciones:	haber arrancado el juego y pulsado nuevo juego
Post-condiciones:	se carga una partida para dos o tres jugadores

Tabla 3.39. CU-02. Elección de jugadores.

Identificador:	CU-03
Título:	Menú
Actores:	Jugador
Objetivo:	mostrar un menú con las opciones de construir, comerciar y paso a siguiente jugador
Precondiciones:	la partida ha de estar comenzada y pulsado el botón menú del smartphone
Post-condiciones:	el sistema muestra el menú de opciones

Tabla 3.40. CU-03. Menú.

Identificador:	CU-04
Título:	Mostrar construcciones
Actores:	Jugador
Objetivo:	mostrar una lista con las posibles construcciones y su coste en materiales
Precondiciones:	haber pulsado el botón menú del smartphone y el botón construir dentro del menú
Post-condiciones:	el sistema muestra una lista de opciones

Tabla 3.41. CU-04. Mostrar construcciones.

Identificador:	CU-05
Título:	Construir poblado
Actores:	Jugador
Objetivo:	situar un poblado en el tablero
Precondiciones:	haber pulsado el botón menú del smartphone, el botón construir dentro del menú, haber seleccionado el poblado y tener los materiales necesarios
Post-condiciones:	el jugador puede situar el poblado en una de las posiciones válidas

Tabla 3.42. CU-05. Construir poblado.

Identificador:	CU-06
Título:	Construir ciudad
Actores:	Jugador
Objetivo:	situar una ciudad en el tablero
Precondiciones:	haber pulsado el botón menú del smartphone, el botón construir dentro del menú, haber seleccionado la ciudad y tener los materiales necesarios
Post-condiciones:	el jugador puede situar una ciudad en una de las posiciones válidas

Tabla 3.43. CU-06. Construir ciudad.

Identificador:	CU-07
Título:	Construir camino
Actores:	Jugador
Objetivo:	situar un camino en el tablero
Precondiciones:	haber pulsado el botón menú del smartphone, el botón construir dentro del menú, haber seleccionado el camino y tener los materiales necesarios
Post-condiciones:	el jugador puede situar un camino en una de las posiciones válidas

Tabla 3.44. CU-07. Construir camino.

Identificador:	CU-08
Título:	Mostrar materiales requeridos
Actores:	Jugador
Objetivo:	mostrar una lista con los materiales requeridos para poder comerciar
Precondiciones:	haber pulsado el botón menú del smartphone y el botón comerciar dentro del menú
Post-condiciones:	se muestra una lista con los materiales requeridos para que se pueda comerciar

Tabla 3.45. CU-08. Mostrar materiales requeridos.

Identificador:	CU-09
Título:	Mostrar material deseado
Actores:	Jugador
Objetivo:	mostrar una lista con los materiales deseados durante la fase de comercio
Precondiciones:	haber pulsado el botón menú del smartphone, el botón comerciar dentro del menú, haber seleccionado uno de los materiales a cambiar y tener ese material en la cantidad suficiente
Post-condiciones:	en la pantalla principal se muestran actualizados los materiales del jugador

Tabla 3.46. CU-09. Mostrar material deseado.

Identificador:	CU-10
Título:	Siguiente jugador
Actores:	Jugador
Objetivo:	pasar el turno al siguiente jugador actualizando sus datos en la pantalla principal
Precondiciones:	haber pulsado el botón menú del smartphone y el botón Siguiente jugador
Post-condiciones:	se pasa al turno del siguiente jugador

Tabla 3.47. CU-10. Siguiente jugador.

Identificador:	CU-11
Título:	Actualizar puntuación
Actores:	Jugador
Objetivo:	actualizar la puntuación del jugador cuando ha añadido alguna nueva construcción
Precondiciones:	haber construido (poblado, ciudad o camino)
Post-condiciones:	se actualiza la puntuación

Tabla 3.48. CU-11. Actualizar puntuación.

Identificador:	CU-12
Título:	Actualizar dados
Actores:	Jugador
Objetivo:	mostrar en el tablero la puntuación obtenida en los dados
Precondiciones:	haber pulsado el botón de la pantalla principal de lanzar los dados
Post-condiciones:	se muestra el valor

Tabla 3.49. CU-12. Actualizar dados.

Identificador:	CU-13
Título:	Actualizar cartas
Actores:	Jugador
Objetivo:	actualizar los materiales disponibles por cada jugador
Precondiciones:	haber pulsado el botón de lanzar los dados, haber construido, haber comerciado
Post-condiciones:	se actualizan los materiales

Tabla 3.50. CU-13. Actualizar cartas.

3.1.5. Diagrama de actividad del sistema

Una vez se han definido las interacciones entre el sistema y el usuario a través de los casos de uso, se especifica el diagrama de actividad: se muestra mediante diagramas de flujo la secuencia de actividades que seguirá el juego.

Un diagrama de actividad es una representación gráfica del flujo de trabajo, en el que las actividades y acciones pueden estar asociadas a una elección, una iteración o una concurrencia. Gracias al diagrama de actividad se comprender mejor las transiciones y eventos del juego.

En la figura 3.2 se muestra el diagrama de actividad asociado al juego.

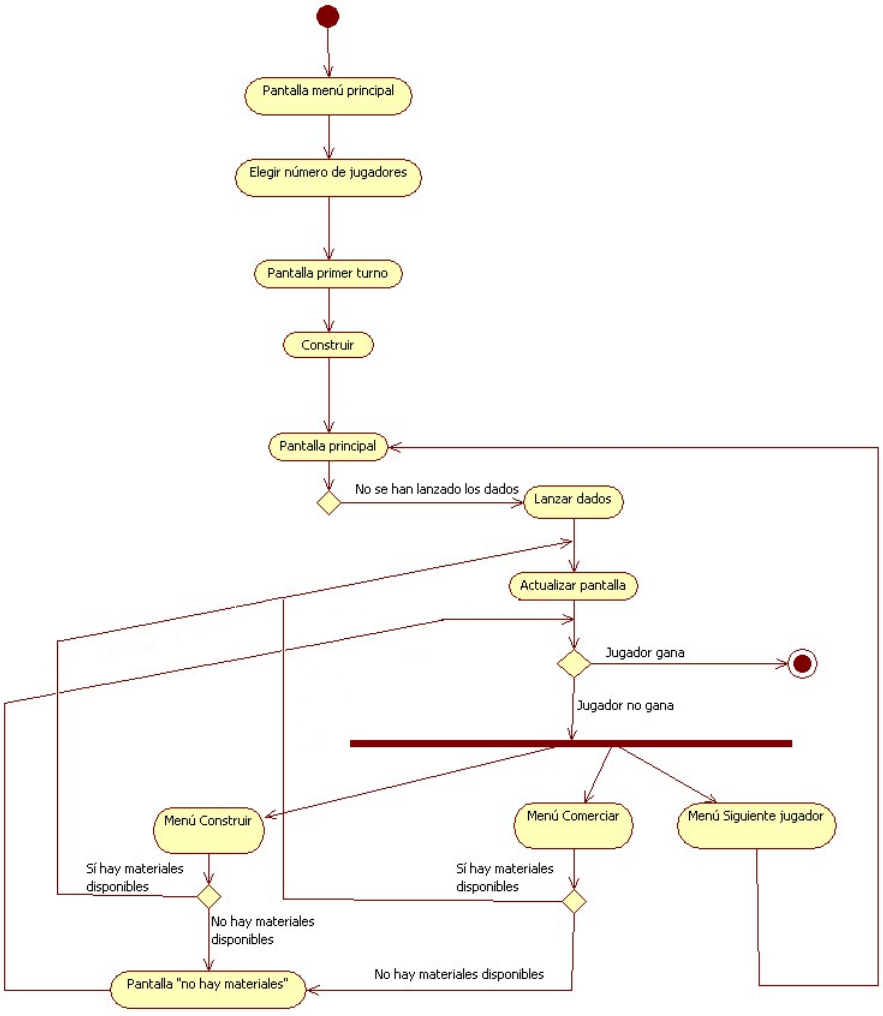


Figura 3.2. Diagrama Actividad. Flujo del juego.

3.1.6. Diagramas de secuencia

Gracias a los diagramas de secuencia, se puede representar la forma en que varios objetos se comunican entre sí para la realización de los casos de uso.

Con los diagramas de secuencia se facilita la creación del modelo de clases, ya que que contiene información sobre la implementación del escenario (incluyendo objetos y clases, y mensajes intercambiados entre objetos).

La figura 3.3 representa el diagrama de secuencia del primer turno.

La clase **Colonos** es la que inicializa el sistema al comienzo del juego, lleva el control de turnos y también la gestión del lanzamiento de los dados. Una vez se inicializan los datos, se guardan a través de la clase **ColonosAppState**. Gracias a esta clase, se tiene un contenedor global con datos comunes, como la disposición del tablero, o datos individuales de cada jugador, como las cartas disponibles.

Durante el primer turno, Colonos llamará a la clase **firstTurn**, que muestra un mensaje al usuario en el que se dan instrucciones acerca de la dinámica a seguir durante el primer turno. Una vez se confirma la lectura del mensaje, firstTurn llama a la clase Player.

La clase **Player** tiene todos los datos relativos al jugador que está en su turno. Al comienzo de la ejecución, se cargarán a través de ColonosAppState los datos de todo el juego y del jugador en cuestión, y antes de pasar al siguiente jugador, se actualizarán todos los que se hayan visto modificados. Los datos que han podido cambiar de valor, pueden ser tanto del jugador que está en su turno, como del jugador o jugadores que estén en espera, como datos globales del juego.De esta manera, al destruirse el hilo de Player, el siguiente jugador tendrá a su disposición todos los datos necesarios para poder continuar con el juego.

Un hilo de esta clase Player, se ejecutará en paralelo a la clase Colonos, la cual no terminará su ejecución hasta la finalización del juego. Dentro de la clase Player, siempre se lanza otro hilo con las vistas dinámicas del juego (construcciones, dados y cartas). Este hilo pertenece a la clase **CitiesView**, que lleva el control de las vistas sujetas a

posibles modificaciones.

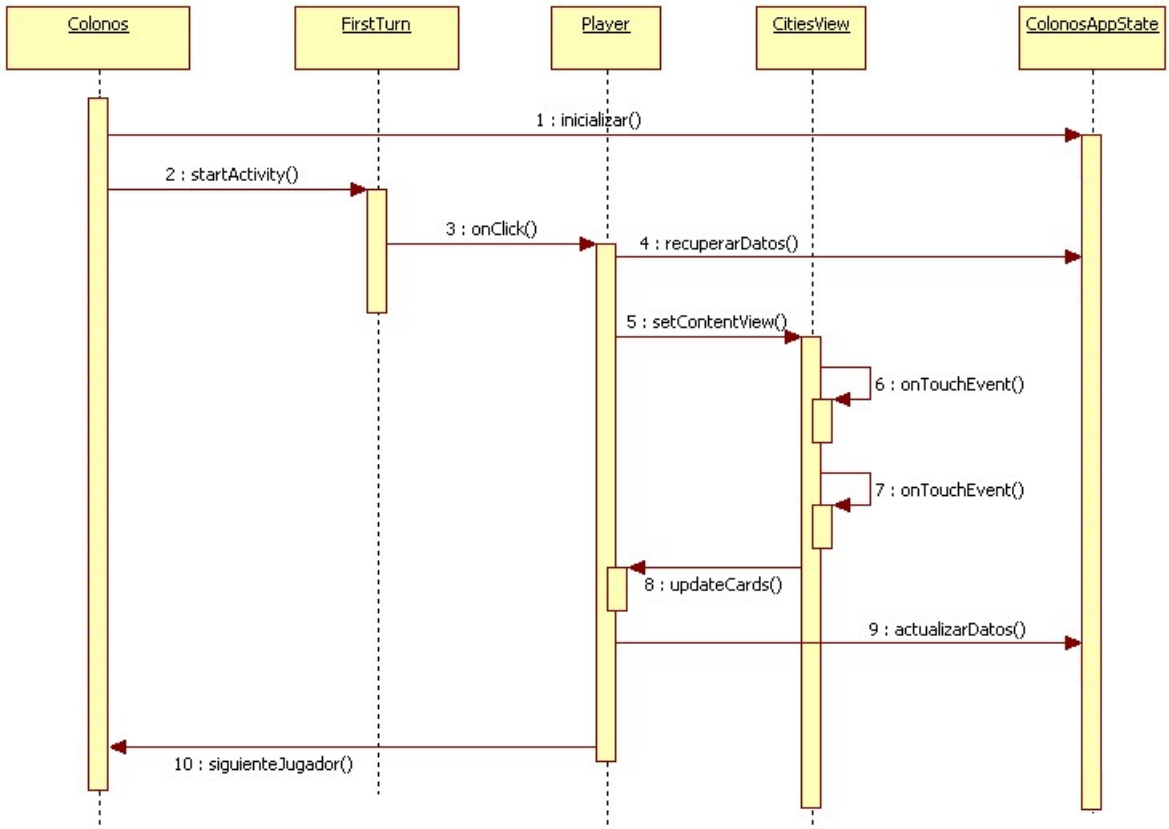


Figura 3.3. Diagrama de secuencia. Primer turno.

La figura 3.4 muestra el diagrama de secuencia de la la construcción durante el juego.

Cuando un jugador desee construir, se comprobará si hay materiales disponibles, y en caso afirmativo, se actualizará la vista del tablero (una vez que el jugador coloque la construcción elegida) y las cartas que quedan disponibles al jugador.

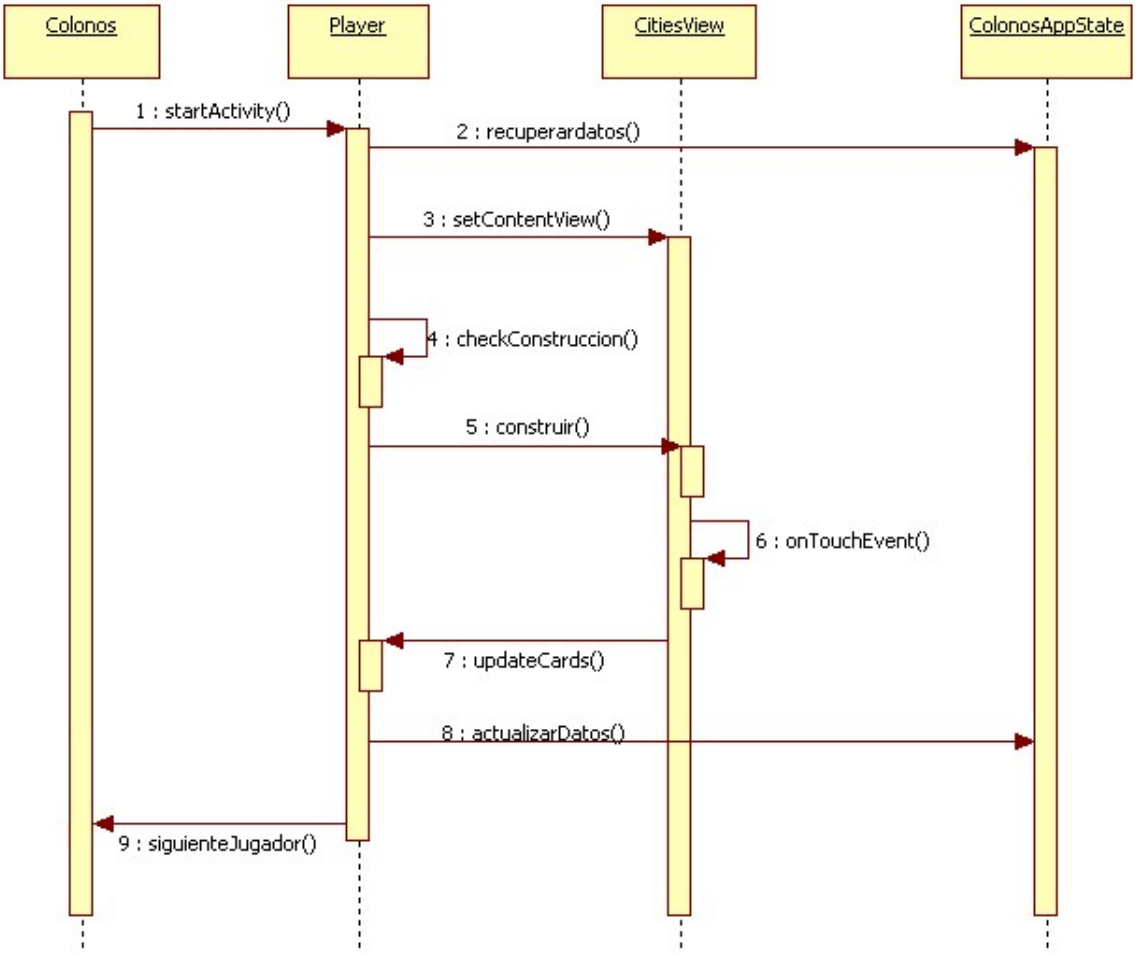


Figura 3.4. Diagrama de secuencia. Construcción.

3.2. Diseño

Con los datos obtenidos durante la fase de análisis, se realiza un diseño de los componentes, de forma que todos los aspectos queden claros y definidos para la fase de implementación. Con toda esta información, durante la fase de implementación no será necesario replantear ningún punto. El diseño se ha de adaptar a las necesidades del juego, y las clases se han de definir de manera exhaustiva, de manera que las relaciones entre ellas queden claras.

Si la fase de diseño no se realiza de manera concienzuda, puede repercutir en una necesidad de replantear el diseño, que implicaría un cambio en la planificación del proyecto y por lo tanto en su fecha de entrega.

Esta sección se ha estructurado en varios apartados:

- **Arquitectura** define a alto nivel la estructura del juego
- **Pantallas** muestra las distintas interfaces a utilizar
- **Carpetas del proyecto** contienen el árbol con la organización de las mismas
- **Diagramas de clases** con sus dependencias entre ellas

3.2.1. Arquitectura

La arquitectura[18] del juego proporciona el diseño de más alto nivel de la estructura del juego y define los módulos principales, las responsabilidades de cada módulo, la interacción que existirá entre dichos módulos y el control de flujo y datos. Aporta una visión abstracta de alto nivel, dejando el detalle de cada módulo a fases posteriores del diseño.

La arquitectura del juego se divide en tres componentes principales.

El componente más importante es el core del juego: inicializa componentes y gestiona todo el flujo del juego. La entrada/salida engloba las interacciones del usuario con el juego y las vistas de los menús.

Figura 3.5. Arquitectura. Diagrama de componentes.

La figura 3.5 muestra la arquitectura del sistema.

3.2.2. Pantallas

En este apartado se definen los diseños preliminares que se realizaron de las diferentes pantallas.

Puesto que la pantalla del dispositivo móvil es de un tamaño muy limitado, para poder aprovechar éste al máximo, todas las pantallas están forzadas a una orientación horizontal.

La figura 3.6 representa el diseño de la pantalla de inicio del juego. Un gran título precede a tres botones que permiten al usuario comenzar una partida, leer las instrucciones o la información sobre la autora.

La pantalla principal (figura 3.7) tendrá un tablero formado por celdas hexagonales (representado mediante un círculo en el diseño preliminar), un botón para lanzar los dados (o el valor obtenido después de ese lanzamiento). un texto con el turno del jugador

Figura 3.6. Diseño de pantallas. Menú de inicio.

Figura 3.7. Diseño de pantallas. Tablero del juego.

Figura 3.8. Diseño de pantallas. Mensaje informativo o de error.

correspondiente, los materiales o cartas disponibles por ese jugador (representado por rectángulos) y otro texto con la puntuación acumulada por el mismo.

Cuando se tenga que mostrar un mensaje informativo o de error, la pantalla (figura 3.8) tendrá ese mensaje y un botón de aceptación o confirmación de lectura de ese mensaje.

La pantalla de instrucciones (figura 3.9) tiene toda la información necesaria para que alguien que no ha jugado nunca, entienda el mecanismo del juego. Esta pantalla tiene una barra de scroll para poder visualizar todas las instrucciones sin problema.

Figura 3.9. Diseño de pantallas. Pantalla con las instrucciones.

3.2.3. Carpetas del proyecto

Al programar con Android y Eclipse, la estructura del proyecto[19] queda definida en la figura 3.10.

La carpeta /**src** almacena todos los ficheros fuente de Java que se van creando durante el desarrollo del juego.

La carpeta /**gen**, contiene el archivo R.java que se genera automáticamente por la plataforma android. Este archivo es un índice con todos los recursos definidos en el fichero y ayuda a localizar rápida e interactivamente la referencia específica que se está buscando.

/**Android 2.1** contiene las librerias (jars) que son necesarias para el proyecto. El 2.1 indica el número de versión.

La carpeta /**res** contiene todos los recursos externos como imágenes, ficheros de datos, archivos de audio, etc. En /**res**/**drawable** se guardan todas las imágenes, iconos o recursos visuales que se usan en el juego. /**res**/**layout** contiene los layouts (en formato XML) de las diferentes interfaces de usuario. /**res**/**menu** almacena el recurso de menú: se trata de un archivo XML que define los opciones posibles cuando se pulsa el botón menú del smartphone. Para guardar los recursos de audio (.mp3), se utiliza la carpeta /**res**/**raw**. /**res**/**values** que también contiene archivos XML, declaran arrays y strings que se referencian en la aplicación, de tal manera que se pueden cambiar sus valores

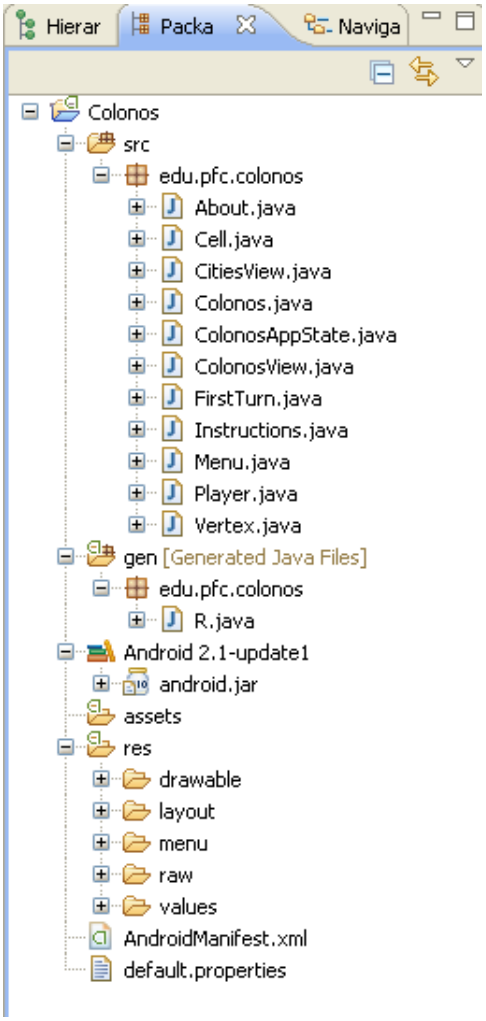


Figura 3.10. Carpetas del proyecto. Organización.

localmente, sin necesidad de cambiar el código fuente. La carpeta /assets también puede contener recursos externos como en la carpeta /res, sin embargo estos archivos se han de guardar en formato raw, por lo que se ha hecho uso de la misma.

Los paquetes de XML se detallan en la figura 3.11.

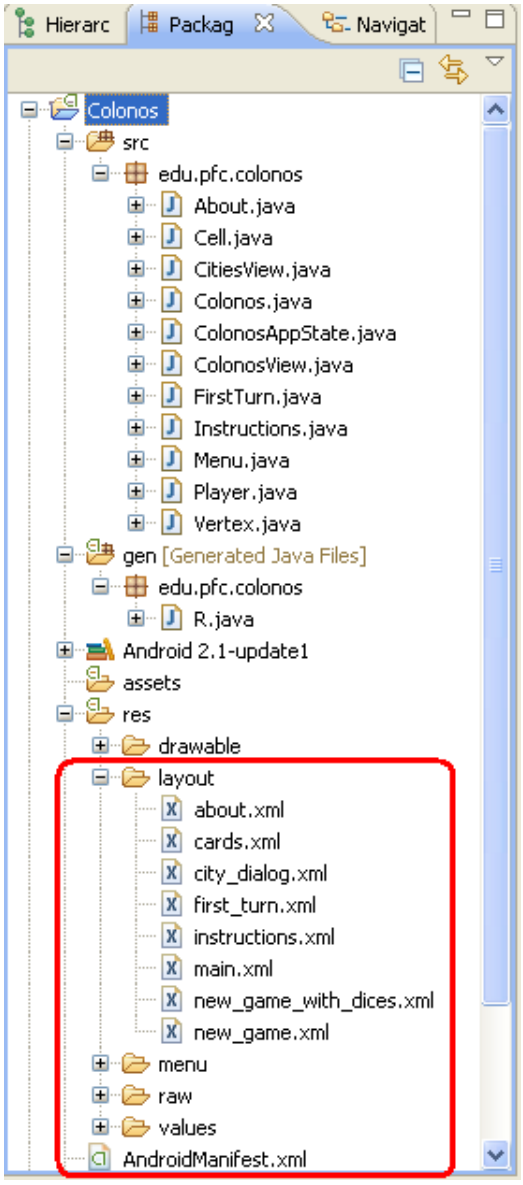


Figura 3.11. Carpetas del proyecto. Organización XML.

3.2.4. Diagrama de clases

En este apartado se definen las clases a utilizar y cuales son las relaciones entre ellas. Según las buenas prácticas para programar una aplicación móvil con Android, se necesita una clase por cada vista o pantalla. Además de estas clases, se tienen otras clases auxiliares necesarias para la correcta gestión del juego. El diagrama de clases representado en la figura 3.12, muestra todas las que se han definido en este proyecto.

Todas las clases que extienden de Activity han de ir definidas en una archivo .xml llamado AndroidManifest. Una Activity es una entidad sencilla que se usa para llevar a cabo acciones. Una aplicación puede tener como es el caso, muchas activities diferentes, pero el usuario sólo interactúa con ellas de una en una.

Otras clase extienden de View, y otras son las que se han definido para ser auxiliares. La clase ColonosAppState extiende de Application.

Además de los métodos indicados, también se han definido gets y sets para recuperar o editar los valores de los atributos privados.

A continuación se detalla cada una de las clases.

La clase **Menu** 3.51 es de la que arranca la aplicación. No tiene ningún miembro y cuenta con cuatro métodos. Esta clase mostrará el menú de inicio (figura 3.6) y en caso de elegir un nuevo juego (startGame(int i)), aparecerá un nuevo cuadro de diálogo en el que se puede elegir el número de jugadores (openNewGameDialog()). El método onCLick(View v) es el que detecta que botón del menú se ha pulsado, y en caso de que se pulse o Instrucciones o Acerca de, se llamará a la clase correspondiente. A través de esta clase Menu, también se fijan en ColonosAppState 3.56 las variables del número de jugadores y que se trata de un juego nuevo.

La clase **Colonos** 3.52 se llama desde la clase Menu en el arranque del juego. La primera vez que es llamada, inicializa todos los recursos (tablero aleatorio, materiales disponibles por cada jugador y construcciones del tablero). Tiene un único miembro (Random rand) que representará el valor obtenido en los dados y es la que lleva el control de turnos (llamará a la clase Player, y cuando el jugador termine su turno, éste

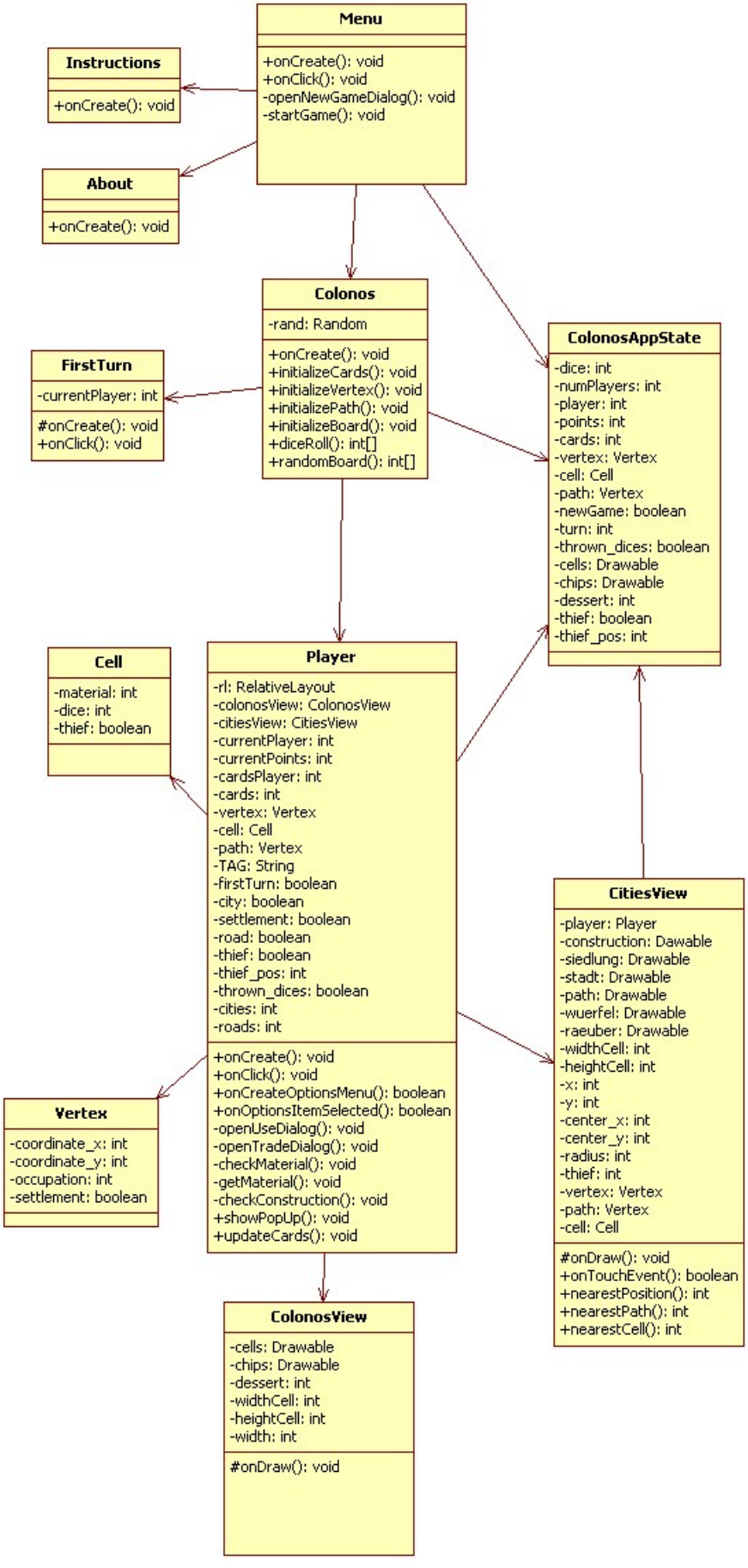


Figura 3.12. Diagrama de clases. Clases y relaciones.

Clase:	Menu	Herencia:	
Miembros:			
Métodos:	void onCreate(Bundle savedInstanceState) void onClick(View v) void openNewGameDialog() void startGame(int i)		

Tabla 3.51. Clase Menu. Miembros y métodos.

volverá a llamar a esta clase).

Clase:	Colonos	Herencia:	
Miembros:	Random rand		
Métodos:	void onCreate(Bundle savedInstanceState) void initializeCards(int numPlayers) void initializeVertex() void initializePath() void initializeBoard() int[] diceRoll() int[] randomBoard(int[] cells)		

Tabla 3.52. Clase Colonos. Miembros y métodos.

La clase **Player** 3.53 gestiona las acciones llevadas a cabo por un jugador. Tiene como miembros los objetos relacionados con las vistas (rl, inflater, colonosView, citiesView), qué jugador tiene el turno en ese momento, su puntuación, los materiales que tiene disponibles (y los materiales del resto de jugadores), el valor que han obtenido en los dados, en que posición se encuentra el ladrón, la información de las construcciones, si es el primer turno, si puede construir, si ha sacado un 7 y puede mover al ladrón o si ha tirado los dados. Desde el método onCreate() se llama a la clase ColonosView y CitiesView, que son las gestionan las vistas estáticas (fondo, celdas del tablero, fichas) y las dinámicas (puntuación, número de materiales disponibles, valor de los dados, construcciones, ladrón). Con el método onClick(), al pulsar el botón de dados, se muestra el valor obtenido y un sonido de lanzamiento. El método onOptionsItemSelected(MenuItem item), es el que se llama con el botón menú del smartphone, permitiendo al jugador construir, comerciar o

pasar al siguiente jugador. Los métodos openUseDialog() y openTradeDialog() muestran cuadros de diálogo para construir y comerciar respectivamente. En caso de que no haya materiales suficientes (se comprueba con checkConstruction() o checkMaterial()), aparecerá un mensaje de error (pantalla de error de la figura 3.8). El método updateCards() se llamará en caso de que se haya comerciado con la banca (se restarán los cuatro materiales que el jugador quería cambiar y se sumará el que no tenía pero deseaba). Este método updateCards() también se llama desde la clase CitiesView siempre que se ha podido construir y hay que restar los materiales que se han gastado. Además de todos estos métodos, se dispone de los gets y sets correspondientes.

La clase **ColonosView** 3.54 es la que se encarga de las vistas estáticas del juego, es decir, todo aquello que no varía una vez se ha inicializado el juego. Sus miembros son cells y chips, arrays de objetos Drawable que representan las celdas del tablero y la ficha con su valor numérico, dessert de tipo entero, que indica en que celda se encuentra el desierto y por tanto no hay que dibujar ficha en ella, widthCell y heightCell, con el tamaño de los Drawable y width, que guarda el tamaño de la pantalla. El único método de esta clase es onDraw(), que se encarga de pintar el fondo del tablero y cada celdas y su ficha correspondiente (excepto en la celda del desierto que no hay ficha).

La clase **CitiesView** 3.55 es llamada por Player a continuación de ColonosView. Es la encargada de cargar y refrescar las vistas dinámicas de la pantalla principal. Tiene como miembros player (para poder acceder a los datos del jugador que está en su turno en ese momento), construction, que almacena los Drawable de los poblados, ciudades y caminos de diferentes colores, widthCell y heightCell, referencias para el tamaño de las construcciones, x e y, coordenadas almacenadas cuando se pulsa sobre el tablero, center_x y center_y, coordenadas exactas donde debería estar la construcción, radius, margen dentro del cual es posible dibujar la construcción, thief, coordenadas donde se coloca el ladrón, vertex y path, con la información de las posiciones válidas que ya han sido ocupadas y por quien, y cell, para poder gestionar en que celda está el ladrón y así no actualizar materiales si es el caso. El método onDraw() se encarga de dibujar. Cada vez que se pone la instruccion invalidate(), se vuelve a llamar a este método para que se refresque la pantalla cuando alguno de los valores ha cambiado. onTouchEvent() es llamado cada vez que se toca la pantalla. OnTouchEvent() realizará distintas acciones

si hay que mover el ladrón (el jugador ha sacado un 7), es el primer turno de cada jugador, o puede poner alguna construccion. Para mover el ladrón se consulta desde onTouchEvent() el método nearestCell(), que devuelve la celda más cercana a la posición que se ha pulsado. Si se pulsa fuera del tablero este método no devuelve las coordenadas de ninguna celda. El método nearestPosition(), devuelve las coordenadas de el vértice más cercano al la parte del tablero pulsada. Es llamado cuando se quiere poner alguna construcción del tipo poblado o ciudad. Si la construcción es un camino, se llama al método nearestPath().

La clase **ColonosAppState** 3.56, tiene todos los datos de la partida: el valor de los dados (dice), los puntos de todos los jugadores (points), si es el primer turno de cada jugador (turn), de qué jugador es el turno (player), en que celda está el desierto (dessert), los materiales disponibles de todos los jugadores (cards), la información de que hay construido y por quién (vertex para poblados y caminos y path para caminos), el valor de dados asociado a cada celda, el material que se obtiene y si está el ladrón (cell), si es nuevo juego (newGame, si se han lanzado los dados (thrown_dices), si se ha sacado un siete y hay que mover el ladrón (thief) y los Drawable para dibujar el tablero (cells y chips). Los métodos de esta clase son los gets y sets.

La clase **Instructions** 3.57 no tiene ningún miembro y el método onCreate es el que se encarga de cargar la vista de esta pantalla.

La clase **About** 3.58 al igual que instructions, se muestra a través de onCreate la vista de esta pantalla.

La clase **Cell** 3.59 tiene como miembros material (con el recurso asociado a esa celda), dice (con el valor de dados) y thief (que indica si en esa celda está el ladrón). Los métodos asociados son los constructores y los gets y sets.

La clase **Vertex** 3.60 tiene las coordinadas x e y de un cada uno de los vértices de las celdas, la ocupación indicando si hay alguna construcción y de qué jugador es y un boolean que indica si, de haber construcción, es poblado o ciudad.

La clase **FirstTurn** 3.61 llama desde onCreate a la vista con indicaciones de la dinámica que han de seguir los jugadores durante el primer turno.

Clase:	Player	Herencia:	
Miembros:	RelativeLayout rl		
	LayoutInflater inflater		
	ColonosView colonosView		
	CitiesView citiesView		
	int currentPlayer, currentPoints		
	int[] cardsPlayer, dices, thief_pos		
	int[][] cards		
	Vertex[][] vertex, path		
	Cell[][] cell		
	boolean firstTurn, city, settlement, road, thief, thrown_dices		
	int cities, roads, INVISIBLE		
Métodos:	void onCreate(Bundle savedInstanceState)		
	void onClick(View v)		
	boolean onCreateOptionsMenu(Menu menu)		
	boolean onOptionsItemSelected(MenuItem item)		
	void openUseDialog(); void openTradeDialog()		
	void checkMaterial(final int i); void getMaterial(int i, int j)		
	void checkConstruction(int i)		
	void showPopUp(final int i)		
	void updateCards()		
	int getCurrentPlayer()		
	int getCurrentPoints()		
	void setCurrentPoints(int points)		
	boolean getCity(); void setCity(boolean city)		
	boolean getThief(); void setThief(boolean thief)		
	int getCities()		
	void setRoads(int roads); int getRoads()		
	void setCities(int cities)		
	boolean getFirstTurn()		
	boolean getSettlement(); void setSettlement(boolean settlement)		
	boolean getRoad(); void setRoad(boolean road)		
	boolean getThrownDices()		
	int[] getDices()		

Tabla 3.53. Clase Player. Miembros y métodos.

Clase:	ColonosView	Herencia:	
Miembros:	Drawable[] cells Drawable[] chips int dessert int widthCell, heightCell int width		
Métodos:	void onDraw(Canvas canvas)		

Tabla 3.54. Clase ColonosView. Miembros y métodos.

Clase:	CitiesView	Herencia:	
Miembros:	Player player Drawable[][] construction Drawable[] siedlung, stadt, path_d_u, path_u_d, path_, wuerfel Drawable raeuber_w int widthCell, heightCell, x, y, center_x, center_y, radius int[] thief Vertex[][] vertex, path Cell[][] cell		
Métodos:	void onDraw(Canvas canvas) boolean onTouchEvent(MotionEvent event) int[] nearestPosition(int x, int y) int[] nearestPath(int x, int y) int[] nearestCell(int x, int y) Vertex[][] getVertex() void setVertex(Vertex[][] vertex) Vertex[][] getPath() void setPath(Vertex[][] path) void setCell(Cell[][] cell) Cell[][] getCell() void setThief(int[] thief) int[] getThief()		

Tabla 3.55. Clase CitiesView. Miembros y métodos.

Clase:	ColonosAppState	Herencia:	
Miembros:	int[] dice, points, thief_pos int numPlayers, player, turn, dessert int[][] cards Vertex[][] vertex, path Cell[][] cell boolean newGame, thrown_dices, thief Drawable[] cells, chips		
Métodos:	int[] getDice(); void setDice(int[] dice) int getNumPlayers(); void setNumPlayers(int numPlayers) int getPlayer(); void setPlayer(int player) void setPointsArray(int i) void setPointsPlayer(int player, int points) int[] getPoints() int getPointsPlayer(int player) void setNewGame(boolean newGame); boolean getNewGame() void setTurn(int turn); int getTurn() void setThrownDices(boolean dices) boolean getThrownDices() void setCards(int[][] cards) void setCardsPlayer(int[] cards, int player); int[] getCardsPlayer(int player) int[][] getCards() Drawable[] getCells(); void setCells(Drawable[] cells) Drawable[] getChips(); void setChips(Drawable[] chips) int getDessert(); void setDessert(int dessert) int[] getThiefPos(); void setThiefPos(int[] thief_pos) boolean getThief(); void setThief(boolean thief) void setVertex(Vertex vertex[][]); Vertex[][] getVertex() void setCell(Cell cell[][]); Cell[][] getCell() void setPath(Vertex path[][]); Vertex[][] getPath()		

Tabla 3.56. Clase ColonosAppState. Miembros y métodos.

Clase:	Instructions	Herencia:	
Miembros:			
Métodos:	void onCreate(Bundle savedInstanceState)		

Tabla 3.57. Clase Instructions. Miembros y métodos.

Clase:	About	Herencia:	
Miembros:			
Métodos:	void onCreate(Bundle savedInstanceState)		

Tabla 3.58. Clase About. Miembros y métodos.

Clase:	Cell	Herencia:	
Miembros:	int material int dice boolean thief		
Métodos:	void setMaterial(int material) void setDice(int dice) void setThief(boolean thief) int getMaterial() int getDice() boolean getThief()		

Tabla 3.59. Clase Cell. Miembros y métodos.

Clase:	Vertex	Herencia:	
Miembros:	int coordinate_x int coordinate_y int occupation boolean settlemen		
Métodos:			

Tabla 3.60. Clase Vertex. Miembros y métodos.

Clase:	FirstTurn	Herencia:	
Miembros:			
Métodos:	void onCreate(Bundle savedInstanceState) void onClick(View v) int getCurrentPlayer()		

Tabla 3.61. Clase FirstTurn. Miembros y métodos.

3.3. Implementación

En esta sección no se comenta el código del juego, si no que se detallan los aspectos importantes y relevantes de esta fase. Sirve como guía de esta fase el estudio anterior de análisis y diseño, y se sigue un ciclo de vida funcional: el juego es operativo desde las primeras etapas de la implementación y se van incrementando las funcionalidades.

El capítulo se organiza de forma paralela al desarrollo del juego: en primer lugar se implementó una correcta navegación entre pantallas, a continuación se desarrollaron las vistas e interfaces de las distintas pantallas y por último se desarrolló el núcleo del juego. En último lugar se añadieron detalles como cuadros de diálogo, sonidos. Además de estos apartados, el primero de ellos muestra que cambios fueron necesarios durante esta fase.

3.3.1. Implementación previa y cambios

Antes de comenzar con la implementación, fue necesario seguir tutoriales de android para llegar a familiarizarse con el entorno de trabajo y la plataforma [20].

Durante la fase de implementación se tuvo que cambiar el diseño de sistema de coordenadas. En un primer momento se planteó como un sistema de coordenadas cartesianas, pero durante la implementación se detectó que no era una forma eficiente de trabajar con la información: cada vez que el sistema debe verificar algún dato relativo a una celda, vértice o lado, es necesario comprobar los datos asociados a las celdas, vértices y lados vecinos. Dado que las celdas son hexagonales, fue necesario cambiar el sistema cartesiano, por otro que cumpliese con las expectativas de eficiencia (sección 3.3.4). Esto provocó un retraso en la fase de implementación, y por lo tanto un cambio en la planificación.

Otro problema que surgió durante la fase de implementación fue el de la orientación de las pantallas. Android permite que la orientación de la pantalla se cambie automáticamente cuando el teléfono se gira y esto es factible de implementar, ya que se pueden definir diferentes layouts dependiendo de la orientación (los layouts se ven en apartados sucesivos). Dado que el tamaño de la pantalla es muy limitado, si la orientación es variable no se aprovechaba la pantalla en su totalidad (el tablero es más ancho que alto, por lo que para que entre de forma vertical y horizontal, el tablero tiene que ser más pe-

queño). En un primer momento se implemento para que el juego funcionase en cualquier orientación, pero a la hora de colocar construcciones, resultaba muy complicado porque el area a pulsar era muy pequeña.

Por ello se cambio la orientación (figura 3.13) para que fuese siempre horizontal y así aprovechar la pantalla al máximo, lo que también acarreó un retraso en el proyecto por que se tuvieron que redifinir los tamaños de las celdas que forman el tablero.

```
android:screenOrientation="landscape"
```

Figura 3.13. Orientación horizontal

3.3.2. Navegación entre distintas pantallas

La primera fase de implementación se centró en la navegación entre pantallas (figura 3.14). Se buscaba que pulsando los distintos botones, se fuese navegando de unas pantallas a otras de forma adecuada. Como se ha comentado, es necesaria al menos una Activity (clase) por pantalla.

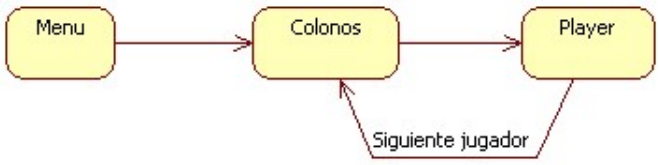


Figura 3.14. Pantallas. Flujo de navegación.

La Activity de Menu llama a Colonos y para hacer esta llamada, se utiliza un mensaje llamado intent. El intent es una estructura de datos pasiva que contiene una descripción abstracta de una operación que ha de ser ejecutada. En el caso de Menu, el objeto Intent se pasa a startActivity() de forma que se lanza la Activity de Colonos (figura 3.15).

```
Intent j = new Intent(this, Colonos.class);
startActivity(j);
```

Figura 3.15. Intent

La clase Colonos es la que lleva el control de turnos, valores de los dados e inicializa todos los datos al comienzo de la partida. Si no se está en el primer turno, Colonos llamará a Player con otro Intent y a continuación llamará al método finish (figura 3.16) que finaliza la Activity de Colonos que ya ha realizado su función.

```
finish();
```

Figura 3.16. Finalizar

El hilo de Menu no finaliza, de forma que si se pulsa la tecla back del smartphone desde Player (pantalla principal), se vuelve a esta pantalla. Si se añade la linea finish(), este hilo si que muere, haciendo que si se pulsa la tecla back, se sale de la aplicación.

Cuando el usuario pulsa siguiente jugador, terminará el hilo de Player después de haber llamado a Colonos.

3.3.3. Interfaz de usuario

En el apartado de Interfaz de usuario se define como se realizó la implementación de las pantallas, teniendo en cuenta el funcionamiento de los archivos de layout, la gestión de eventos de la interfaz y los menús.

Layouts

Los layouts [21] son la arquitectura para la interfaz de usuario en una Activity. Define la estructura del layout y contiene todos los elementos que aparecerán al jugador. Se pueden declarar layouts de dos formas:

- **Declarar elementos de interfaz de usuario en XML**: Android proporciona vocabulario XML que corresponde a las clases y subclases de View. Este tipo de layout se ha empleado en las interfaces sencillas como las que muestran el *Acerca de*, *Instrucciones*, los cuadros de diálogo, la información del primer turno, la pantalla principal con la elección de nuevo juego, etc.
- **Instanciar elementos de layout en tiempo de ejecución**: la aplicación puede

crear en tiempo de ejecución objetos de View y manipular sus propiedades programando. Esto se ha empleado para interfaces más complicadas como la vista de la pantalla con el tablero o las construcciones que van añadiendo los jugadores.

Por ejemplo para la clase Menu, se define un layout con tres botones. El primer paso es definir el archivo XML (figura 3.17).

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/
   android"
        android:layout_width="fill_parent" android:layout_height="
           fill_parent"
        android:background="@drawable/catan">
        <Button android:id="@+id/new_game_button" android:text="
           Nuevo juego"
                android:background="@drawable/new_game_button"
                   android:layout_marginLeft="5dip"
        android:layout_marginRight="5dip" android:layout_marginTop
           ="170dip" android:layout_marginBottom="10dip"/>
        <Button android:id="@+id/instrucciones_button"
           android:text="Instrucciones"
                android:background="@drawable/new_game_button"
                   android:layout_marginLeft="5dip"
        android:layout_marginRight="5dip"
           android:layout_marginBottom="10dip"/>
        <Button android:id="@+id/about_button" android:text="
           Acerca de..."
                android:background="@drawable/new_game_button"
                   android:layout_marginLeft="5dip"
                   android:layout_marginRight="5dip"/>
</TableLayout>
```

Figura 3.17. XML para botones

Una vez se ha definido el archivo XML, se carga desde el código (figura 3.18). En el método onCreate() (el primero que se llama cuando se lanza la Activity), se llama a setContentView, pasando como referencia el layout correspondiente con el formato R.layout.main. A continuación se crean instancias de los objetos de vistas que se han

definido, tres botones en este caso.

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        View newGameButton = this.findViewById(R.id.
            new_game_button);
        newGameButton.setOnClickListener(this);
        View instructionsButton = this.findViewById(R.id.
            instrucciones_button);
        instructionsButton.setOnClickListener(this);
        View aboutButton = this.findViewById(R.id.
            about_button);
        aboutButton.setOnClickListener(this);
}
```

Figura 3.18. Código para botones

Uno de los casos en los que se instancia el layout en tiempo de ejecución, es cuando se pinta la pantalla principal. Se han explorado dos formas de pasar los datos necesarios a las Views. La primera de ellas es modificar el constructor de la View para pasarle todos los atributos que necesite (ColonosView) y la otra es no pasarle ninguno de estos datos para que acceda a través de los gets y sets de Player (CitiesView).

Desde Player (onCreate()) se van a cargar las distintas vistas. En primer lugar se carga colonosView: se crea una instancia de la vista y después se llama a setContentView() con atributo la vista. En el caso anterior se pasaba el layout XML, ahora se pasa una instancia de View (figura 3.19).

```
Drawable[] cells = ((ColonosAppState) getApplication()).getCells();
Drawable[] chips = ((ColonosAppState) getApplication()).getChips();
int dessert = ((ColonosAppState) getApplication()).getDessert();
colonosView = new ColonosView(this, cells, chips, dessert);
setContentView(colonosView);
```

Figura 3.19. View

Ya que la vista de la pantalla principal está customizada, es necesario inflar el layout como se muestra en la figura 3.20 (en el caso de tener definido un XML como en el ejemplo del menú, esta operación se hace automáticamente):

```
this.inflater = (LayoutInflater) this.getSystemService(Context.
    LAYOUT_INFLATER_SERVICE);
```

Figura 3.20. Layout inflater

Se crea otra instancia de citiesView y a través de sets, se pasan los parámetros necesarios. Después de tener las instancia, ésta se debe añadir a la interfaz de manera que se visualicen los elementos definidos en ella. Para ello se emplea addContentView (figura 3.21).

```
citiesView = new CitiesView(this);
citiesView.setVertex(vertex);
citiesView.setPath(path);
citiesView.setCell(cell);
citiesView.setThief(thief_pos);

addContentView(citiesView, new LayoutParams(LayoutParams.
    FILL_PARENT,LayoutParams.FILL_PARENT));
```

Figura 3.21. AddContentView

Todavía no se ha terminado con la construcción de la interfaz de la pantalla principal, ya que se han de añadir los dibujos de las cartas y el botón de dados. Como esto es un layout sencillo, se opta por emplear un XML (figura 3.22).

```
rl = (RelativeLayout) inflater.inflate(R.layout.new_game, null);

addContentView(rl, new LayoutParams(LayoutParams.FILL_PARENT,
LayoutParams.FILL_PARENT));

Button diceButton = (Button) this.findViewById(R.id.dice_button);
diceButton.setOnClickListener(this);
```

Figura 3.22. Carga de layout

Durante la implementación del juego se han llevado a cabo todas las variantes relativas a layouts: con archivo XML, desde el código y combinando las dos.

En el juego se han empleado distintos tipos de layout (relative, table, etc.) para poder explorar las distintas posibilidades que ofrecen.

Como se definió en el apartado de carpetas del proyecto 3.2.3, en la figura 3.10 vienen definidos todos los layouts XML del juego:

- **about.xml**: LinearLayout con un fondo en el que se añade un texto con las palabras *Acerca de* y un ScrollView que contiene el TextView con los datos a mostrar.

- **cards.xml**: RelativeLayout con imágenes de los recursos y textos que se modificarán en tiempo de ejecución. Los textos muestran que cantidad hay de cada material, y se actualiza con el método updateCards().

- **city_dialog.xml**: RelativeLayout formado con una imagen, un texto que se modificará en tiempo de ejecución y un botón. El texto se modificará a través del método showPopUp(int i): dependiendo del valor de i, se muestra un mensaje u otro.

- **first_turn.xml**: LinearLayout con dos textos (título y descripción) y un botón para aceptar y pasar a la siguiente pantalla.

- **instructions.xml**: LinearLayout con un texto y un ScrollView. El ScrollView permite que el aunque el texto que contenga sea muy grande, con una barra lateral se puede visualizar por completo.

- **main.xml**: TableLayout con tres botones para *Nuevo juego*, *Instrucciones* y *Acerca de*.

- **new_game_with_dices.xml**: RelativeLayout que incluye cards.xml.

- **new_game.xml**: RelativeLayout que además de cards.xml, incluye un botón para lanzar los dados.

Gráficos en pantalla

Una vez se han definido los layouts de las distintas pantallas, el paso siguiente es dibujar el tablero de la pantalla principal. Todos los recursos de imágenes están en la carpeta res/drawable/ y son objetos de tipo Drawable (una abstracción general para algo que se puede "pintar").

El primer paso es definir un fondo y esto se hace desde el constructor de ColonosView (figura 3.23):

```
setBackgroundDrawable(context.getResources().getDrawable(R.drawable.fondo));
```

Figura 3.23. Fondo

El método onDraw() de ColonosView es el que se encarga de dibujar todo el tablero (como se puede observar en la figura 3.24), esto incluye todas las celdas y las fichas con el valor de los dados que lleva asociado esa celda.

Figura 3.24. Tablero. Celdas y fichas.

Los recursos de Drawable (celdas y fichas), deben tener la información de dónde van a ser pintados. Para ello se utiliza el método setBounds al que se le pasa información de las coordenadas donde debe ser pintado y su tamaño. El método draw es el que pinta en esos límites establecidos por setBounds (figura 3.25).

```
cells[0].setBounds(w, h, (w + widthCell), (h + heightCell));
cells[0].draw(canvas);
```

Figura 3.25. setBounds y draw

ColonosView recorre la matriz de Drawables *cells* y va componiendo el tablero. Cada vez que pinta una de las celdas, comprueba si es desierto, y en caso negativo, pinta también el Drawable correspondiente de *chips*. En la celda de desierto, no va ninguna ficha ya que el desierto no va a producir ningún material.

CitiesView se encarga de pintar las construcciones que cada jugador va añadiendo. Cuando un jugador quiere añadir alguna construcción, Player se encarga de verificar si hay materiales disponibles. En caso afirmativo, pone a true la etiqueta correspondiente a la construcción de manera que, con esa etiqueta, en el método onTouch de CitiesView se buscará la posición más cercana a donde ha pulsado el jugador (a través de Player), se actualizará el campo occupation de la matriz de vertex, y a continuación se llama al método invalidate(). Lo que hace invalidate() es llamar a onDraw, de forma que se repinta la interfaz.

Las construcciones posibles son los poblados (figura 3.26), ciudades (figura 3.27) y caminos (figura 3.28). El juego permite como máximo tres jugadores, por lo que los colores disponibles para cada construcción son el rojo, el naranja y el azul. En el caso de los poblados y las ciudades, sólo hay un esquema de construcción en el que se cambia el color. Sin embargo para los caminos hay tres posibilidades que permiten que éste encaje en todos los lados de las celdas.

(a) Azul (b) Rojo (c) Naranja

Figura 3.26. Poblados

(a) Azul (b) Rojo (c) Naranja

Figura 3.27. Ciudades

(a) (b) (c)

Figura 3.28. Caminos

Diálogos

A lo largo del juego se usan distintos cuadros de diálogo [22]: pequeñas ventanas que aparecen delante de la Activity que esté en curso. La Activity que queda por debajo, pierde foco y el diálogo pasa a aceptar toda interacción del usuario. Los diálogos se usan normalmente como notificaciones que deben interrumpir al usuario y realizar alguna pequeña tarea que está en relación directa con la aplicación en progreso.

Dos tipos de diálogos se han implementado en el juego:

- **Dialog**: permite hacer un cuadro de diálogo customizable (figura 3.29). En primer lugar se define un archivo de layout .xml, que define que tiene una imagen, un texto y un botón (figura 3.30).

Figura 3.29. Mensaje de información. CustomDialog.

- **AlertDialog**: extiende de la clase Dialog y para el caso del juego, gestiona una lista de items seleccionables. Para crear un AlertDialog, se utiliza la subclase AlertDialog.Builder que con los métodos públicos de la clase, se definen todas las propiedades del AlertDialog. Para que el diálogo muestre la lista con las opciones a selec-

```
final Dialog dialog = new Dialog(this);
dialog.setContentView(R.layout.cit_dialog);
dialog.setCancelable(false);
text.setText(R.string.city\_dialog_text);

ImageView img = (ImageView) dialog.findViewById(R.id.
   ImageView01);
img.setImageResource(R.drawable.colonos\_dialog);

Button button = (Button) dialog.findViewById(R.id.Button01);
button.setOnClickListener(new OnClickListener() {

        public void onClick(View v) {
        dialog.dismiss();
                if (i == 4)
                finish(); }
                          });
        dialog.show();
```

Figura 3.30. Dialog

cionar (figura 3.31), se debe utilizar el método setItems(), cuya lista se encuentra almacenada en res/values/arrays.xml. Este tipo de diálogo se usa para seleccionar el número de jugadores (dos o tres), la construcción que el jugador quiere poner en el tablero (poblado, ciudad o camino), los cuatro materiales que quiere cambiar (madera, oveja, trigo, ladrillo o piedra) y por el que se quiere cambiar (madera, oveja, trigo, ladrillo o piedra).

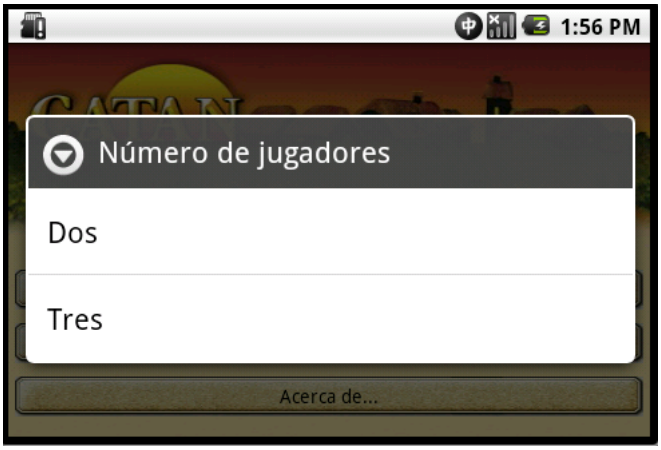


Figura 3.31. Lista de selección. AlertDialog.

Eventos de interfaz de usuario

El jugador interactúa con el sistema a través de la pantalla [23], por lo que se deben capturar desde el objeto de la View específica con la que interactúa el jugador. Existen dos opciones: que el jugador pulse alguno de los botones disponibles o que toque directamente la pantalla.

Para el caso de los botones, la clase que implementa el botón debe implementar OnClickListener (figura 3.32). El método onClick(View v), se llama automáticamente cuando se pulsa el botón.

```
View newGameButton = this.findViewById(R.id.new_game_button);
newGameButton.setOnClickListener(this);}
```

Figura 3.32. onClickListener

Dentro del método onClick, se define qué debe ocurrir cuando se ha pulsado el botón.

Los botones se han definido de forma tienen una imagen de fondo y encima se sitúa el texto que corresponda para cada caso.

Para el caso en que el usuario toque la pantalla, se llama al método onTouchEvent(MotionEvent event) de la View correspondiente. Para situar las construcciones en la pantalla, se toman las coordenadas una vez se separa el dedo de la pantalla. Esto se realiza de esta forma para que sólo se sitúe una construcción cada vez: de no tomar

las coordenadas en este punto, se toman muchas veces mientras el dedo siga tocando la pantalla (figura 3.33).

```
if (event.getAction() == MotionEvent.ACTION_DOWN) {
                        return true;
        } else if (event.getAction() == MotionEvent.ACTION_UP) {
                this.x = (int) event.getX();
                this.y = (int) event.getY();
        }
```

Figura 3.33. Captura de coordenadas

Menús

Android ofrece un sencillo framework para añadir menús [24]. Para el juego se ha implementado un menú de opciones que aparece cuando el usuario pulsa el botón MENU del smartphone. En lugar de instanciar el menú en el código de la aplicación, se define junto con sus items un recurso de menu XML: esto es una buena práctica que permite separar el contenido del menú, del código de la aplicación; también es más sencillo visualizar la estructura y contenido de un menú en un archivo XML.

El archivo XML se debe crear dentro del directorio del proyecto /res/menu, y debe tener los siguientes elementos:

- <**menu**>: define un objeto de la clase Menu, que es el contenedor de los items. Un elemento <menu>debe ser el nodo raíz para un archivo, y puede tener uno o más elementos <item>
- <**item**>: define un objeto de la clase MenuItem, que representa un único item en un menú.

Iconos

Con Android es posible definir un icono que irá asociado al ejecutable del juego. Este icono [25] es un gráfico que representa el juego en la pantalla principal del dispositivo y en el "Launcher". El usuario abre el Launcher tocando la pequeña flecha que se encuentra

en la parte inferior de la pantalla. Una vez está abierto, se muestran todos los iconos de todas las aplicaciones que están instaladas, de modo que el usuario puede ejecutar el juego seleccionando el icono. Se guarda en la carpeta res/drawable, con un formato .png (figura 3.34). Para aplicaciones o juegos que se instalen en distintos dispositivos, hay que tener en cuenta que se han de crear diferentes iconos para pantallas con baja, media y alta resolución. Esto asegura que el icono se mostrará adecuadamente en los distintos dispositivos en los que se pueda instalar la aplicación. Para el caso de Los Colonos de Catán, se ha diseñado un icono de alta resolución para el modelo HTC Legend.

Figura 3.34. Icono. Lanzamiento.

3.3.4. Núcleo del juego

En este apartado se explican detalles de la gestión del tablero, las posiciones cercanas y el ladrón.

Gestión del tablero

En el tablero se define mucha cantidad de información: cada celda producirá un determinado material, siempre y cuando los dados sumen el valor que se indica en ella y no esté el ladrón; en los vértices puede haber construcciones de los jugadores; y en los lados de las celdas puede haber caminos. Dado que el tablero está formado por celdas hexagonales, no se podía hacer su gestión de manera intuitiva con un sistema de coordenadas cartesianas. También no es sólo es necesario conocer las coordenadas de una celda, si no también que vértices y lados hay entre dos celdas y es necesario que sea posible calcular las coordenadas de cualquier celda, o vértice o lado, relativas a cualquier otra celda, vértice o lado. Por ello se optó por un sistema de coordenadas[26] en el que las

celdas se etiquetan utilizando dos ejes que forman un ángulo de 120º empezando por la parte izquierda y central del tablero. Uno de los ejes aumenta en dirección noreste y el otro en dirección sureste. Ya que se requieren muchos cálculos con el sistema de coordenadas, era necesario que la representación fuese compacta y permitiese cálculos con un número mínimo de cantidad de cómputos. Con estos objetivos en mente, las coordenadas se representan con dos números hexadecimales. La distancia a través del eje sureste se representa por el dígito más significativo, mientras que la del eje noreste con en el menos significativo (figura 3.35). Aunque este sistema permite representaciones limitadas en tamaño, para el caso de los colonos de catán es suficiente.

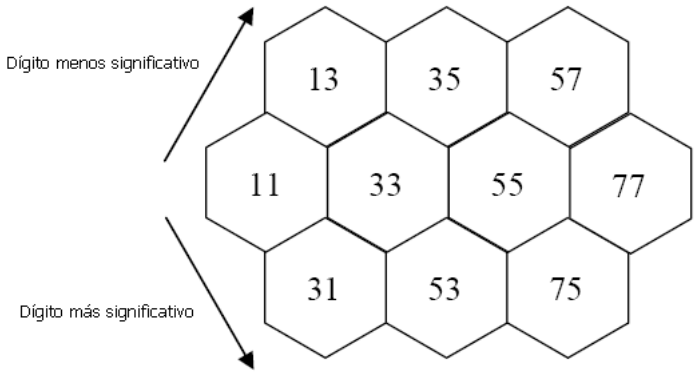


Figura 3.35. Celdas. Sistema de coordenadas.

Para las coordenadas de las celdas, el incremento a lo largo de cada eje es por dos. Por ejemplo, si se quiere obtener las coordenadas inmediamente al sureste de la celda 11, se incrementa 20. Incrementando en dos, se simplifica la tarea del cálculo de coordenadas de un lado o un vértice dadas las coordenadas de una celda. los lados se etiquetan de forman similar donde el dígito de la izquierda representa la distancia en la dirección sureste, y el dígito de la derecha la distancia en la dirección noreste (figura 3.36).

Los vértices también se etiquetan siguiendo el mismo método(figura 3.37).

Usando este esquema de coordenadas, es muy sencillo calcular las coordenadas de una celda, un lado o un vértice relativos a otra celda, lado o vértice. Esto resulta muy importante durante el juego. Por ejemplo, después de lanzar los dados, los jugadores reciben recursos si tienen un poblado o ciudad tocando la celda cuyo valor coincide con lo sacado en los dados. La forma de comprobar esto, es averiguar las coordenadas de la celda cuyo valor coincide con los dados y después calcular las coordenadas de los vértices

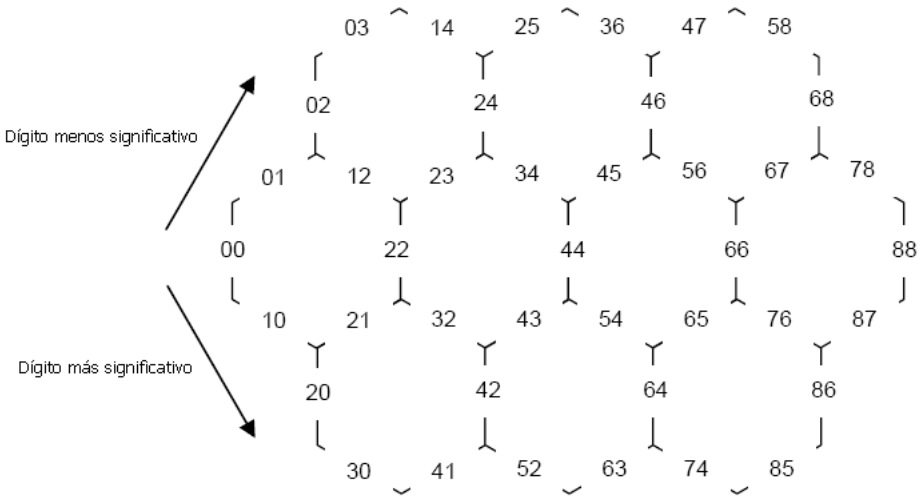


Figura 3.36. Lados. Sistema de coordenadas.

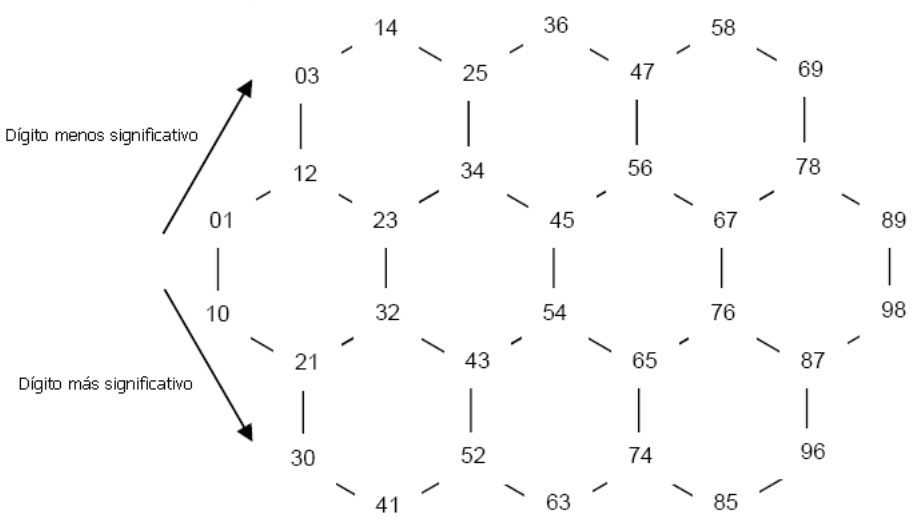


Figura 3.37. Vértices. Sistema de coordenadas.

adyacentes, para verificar entonces si en esos vértices alguno de los jugadores tiene un poblado o una ciudad. La figura 3.38 muestra como calcular las coordenadas de cualquier vértice o lado alrededor de una celda sumando o restando dígitos de izquierda a derecha a las coordenadas de la celda. Por ejemplo, para obtener las coordenadas del vértice más superior de una celda con coordenadas 11, simplemente se suma 0 al primer dígito y 1 al segundo obteniendo 12.

De forma similar las coordenadas de cualquier elemento se pueden calcular dadas las de otro. El Apendice A incluye un resumen de como se ha hecho para el tablero

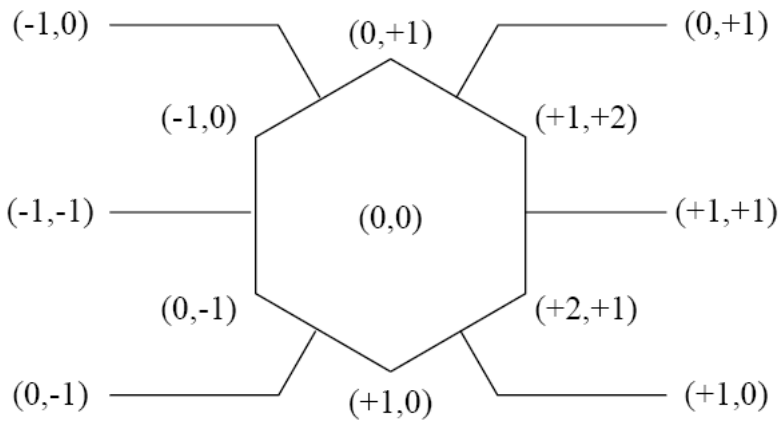


Figura 3.38. Coordenadas relativas. Sistema de coordenadas.

completo.

Posiciones cercanas

Una de las restricciones, es la que impone que los poblados y ciudades han de estar en los vértices de las celdas, y los caminos en los lados.

En el método onTouch() de CitiesView, se guardan las coordenadas del lugar donde se ha pulsado, pero estas coordenadas no pueden ser las que sirvan como referencia para pintar. Para ello se busca cuales son las coordenadas del vértice o lado más cercano. Para el caso de poblados o ciudades, el método utilizado para ello es nearestPosition(int x, int y) (figura 3.39).

```
    x_comp = vertex[i][j].getCoordinateX();
    if ((x < x_comp + radius) && (x > x_comp - radius)) {
            position[0] = i;
    }
y_comp1 = vertex[i][j].getCoordinateY();
    y_comp2 = vertex[i + 1][j + 1].getCoordinateY();
    if ((y < y_comp1 + radius) && (y > y_comp2 - radius)) {
            position[1] = i / 2;
    }
```

Figura 3.39. Calculo de la posición cercana

En la figura 3.40 se muestra un dibujo con el tratamiento que se hace para el cálculo de la posición más cercana. Las coordenadas de donde se ha pulsado son x e y, y con las que se compara x_comp e y_comp1 y_comp2. El int radius tiene un valor de 12, que es el margen que se toma para este cálculo. En el eje y se tienen dos coordenadas: en primer lugar se detecta en que "fila"se está trabajando. El recuadro rojo representa el area donde se compara: si el punto en el que se ha pulsado la pantalla se encuentra dentro de ese recuadro (x_comp - radius, x_comp +radius, y_comp1 + radius, y_comp2 - radius), las coordenadas en las que pintará el poblado o la ciudad son x_comp, y_comp2. La elección de y_comp1 o y_comp2, dependerá de la coordenada x elegida. Por fila (figura 3.41) se define aquella formada por las coordenadas a y b, o c y d de un conjunto de celdas.

Para calcular el camino más cercano se emplea nearestPath. La idea es similar a nearestPosition: se mira cual es la posición más cercana y se actúa en consecuencia.

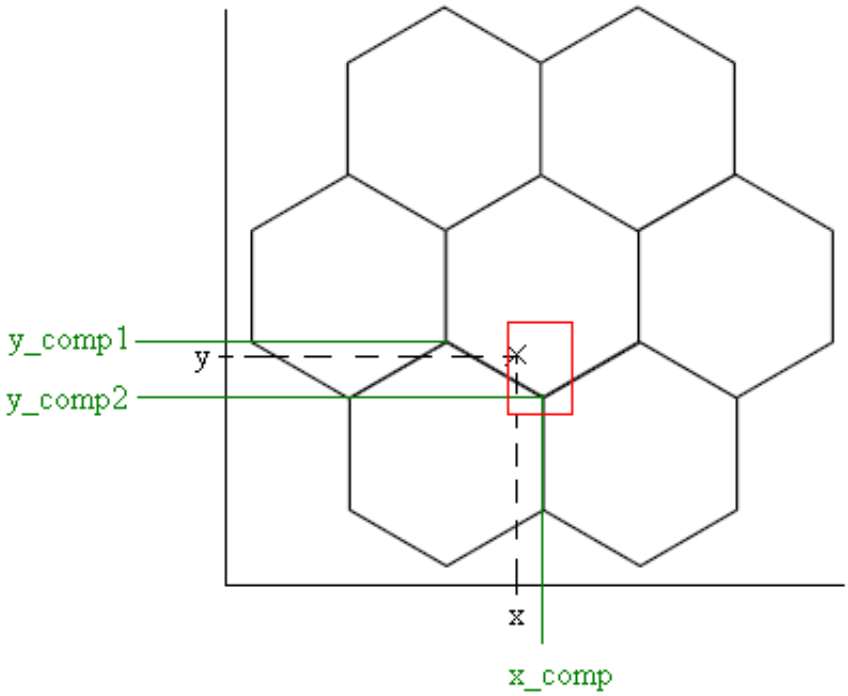


Figura 3.40. NearestPosition. Cálculo del vértice más cercano.

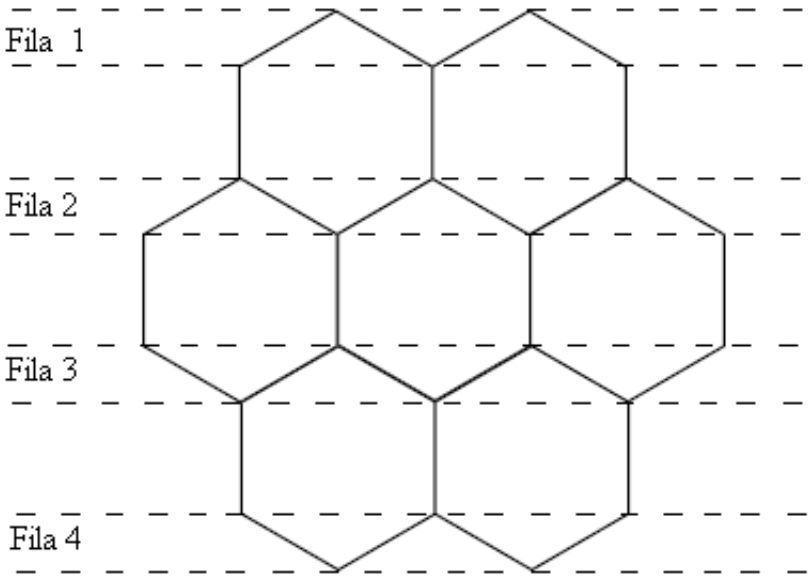


Figura 3.41. Filas

Además de esto también es necesario calcular la celda más cercana a las coordenadas que se han pulsado para cambiar el ladrón de sitio. Esto se hace con nearestCell, que en esta ocasión tiene cuatro coordenadas a comparar.

En el siguiente fragmento de código, se muestra una porción del tipo de comparaciones que se realizan (figura 3.42):

```
x_comp1 = vertex[i - 1][j].getCoordinateX();
y_comp1 = vertex[i - 1][j].getCoordinateY();
x_comp2 = vertex[i + 2][j + 1].getCoordinateX();
y_comp2 = vertex[i + 2][j + 1].getCoordinateY();
if ((x > x_comp1 + radius) && (x < x_comp2 - radius)&& (y <
   y_comp2) && (y > y_comp1)) {
        position[0] = i;
        position[1] = j;
}
```

Figura 3.42. Comparaciones

Si la zona donde se pulsa se encuentra dentro del recuadro rojo (figura 3.43), entonces el ladrón se situará en el centro de esa celda. Los límites de este recuadro vienen marcados por dos puntos, que son la esquina superior izquierda de la celda (x_comp1, y_comp1) y la esquina inferior derecha (x_comp2, y_comp2). a estos puntos se les aplica el mismo marge que en caso anterior (radius).

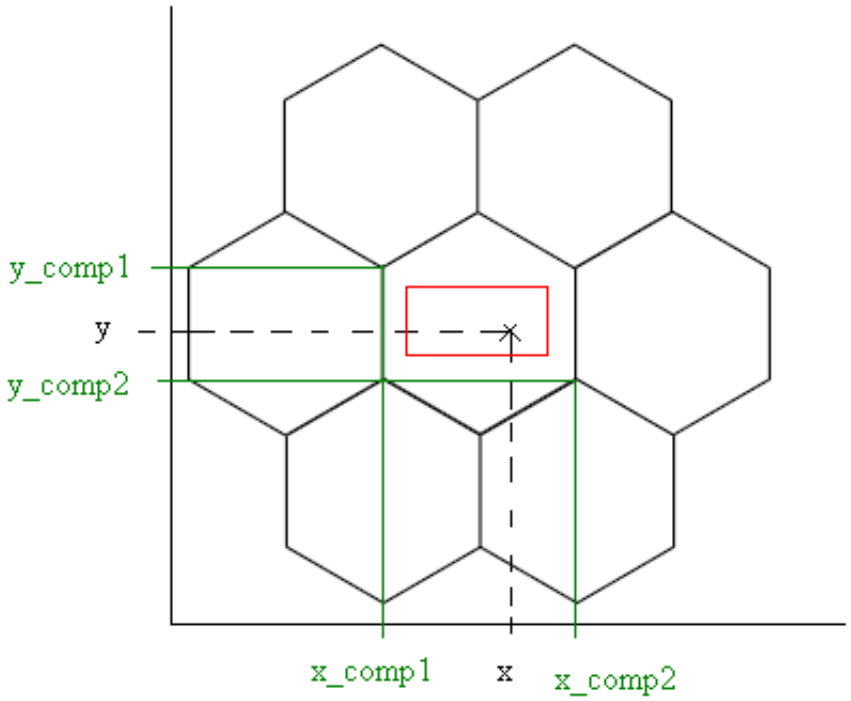


Figura 3.43. NearestCell

Ladrón

El ladrón se ha de mover cada vez que se lanza un dado y se saca un 7. En ese caso se lanza un diálogo (figura 3.44) que informa al jugador que puede mover el ladrón de sitio. Como se ha comentado, en la celda en la que esté el ladrón no se va a producir material. Esto se gestiona en la clase Player:

El material ha de ser diferente de 6 (desierto) y no debe estar el ladrón (figura 3.45). Si se cumplen las dos condiciones, en los vertices en los que haya casas, se comprobará a que jugador pertenecen, y se actualizará el número de recursos (cartas) dependiendo del material que se tenga en esa celda.

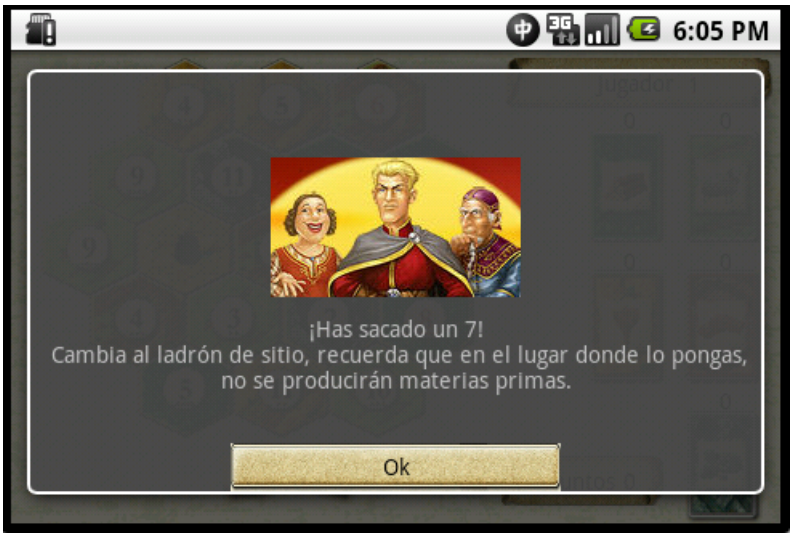


Figura 3.44. Ladrón. Cuadro de diálogo.

```
is_thief = cell[i][j].getThief();
if (material != 6 && !is_thief) {
        // Recorro todos los vértices para ver si hay casas
}
```

Figura 3.45. Ladrón

3.3.5. Otros

En esta sección se definen detalles adicionales de la implementación que no se incluyen en ninguno de los apartados anteriores.

Sonidos

Para la gestión de sonidos [27], se usa la clase MediaPlayer de android, que controla ficheros y streams de audio y video.

A lo largo del juego, tres sonidos van asociados a diferentes estados:

- Al comienzo del juego un fichero con **sonidos de bosque** se ejecutará en un bucle (en Menu) y este sonido finaliza cuando se lanza el hilo de la siguiente clase
- Cuando un jugador no tiene los materiales suficientes, se muestra junto con el cuadro de diálogo un **sonido de error**
- Se oye **un aplauso** cuando el jugador gana la partida
- También hay un **sonido de dados** cuando éstos se lanzan.

Para que cualquiera de los sonidos se oiga permanentemente, se ha de implementar un bucle, y esto se consigue con la instrucción setLooping (figura 3.46). Si se omite esta instrucción, el sonido sólo se oirá una vez.

```
MediaPlayer mp = MediaPlayer.create(this, R.raw.spring\_weather);
mp.setLooping(true);
mp.start();
```

Figura 3.46. Sonidos

Strings y arrays

Para optimizar los recursos, es posible guardar valores de arrays y strings en los archivos xml contenidos en la carpeta res/values (figuras 3.47 3.48).

Para poder tener acceso a los strings y arrays, se hace uso de la etiqueta name (number_of_players en este caso), de modo que se toma y usa el valor *Número de jugadores* (figura 3.49).

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <array name="Number_of_players">
                <item>@string/two_label</item>
                <item>@string/three_label</item>
        </array>
//Resto de arrays
</resources>
```

Figura 3.47. Arrays

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string name="app_name">Los Colonos de Catán</string>
        <string name="number_of_players">Número de jugadores</
           string>
        <string name="colonos_title">Colonos</string>
        <string name="two_label">Dos</string>
        <string name="city_label">Poblado (ladrillo, madera, trigo
            y oveja)</string>
    //Resto de strings
</resources>
```

Figura 3.48. Strings

```
<string name="number_of_players">Número de jugadores</string>
R.string.number_of_players
```

Figura 3.49. Acceso

Capítulo 4

Conclusiones

Finalizado el proyecto, se analiza el resultado para comprobar si se han cumplido con las expectativas. Se puede afirmar que los objetivos propuestos se han logrado.

4.1. Implementación de un juego

El primero de los objetivos, implementar un juego bajo la plataforma Android, se ha completado satisfactoriamente. Se ha creado una aplicación que funciona según los requisitos planteados en el smartphone HTC Legend.

4.2. Posibilidades de Android

El segundo de los objetivos buscaba explorar las opciones que ofrecía Android. También se ha cumplido con este objetivo, ya que se han implementado distintas funcionalidades para aprender su uso y aplicación:

- **Botones**: se ha aprendido a incluirnos en la pantalla, capturar la pulsación, realizar alguna acción una vez pulsados y optimizarlos visualmente mediante el uso de imágenes.
- **Menús**: se han implementado menús de opciones que realizan distintas acciones,

desde cambiar de Activity o mostrar cuadros de diálogo.

- **Cuadros de diálogo**: se han utilizado para mostrar tanto mensajes informativos o de error, como listas para hacer selecciones.

- **Activities**: se ha aprendido que cada pantalla a mostrar requiere de una Activity, y cómo se ha de hacer la gestión entre las distintas Activities dependiendo del resultado que se quiera obtener.

- **Imágenes**: se ha ahondado en la forma de gestionar el uso de imágenes, instanciándolas, modificándolas y refrescando sus vistas.

- **Sonidos**: se han incluido sonidos tanto en bucle (para que se oigan de forma permanente en una pantalla determinada), como puntuales (para que sólo se oigan al mostrarse un mensaje).

Gracias a que Android basa su sistema operativo en código libre, un gran número de programadores desarrollan aplicaciones de manera que es más sencillo encontrar documentación y ejemplos de código. También esto hace que a la hora de gestionar los problemas o errores, la búsqueda de soluciones es más rápida que en sistemas operativos con código cerrado o propietario.

4.3. Futuros desarrollos

En el caso de que futuros estudiantes deseen implementar una aplicación o juego en la misma plataforma, este proyecto les sirve de base o guía para dar esos primeros pasos.

También se ha cumplido el objetivo del que el juego sirviese como base a futuros desarrolladores de Android. El juego implementado cumple unas funcionalidades básicas que lo hacen jugable, sin embargo da pie a que se realicen ampliaciones y mejoras en el mismo. Estos puntos se tratan en el capítulo de Líneas Futuras 5.

4.4. Otras conclusiones

Adicionalmente a los objetivos que se habían marcado, dado que en Android se usan muchos recursos con formato **XML**, se ha podido profundizar en este lenguaje y en las capacidades y funcionalidades que ofrece.

También se han revisado las posibilidades que ofrece el lenguaje **UML** para la fase de diseño del proyecto. Gracias al mismo se puede hacer un diseño y exhaustivo de clases, flujos, etc, que hacen la fase de implementación más eficiente y sencilla.

A nivel personal, es muy satisfactorio el poder desarrollar por primera vez un juego, y en una plataforma en la que no se tenía ninguna base. Una vez conocida la plataforma, y gracias a los conocimientos adquiridos, se espera seguir desarrollando aplicaciones y juegos, ya que resulta muy gratificante una vez se llega al resultado final.

Capítulo 5

Líneas futuras

Puesto que había una limitación en tiempo para la realización del proyecto, no ha sido posible implementar todas las funcionalidades que se hubieran deseado. Es por ello que en este capítulo se muestran algunas de esas posibles mejoras para realizar en un futuro.

Con estas mejoras, se conseguiría un juego más completo y con más funcionalidades que permitiría al usuario disfrutar más del mismo.

5.1. Multijugador en varios terminales

El juego implementado es para un mínimo de dos personas que juegan pasándose el terminal. Otra de las posibles mejoras sería la de modificar el juego de manera que cada jugador pudiese usar su propio smartphone.

Para conseguir esto se ha de permitir la comunicación entre los terminales de los distintos jugadores y esto se puede conseguir con el uso del **Bluetooth**. Android permite a un dispositivo intercambiar datos sin cables con otros dispositivos Bluetooth. Para ello se emplearía el API Bluetooth para las cuatro tareas principales: configurar Bluetooth, encontrar qué dispositivos hay disponibles en el área de alcance, conectarlos y transferir datos entre ellos.

5.2. Versión ampliada del juego

La versión que se ha desarrollado está simplificada con respecto al juego original, por lo que las normas no son exactamente las mismas.

Otra posible mejora sería la de replicar exactamente el juego original. Para ello se han de incluir aspectos como cartas especiales al juego, puntos para el jugador con el camino más largo, posibilidad de negociar con el resto de jugador o inclusión de puertos en los que poder intercambiar materias primas.

5.3. Elección de niveles

Como el juego de tablero original tuvo mucho éxito, se sacaron al mercado nuevas versiones del juego. Estas versiones también llamadas ampliaciones o extensiones, se podrían tener en cuenta para las líneas futuras.

Estas extensiones incluyen la ampliación 5 y 6 jugadores, los Navegantes de Catán (el jugador puede crear los mapas a su antojo en lugar de utilizar el hexágono), Ciudades y Caballeros de Catán (dispone de más opciones de desarrollo) o Mercaderes y Bárbaros (añade gran variedad de misiones y modos de juego).

Una mejora a realizar en el juego sería que fuese posible elegir distintos niveles que replicasen los juegos que hay en le mercado.

5.4. Otras mejoras

Además de estas mejoras, se pueden realizar otras pequeñas modificaciones que harían del juego un entretenimiento más cómodo de usar.

- Pedir confirmación de donde se ha pulsado: aunque en el juego desarrollado sólo se permite construir en los lugares habilitados para ello, una vez el jugador ha situado la construcción en un determinado punto, ya no puede cambiarla. Una buena práctica sería la de pedir confirmación al jugador de si la construcción se

va a poner en el punto que realmente desea (dada la limitación de tamaño de la pantalla, no es relativamente complicado el pulsar en un punto cercano pero no exacto al deseado).

- Zoom del tablero: de nuevo, por la limitación de tamaño de la pantalla, la posibilidad de hacer zoom permitiría un uso más sencillo del juego. Si la superficie de pulsación es muy amplia, gracias al zoom no existiría la posibilidad de tener confusiones a la hora de situar construcciones sobre el tablero. Estas confusiones incluyen tanto el que el juego de por no válida una posición que está fuera de límites, tanto como se sitúe en posiciones adyacentes.

- Inteligencia artificial: si se implementa la capacidad de que el juego sea capaz de responder a las acciones de un jugador, se permitía la posibilidad de que un único usuario jugase una partida contra la máquina.

Apéndices

Anexo A

Planificación y presupuesto

Antes de comenzar el proyecto, es necesario realizar una planificación que permita dividirlo en distintas fases y asociar a cada una de ellas tanto recursos materiales como personales, costes y plazos estimados. Gracias a esta planificación en etapas del proyecto, se consigue hacer un seguimiento del mismo y así comprobar si se están cumpliendo los fechas comprometidas en un primer momento.

En el anexo, se muestran el ciclo de vida de la realización de un videojuego, el ciclo de vida de *Los Colonos de Catán* y un diagrama de planificación inicial.

A.1. Ciclo de vida de un juego comercial

En este apartado se definen las distintas etapas que componen el ciclo de vida de un juego comercial. Estas etapas no se han seguido exactamente en el proyecto, pero sirven de ayuda y guía para la planificación de cualquier juego.

- **Concepto**: durante esta primera fase, se define el concepto del juego a partir de una idea de origen o un "brainstorming", creándose la propuesta del juego y el arte conceptual. Esto servirá como una primera base que define la historia del juego. Además se incluyen en esta fase, la definición de diferentes características del juego como ambientación, género, plataformas de desarrollo, cronograma estimado,

presupuesto y análisis de riesgos. Diseñadores, analistas y artistas conceptuales, participan en esta fase.

- **Pre-producción**: en la fase de pre-producción se estudia si es viable realizar el juego, definiendo un estudio completo con la mecánica del mismo, niveles, pantallas, personajes, etc. Se ha de tener también un calendario aproximado para la realización de las tareas del proyecto.
- **Producción**: es la parte más importante del proyecto ya que es cuando se construye e implementa el juego. Cuando se llega a un nivel suficiente para poder probar, los testeadores se encargan junto con desarrolladores de corregir errores, realizar avances y añadir nuevas funcionalidades. Consta de varias fases como son el diseño, la programación, la creación de niveles, el diseño artístico, la producción de audio o el testeo.
- **Alfa**: durante la fase alfa, el juego ya es operativo por completo (motor, interfaz de usuario, etc), pero todavía pueden quedar detalles o errores por depurar.
- **Beta**: fase cuya finalidad es dejar el juego libre de errores y fallos. El incremento de características se ha finalizado, y solo quedan por corregir los errores.
- **Congelación del código**: se congela la versión del juego de forma que ya no se apliquen más cambios. En este punto queda pendiente de la aprobación.
- **Liberación**: una vez se ha aprobado, se libera a los medios de distribución para acercarlo al público.
- **Parches**: cuando los usuarios encuentran fallos o errores en el juego que deben ser reparados, se lanzan parches para solucionar el problema.
- **Actualizaciones**: pueden ir desde pequeños detalles, hasta niveles o mundos completamente nuevos e innovadores.

A.2. Planificación y etapas del proyecto

Antes de planificar, es necesario elegir un modelo de ciclo de vida que se seguirá durante el proyecto. Después de evaluar las distintas posibilidades, se ha optado por el modelo de desarrollo evolutivo (figura A.1).

El modelo evolutivo tiene como objetivo principal construir un prototipo de una forma estructurada, de manera que se pueda ir refinando y reconstruyendo. Esto permite al equipo de desarrollo añadir características o hacer cambios, que no se concibieron durante la fase de requerimientos y diseño.

Gracias a este modelo, los desarrolladores se pueden centrar en una parte del sistema que comprendan, en lugar de tener que abarcar el sistema completo.

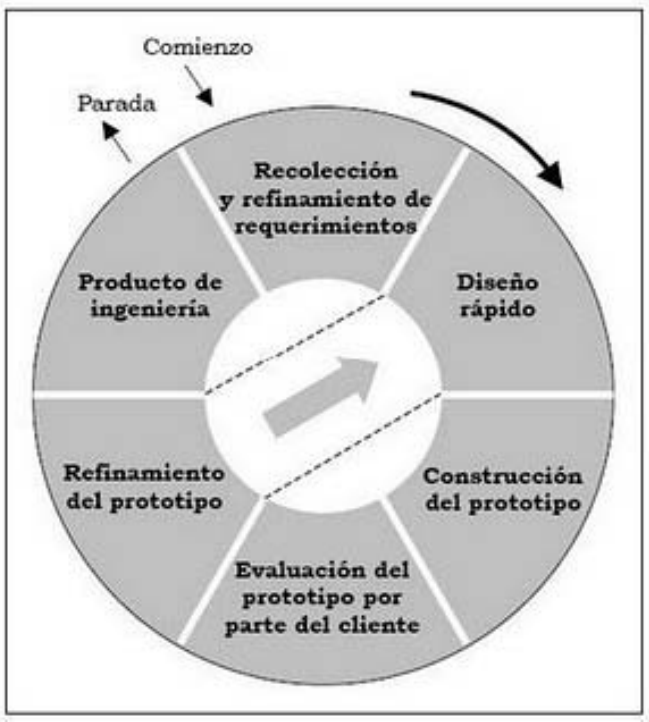


Figura A.1. Modelo evolutivo. Diagrama.

Con este modelo es posible el centrarse en módulos, de forma que el juego sea operativo y jugable desde el principio, y se vayan añadiendo distintas funcionalidades y características.

El proyecto consta de varios prototipos o componentes, y en cada uno de ellos se ha aplicado el modelo evolutivo. Como se muestra en la figura A.1, al comienzo del desarrollo se recogen y refinan los requerimientos, se realiza un diseño rápido para construir el prototipo, se evalúa y se refina.

Los componentes son:

- **Formación previa**: estudio de qué elementos serán necesarios para la realización del proyecto. Para el desarrollo del juego, es necesario tener conocimientos de Android, por lo que esta fase no aplica en caso de haber implementado con anterioridad alguna aplicación con Android.

- **Desarrollo en la documentación y memoria**: a medida que se van realizando o modificando prototipos, se recoge su documentación asociada (análisis, diseño, implementación, evaluación y problemas encontrados), para ir componiendo así este documento final.

- **Estructura básica del juego**: proporcionará la funcionalidad básica para que el juego arranque y sirva de base para el resto de componentes. Incluye la navegación entre pantallas, eventos simples (botones) e interrupción y salida del juego.

- **Interfaz de usuario**: en este apartado se define el diseño gráfico de pantallas, los cuadros de diálogo, los menús y los iconos.

- **Núcleo del juego**: en este prototipo se diseña e implementa todo lo relacionado con la gestión del tablero, la detección de las posiciones más cercanas y los movimientos del ladrón.

- **Otros**: para esta parte del desarrollo, se tienen en cuenta los sonidos y la gestión del texto.

- **Evaluación y pruebas generales**: una vez se tienen completos todos los prototipos, en el de evaluación y pruebas generales se comprueba el correcto funcionamiento del juego.

Estos prototipos se muestran en detalle gracias a los diagramas de planificación, que además estiman los tiempos y los recursos necesarios.

1. Formación previa	**30 días**
1.1 Estudio Android	25 días
1.2 Estudio Eclipse	5 días
2. Desarrollo de la documentación y la memoria	**150 días**
3. Fase 1 - Estructura básica del juego	**18 días**
3.1 Requisitos fase 1	5 días
3.1.1 Aspecto básico y dimensiones	4 días
3.1.2 Navegación entre las distintas pantallas	5 días
3.1.3 Eventos básicos	5 días
3.1.4 Interrupción y salida del juego	5 días
3.2 Diseño	4 días
3.3 Implementación	8 días
3.4 Evaluación	1 día
4. Fase 2 - Interfaz de usuario	**30 días**
4.1 Requisitos fase 2	6 días
4.1.1 Diseño gráfico de pantallas	3 días
4.1.2 Cuadros de diálogo	2 días
4.1.3 Menús	3 días
4.1.4 Icono	1 días
4.2 Diseño	6 días
4.3 Implementación	15 días
4.4 Evaluación	3 días

Tabla A.1. Tareas del proyecto. Planificación.

5. Fase 3 - Núcleo del juego	**25 días**
5.1 Requisitos fase 3	8 días
5.1.1 Gestión del tablero	6 días
5.1.2 Posiciones cercanas	6 días
5.1.3 Ladrón	2 días
5.2 Diseño	7 días
5.3 Implementación	9 días
5.4 Evaluación	1 día
6. Fase 4 - Otros	**15 días**
6.1 Requisitos fase 4	5 días
6.1.1 Sonidos	3 días
6.1.2 Gestión de texto	2 días
6.2 Diseño	3 días
6.3 Implementación	5 días
6.4 Evaluación	2 días
7. Evaluación y pruebas finales	**5 días**

Tabla A.2. Tareas del proyecto II. Planificación.

Todas las tareas de las tablas A.1 y A.2, deben llevar asociado un recurso, o bien material o bien personal. Para los proyectos en los que se dispone de varios recursos, se posibilita el llevar a cabo varias tareas en paralelo, optimizando de este modo el tiempo invertido. Además de las tareas señaladas, es importante planificar reuniones de seguimiento, mensuales o semanales dependiendo de la importancia fase.

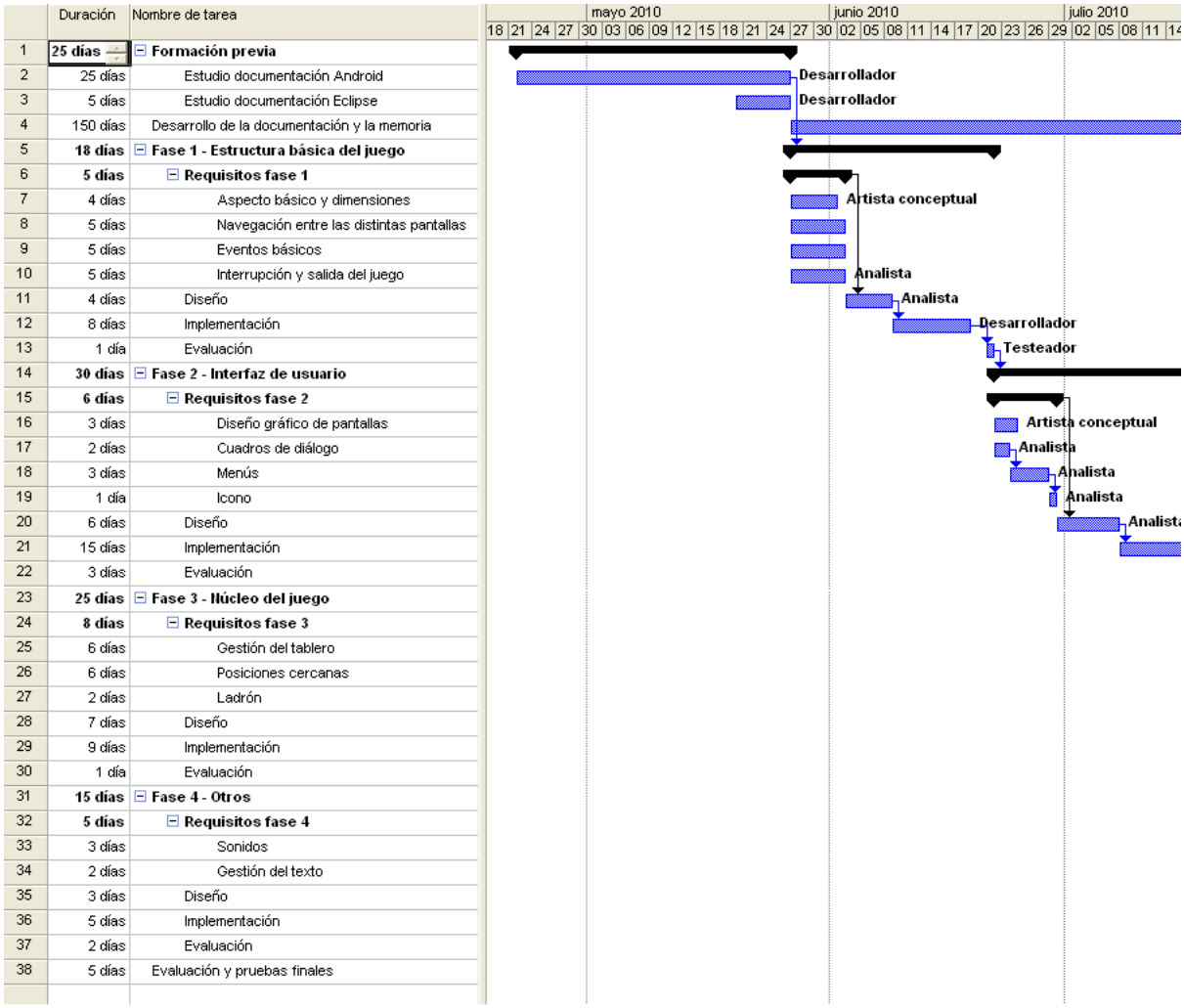
La figura A.2 muestra la planificación con las tareas, fechas de inicio y fin, y los recursos asignados a cada una. Aunque se han definido distintos recursos humanos (analista, desarrollador, artista conceptual o testeador), a efectos prácticos sólo hay un único recurso, la autora de este proyecto.

Las figuras A.3, A.4 y A.5, muestran el detalle de la duración de las tareas, así como su evolución en el tiempo. La planificación del proyecto ha establecido una duración de 8 meses, desde Abril de 2010 hasta Diciembre de 2010, y sábados y domingos no se han tenido en cuenta.

Diagrama de Gantt

	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
1	**Formación previa**	**25 días**	**jue 22/04/10**	**mié 26/05/10**	**Desarrollador**
2	Estudio documentación Android	25 días	jue 22/04/10	mié 26/05/10	Desarrollador
3	Estudio documentación Eclipse	5 días	jue 20/05/10	mié 26/05/10	Desarrollador
4	Desarrollo de la documentación y la memoria	150 días	jue 27/05/10	mié 22/12/10	Desarrollador
5	**Fase 1 - Estructura básica del juego**	**18 días**	**jue 27/05/10**	**lun 21/06/10**	
6	**Requisitos fase 1**	**5 días**	**jue 27/05/10**	**mié 02/06/10**	
7	Aspecto básico y dimensiones	4 días	jue 27/05/10	mar 01/06/10	Artista conceptual
8	Navegación entre las distintas pantallas	5 días	jue 27/05/10	mié 02/06/10	
9	Eventos básicos	5 días	jue 27/05/10	mié 02/06/10	
10	Interrupción y salida del juego	5 días	jue 27/05/10	mié 02/06/10	Analista
11	Diseño	4 días	jue 03/06/10	mar 08/06/10	Analista
12	Implementación	8 días	mié 09/06/10	vie 18/06/10	Desarrollador
13	Evaluación	1 día	lun 21/06/10	lun 21/06/10	Testeador
14	**Fase 2 - Interfaz de usuario**	**30 días**	**mar 22/06/10**	**lun 02/08/10**	
15	**Requisitos fase 2**	**6 días**	**mar 22/06/10**	**mar 29/06/10**	
16	Diseño gráfico de pantallas	3 días	mar 22/06/10	jue 24/06/10	Artista conceptual
17	Cuadros de diálogo	2 días	mar 22/06/10	mié 23/06/10	Analista
18	Menús	3 días	jue 24/06/10	lun 28/06/10	Analista
19	Icono	1 día	mar 29/06/10	mar 29/06/10	Analista
20	Diseño	6 días	mié 30/06/10	mié 07/07/10	Analista
21	Implementación	15 días	jue 08/07/10	mié 28/07/10	Desarrollador
22	Evaluación	3 días	jue 29/07/10	lun 02/08/10	Testeador
23	**Fase 3 - Núcleo del juego**	**25 días**	**mar 03/08/10**	**lun 06/09/10**	
24	**Requisitos fase 3**	**8 días**	**mar 03/08/10**	**jue 12/08/10**	
25	Gestión del tablero	6 días	mar 03/08/10	mar 10/08/10	
26	Posiciones cercanas	6 días	mar 03/08/10	mar 10/08/10	Analista
27	Ladrón	2 días	mié 11/08/10	jue 12/08/10	Analista
28	Diseño	7 días	vie 13/08/10	lun 23/08/10	Analista
29	Implementación	9 días	mar 24/08/10	vie 03/09/10	Desarrollador
30	Evaluación	1 día	lun 06/09/10	lun 06/09/10	Testeador
31	**Fase 4 - Otros**	**15 días**	**mar 07/09/10**	**lun 27/09/10**	
32	**Requisitos fase 4**	**5 días**	**mar 07/09/10**	**lun 13/09/10**	
33	Sonidos	3 días	mar 07/09/10	jue 09/09/10	Analista
34	Gestión del texto	2 días	vie 10/09/10	lun 13/09/10	Analista
35	Diseño	3 días	mar 14/09/10	jue 16/09/10	Analista
36	Implementación	5 días	vie 17/09/10	jue 23/09/10	Desarrollador
37	Evaluación	2 días	vie 24/09/10	lun 27/09/10	Testador
38	Evaluación y pruebas finales	5 días	mar 28/09/10	lun 04/10/10	Desarrollador;Testeador

Figura A.2. Diagrama de Gantt. Tareas.



	Duración	Nombre de tarea
1	**25 días**	**Formación previa**
2	25 días	Estudio documentación Android
3	5 días	Estudio documentación Eclipse
4	150 días	Desarrollo de la documentación y la memoria
5	**18 días**	**Fase 1 - Estructura básica del juego**
6	**5 días**	**Requisitos fase 1**
7	4 días	Aspecto básico y dimensiones
8	5 días	Navegación entre las distintas pantallas
9	5 días	Eventos básicos
10	5 días	Interrupción y salida del juego
11	4 días	Diseño
12	8 días	Implementación
13	1 día	Evaluación
14	**30 días**	**Fase 2 - Interfaz de usuario**
15	**6 días**	**Requisitos fase 2**
16	3 días	Diseño gráfico de pantallas
17	2 días	Cuadros de diálogo
18	3 días	Menús
19	1 día	Icono
20	6 días	Diseño
21	15 días	Implementación
22	3 días	Evaluación
23	**25 días**	**Fase 3 - Húcleo del juego**
24	**8 días**	**Requisitos fase 3**
25	6 días	Gestión del tablero
26	6 días	Posiciones cercanas
27	2 días	Ladrón
28	7 días	Diseño
29	9 días	Implementación
30	1 día	Evaluación
31	**15 días**	**Fase 4 - Otros**
32	**5 días**	**Requisitos fase 4**
33	3 días	Sonidos
34	2 días	Gestión del texto
35	3 días	Diseño
36	5 días	Implementación
37	2 días	Evaluación
38	5 días	Evaluación y pruebas finales

Figura A.3. Diagrama de Gantt. Tareas.

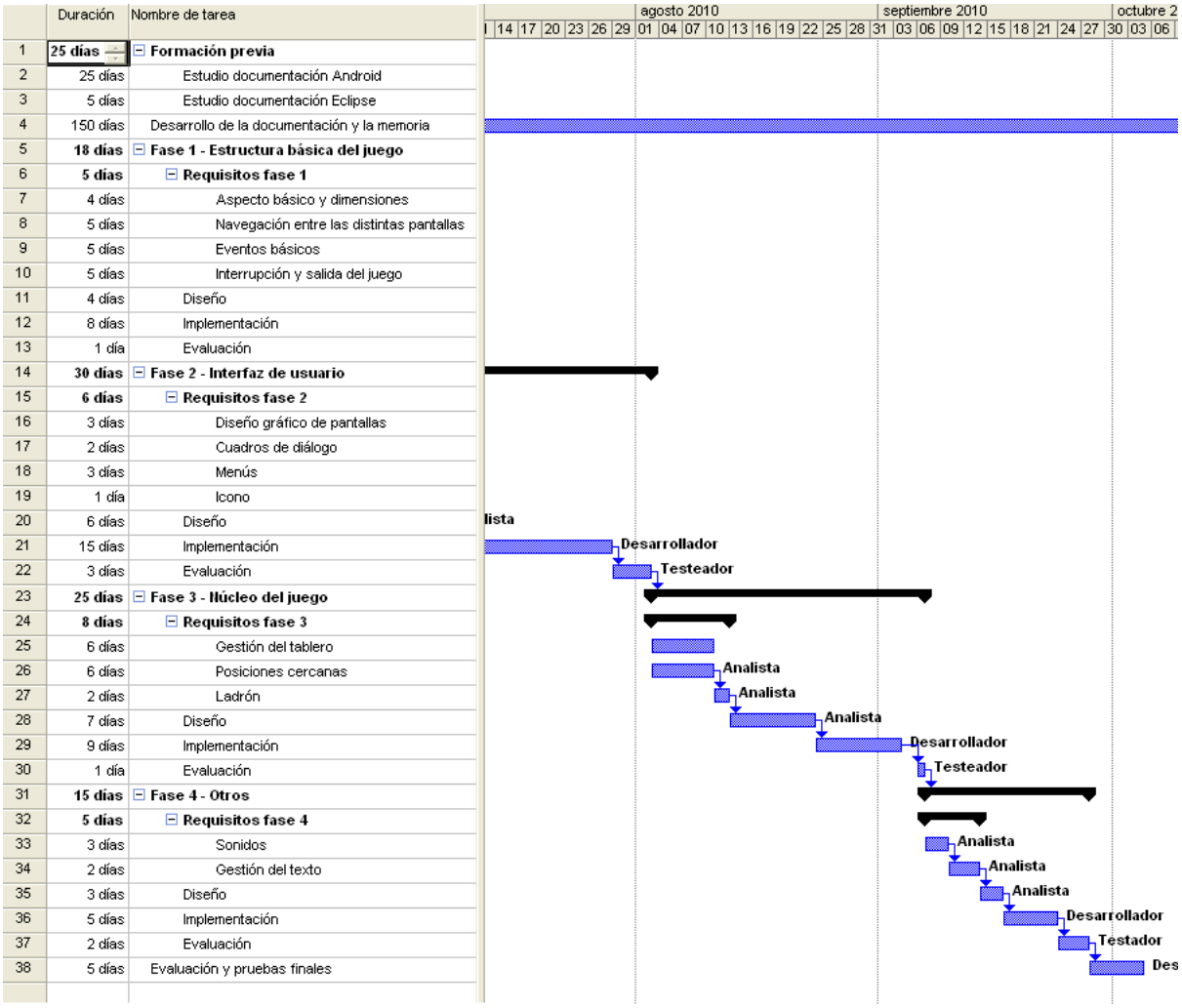
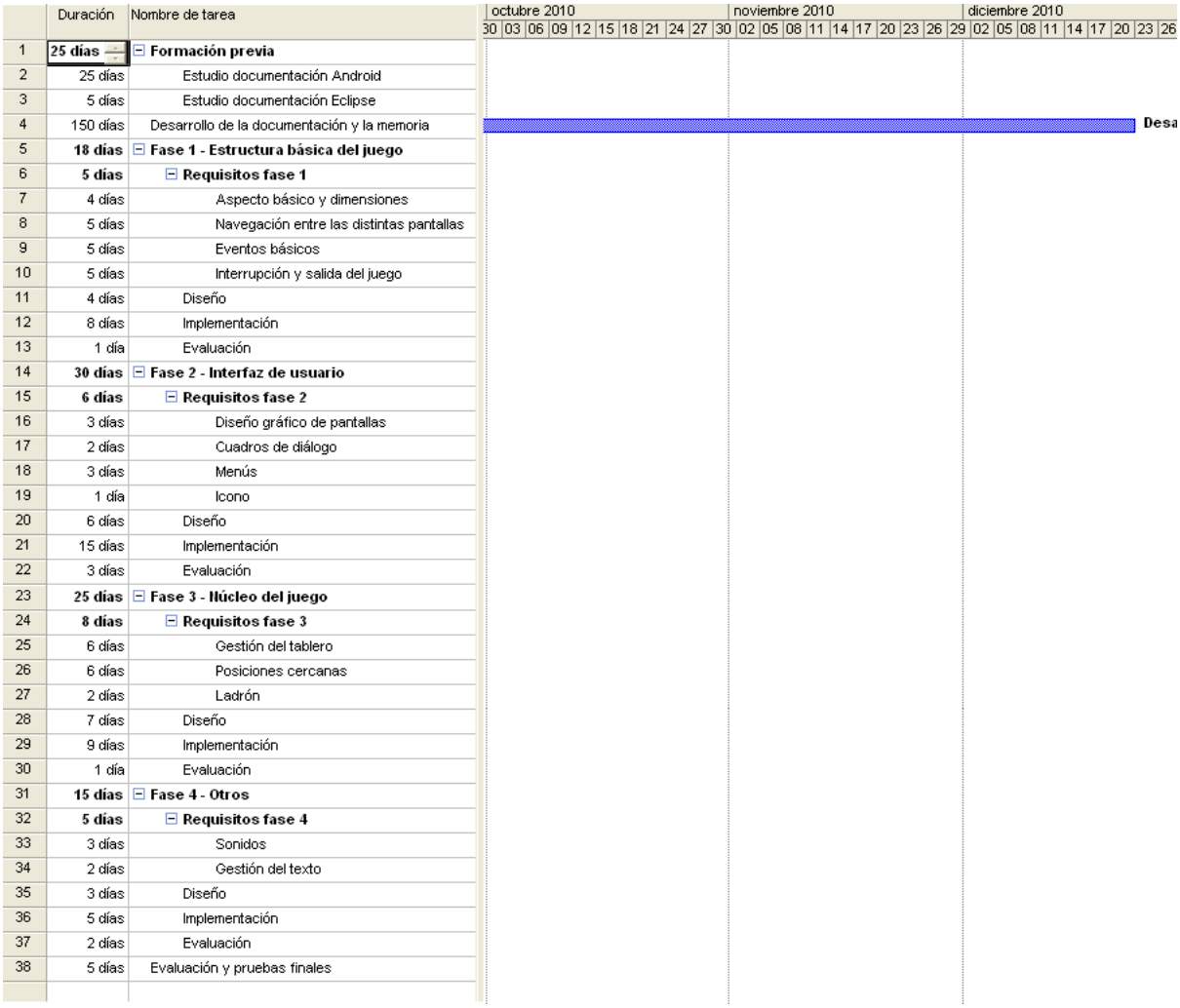


Figura A.4. Diagrama de Gantt. Tareas.



	Duración	Nombre de tarea
1	**25 días**	**Formación previa**
2	25 días	Estudio documentación Android
3	5 días	Estudio documentación Eclipse
4	150 días	Desarrollo de la documentación y la memoria
5	**18 días**	**Fase 1 - Estructura básica del juego**
6	**5 días**	**Requisitos fase 1**
7	4 días	Aspecto básico y dimensiones
8	5 días	Navegación entre las distintas pantallas
9	5 días	Eventos básicos
10	5 días	Interrupción y salida del juego
11	4 días	Diseño
12	8 días	Implementación
13	1 día	Evaluación
14	**30 días**	**Fase 2 - Interfaz de usuario**
15	**6 días**	**Requisitos fase 2**
16	3 días	Diseño gráfico de pantallas
17	2 días	Cuadros de diálogo
18	3 días	Menús
19	1 día	Icono
20	6 días	Diseño
21	15 días	Implementación
22	3 días	Evaluación
23	**25 días**	**Fase 3 - Húcleo del juego**
24	**8 días**	**Requisitos fase 3**
25	6 días	Gestión del tablero
26	6 días	Posiciones cercanas
27	2 días	Ladrón
28	7 días	Diseño
29	9 días	Implementación
30	1 día	Evaluación
31	**15 días**	**Fase 4 - Otros**
32	**5 días**	**Requisitos fase 4**
33	3 días	Sonidos
34	2 días	Gestión del texto
35	3 días	Diseño
36	5 días	Implementación
37	2 días	Evaluación
38	5 días	Evaluación y pruebas finales

Figura A.5. Diagrama de Gantt. Tareas.

A.3. Presupuesto del proyecto

A continuación se detallan los costes asociados al desarrollo de este juego, incluyendo costes de personal y gastos de material y equipos.

El coste total del proyecto es de 17.664 euros, de los cuales la cantidad mayor es la que implica la contratación de personal (14.500 euros). También suman costes al proyecto equipos necesarios para la fase de desarrollo y pruebas (un ordenador para la programación y el diseño gráfico de interfaces, y un smartphone HTC Legend para las pruebas y evaluaciones finales). Se incluye además el software de Adobe Photoshop para los diseños de las interfaces. Para las reuniones mensuales, también es necesario tener en cuenta los desplazamientos que se incluyen en Viajes.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Cristina Fernández Jiménez
2.- Departamento:
Departamento de Informática
3.- Descripción del Proyecto:
- Titulo: Diseño e implementación de un juego para smartphones con Android: Los Colonos de Catán
- Duración (meses): 8
Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):
Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación [a] (hombres mes)	Coste hombre mes	Coste (Euro)	Firma de conformidad
Peralta Donate, Juan		Ingeniero Senior	1	2.500,00	2.500,00	
Fernández Jiménez, Cristina		Ingeniero	8	1.500,00	12.000,00	
					0,00	
					0,00	
		Hombres mes 9		Total	14.500,00	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Smartphone HTC Legend	49,00	100	8	60	6,53
Ordenador programación	1.000,00	100	8	60	133,33
		100		60	0,00
		100		60	0,00
		100		60	0,00
					0,00
				Total	139,87

Figura A.6. Presupuesto del proyecto I. Detalle.

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} xCxD$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Adobe photoshop	Adobe	50,00
Viajes		30,00
	Total	80,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	14.500
Amortización	140
Subcontratación de tareas	0
Costes de funcionamiento	80
Costes Indirectos	2.944
Total	17.664

Figura A.7. Presupuesto del proyecto II. Detalle.

A.4. Comercialización del juego

Una vez se tiene una planificación y un presupuesto, es necesario realizar un estudio de mercado para evaluar si el producto será rentable.

Dado que el juego se ha desarrollado para la plataforma Android, el medio de difusión del mismo será el Market [28].

El Market de Android permite a los desarrolladores publicar y distribuir fácilmente sus juegos o aplicaciones directamente a usuarios con dispositivos compatibles con Android y está abierto a todos los desarrolladores de aplicaciones de Android. Una vez se han registrado, los desarrolladores tienen control completo sobre cuándo y cómo van a hacer disponibles sus aplicaciones a los usuarios. Los desarrolladores pueden gestionar de manera sencilla su colección de aplicaciones donde pueden visualizar información sobre descargas, ratings y comentarios. También pueden publicar updates y versiones nuevas de sus aplicaciones.

Para poder publicar el juego en el Market hay que seguir tres pasos:

- Crear un perfil de desarrollador
- Pagar una cuota de registro de 18 euros
- Aceptar el Acuerdo de distribución para desarrolladores de Android Market

Para el caso en el que el juego tenga un precio (como es el caso), la tarifa de transacción será del 30 % del valor. El valor del juego es de 0,99 euros, por lo que se saca de beneficio 0,693 euros por copia vendida.

En Android Market hay varias opciones [29] para conseguir una ubicación destacada y promocionar el juego. Así se facilita la búsqueda y se recompensa el desarrollo de aplicaciones de Android interesantes.

- **Destacadas**: aplicaciones interesantes seleccionadas por el equipo de Android Market
- **Gratuitas**: aplicaciones gratuitas populares
- **Principales novedades gratuitas**: aplicaciones gratuitas populares con menos de 30 días de antigüedad
- **Lo más vendido**: las aplicaciones de pago más populares de todos los tiempos
- **Principales novedades de pago**: aplicaciones de pago populares con menos de 30 días de antigüedad
- **Éxitos principales**: aplicaciones y juegos que generan el máximo nivel de ingresos, incluidas compras de aplicaciones y pagos a través de las mismas
- **Aplicaciones populares**: aplicaciones más instaladas en las últimas 24 horas
- **Selección de los editores**: algunas de las mejores aplicaciones para Android seleccionadas por el equipo de Android Market
- **Principales desarrolladores**: algunos de los mejores desarrolladores de Android Market, seleccionados por el equipo de Android Market

El equipo de Android es el encargado de seleccionar las aplicaciones de las categorías Destacadas y Selección de los editores, por lo que no es posible contratar anuncios publicitarios o espacios promocionales de pago en Android Market.

Una opción para obtener más beneficios por la aplicación, es la de insertar publicidad .Pero esto es una práctica que normalmente se lleva realiza en aplicaciones gratuitas así que no se aplica en este juego.

El coste presupuestado es de 17.664 euros y cada copia de la aplicación vendida da unos beneficios de 0,693 euros. En un año se han de vender alrededor de 25.000 copias para recuperar la inversión (**2125 copias al mes**), cifra factible después de revisar los datos de juegos similares en el Market.

Para conseguir estas ventas mensuales, hay que mantener el juego con actualizaciones periódicas y mejoras continuas de coste mínimo.

A.5. Planificación final

La planificación inicial sufrió cambios durante el desarrollo, por lo que a continuación se muestra la planificación real y las modificaciones que afectan al presupuesto.

Las siguientes tareas se tuvieron que modificar:

- **Desarrollo de la documentación y la memoria**: En lugar de comenzar a realizar la memoria una vez finalizado el periodo de formación, se comenzó cuando ya se había comenzado con el desarrollo.
- **Fase 1 - Estructura básica del juego**: los cálculos iniciales no tuvieron en cuenta que no se conocía la plataforma bajo la que se iba a desarrollar el juego, por lo que el incremento de tiempo en esta fase fue muy significativo (de 18 días que se habían planificado en un primer momento, se paso a 36).

  Aspectos básicos y dimensiones: Tenía una planificación inicial de 4 días, sin embargo por modificaciones en el diseño de la pantalla, se tardó más (7 días).

  Eventos básicos: En la planificación inicial, se realizaba en paralelo al resto de

tareas de los requisitos de la fase 1, sin embargo en la planificación final se tuvo que realizar con anterioridad a la Navegación entre pantallas (para poder navegar entre pantallas, era necesario controlar los eventos con los botones).

Interrupción y salida del juego: se implemento una vez se tenía controlada la navegación entre pantallas.

Implementación: Al tratarse de la base del juego, sólo se planificaron 8 días para la implementación. Aunque se trataba de una tarea sencilla, el hecho de ser los primeros intentos programando con Android, provocaron que esta tarea fuese mucho más extensa (15 días).

Evaluación: Pasó de 1 día a 2.

- **Fase 2 - Interfaz de usuario**: No se comenzó con un nuevo item hasta que se terminaba el anterior, lo que también hizo que se incrementase el número de días necesarios. En total, se pasó de 30 días a 40 días.

  Implementación: pasó de 15 a 22 días.

- **Fase 3 - Núcleo del juego**: Tuvo un incremento significativo (de 25 días a 40)

  Implementación: ocasionados principalmente por la implementación que de los 9 días planificados en un primer momento, se tardaron 24 en la realidad. Este gran aumento fue debido al cambio en la gestión del tablero.

- **Fase 4 - Otros**: sólo se vio incrementada en un día (de 15 a 16)

  Diseño: el incremento vino por la etapa de diseño.

- **Evaluación y pruebas finales**: de 5 días se paso a 10, dado que se quiso probar distintas y variadas posibilidades de juego para verificar su correcto funcionamiento.

Como se ha visto, los mayores retrasos han venido generados por la implementación (se trabajó con una plataforma en la que no se había programado con anterioridad, por lo que los problemas que aparecían siempre eran nuevos para el desarrollador).

Además de todo esto, hay que tener en cuenta que dado que sólo se disponía de un ingeniero para el diseño e implementación, en agosto no se contó con el por la llegada de vacaciones.

Diagrama de Gantt

	Nombre de tarea	Duración	Comienzo	Fin
1	**Formación previa**	**25 días**	**jue 22/04/10**	**mié 26/05/10**
2	Estudio documentación Android	25 días	jue 22/04/10	mié 26/05/10
3	Estudio documentación Eclipse	5 días	jue 20/05/10	mié 26/05/10
4	Desarrollo de la documentación y la memoria	150 días	lun 18/10/10	vie 13/05/11
5	**Fase 1 - Estructura básica del juego**	**36 días**	**jue 27/05/10**	**jue 15/07/10**
6	**Requisitos fase 1**	**15 días**	**jue 27/05/10**	**mié 16/06/10**
7	Aspecto básico y dimensiones	7 días	jue 27/05/10	vie 04/06/10
8	Navegación entre las distintas pantallas	5 días	jue 03/06/10	mié 09/06/10
9	Eventos básicos	5 días	jue 27/05/10	mié 02/06/10
10	Interrupción y salida del juego	5 días	jue 10/06/10	mié 16/06/10
11	Diseño	4 días	jue 17/06/10	mar 22/06/10
12	Implementación	15 días	mié 23/06/10	mar 13/07/10
13	Evaluación	2 días	mié 14/07/10	jue 15/07/10
14	**Fase 2 - Interfaz de usuario**	**40 días**	**vie 16/07/10**	**lun 11/10/10**
15	**Requisitos fase 2**	**9 días**	**vie 16/07/10**	**mié 28/07/10**
16	Diseño gráfico de pantallas	3 días	vie 16/07/10	mar 20/07/10
17	Cuadros de diálogo	2 días	mié 21/07/10	jue 22/07/10
18	Menús	3 días	vie 23/07/10	mar 27/07/10
19	Icono	1 día	mié 28/07/10	mié 28/07/10
20	Diseño	6 días	jue 29/07/10	lun 06/09/10
21	Implementación	22 días	mar 07/09/10	mié 06/10/10
22	Evaluación	3 días	jue 07/10/10	lun 11/10/10
23	**Fase 3 - Núcleo del juego**	**40 días**	**mar 12/10/10**	**lun 06/12/10**
24	**Requisitos fase 3**	**8 días**	**mar 12/10/10**	**jue 21/10/10**
25	Gestión del tablero	6 días	mar 12/10/10	mar 19/10/10
26	Posiciones cercanas	6 días	mar 12/10/10	mar 19/10/10
27	Ladrón	2 días	mié 20/10/10	jue 21/10/10
28	Diseño	7 días	vie 22/10/10	lun 01/11/10
29	Implementación	24 días	mar 02/11/10	vie 03/12/10
30	Evaluación	1 día	lun 06/12/10	lun 06/12/10
31	**Fase 4 - Otros**	**16 días**	**mar 07/12/10**	**mar 28/12/10**
32	**Requisitos fase 4**	**5 días**	**mar 07/12/10**	**lun 13/12/10**
33	Sonidos	3 días	mar 07/12/10	jue 09/12/10
34	Gestión del texto	2 días	vie 10/12/10	lun 13/12/10
35	Diseño	4 días	mar 14/12/10	vie 17/12/10
36	Implementación	5 días	lun 20/12/10	vie 24/12/10
37	Evaluación	2 días	lun 27/12/10	mar 28/12/10
38	Evaluación y pruebas finales	10 días	mié 29/12/10	mar 11/01/11

Figura A.8. Planificación final. Cambios.

Anexo B

Sistema de coordenadas del tablero

En este apéndice se incluye el sistema de coordenadas del tablero. Se detalla la numeración seguida en las celdas, vertices y lados, así como los cálculos necesarios para poder acceder a los elementos adyacentes a uno dado.

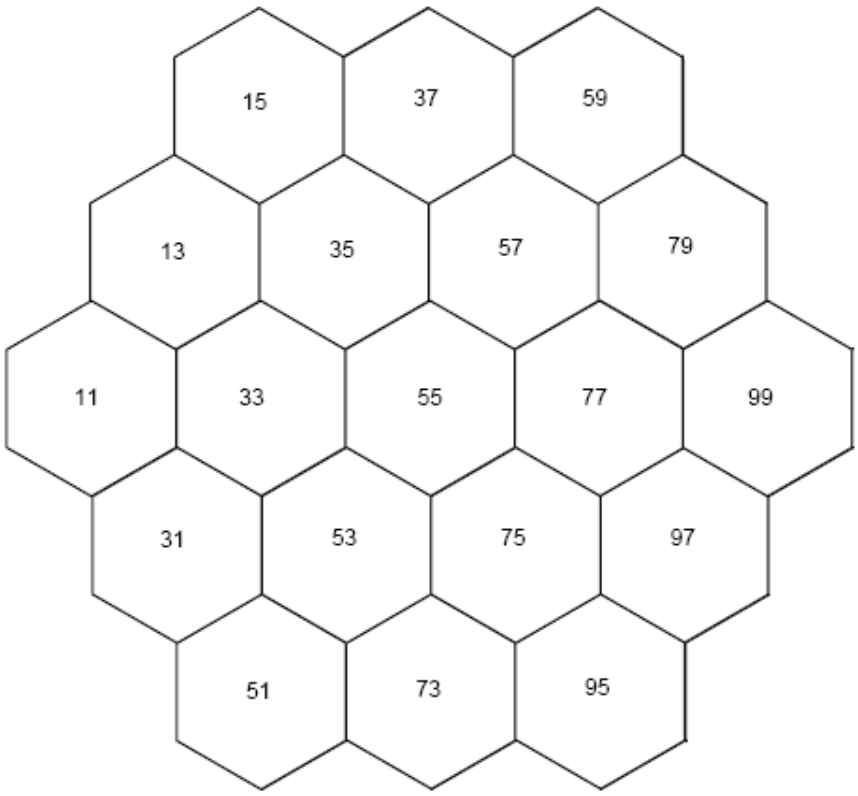


Figura B.1. Celdas. Interfaz del tablero.

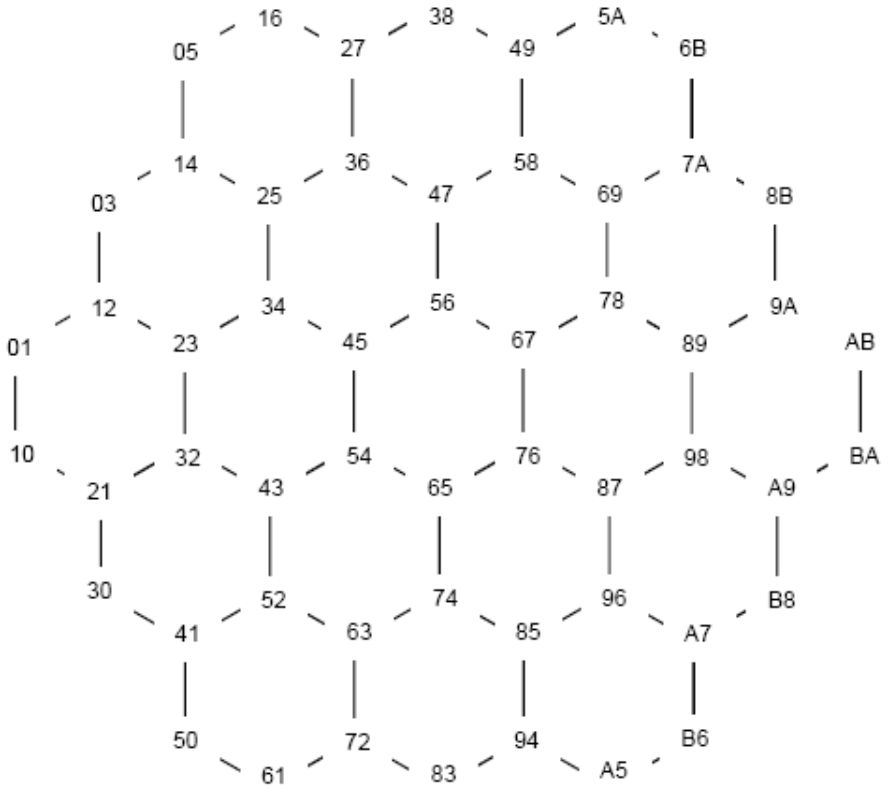


Figura B.2. Vértices. Sistema de coordenadas.

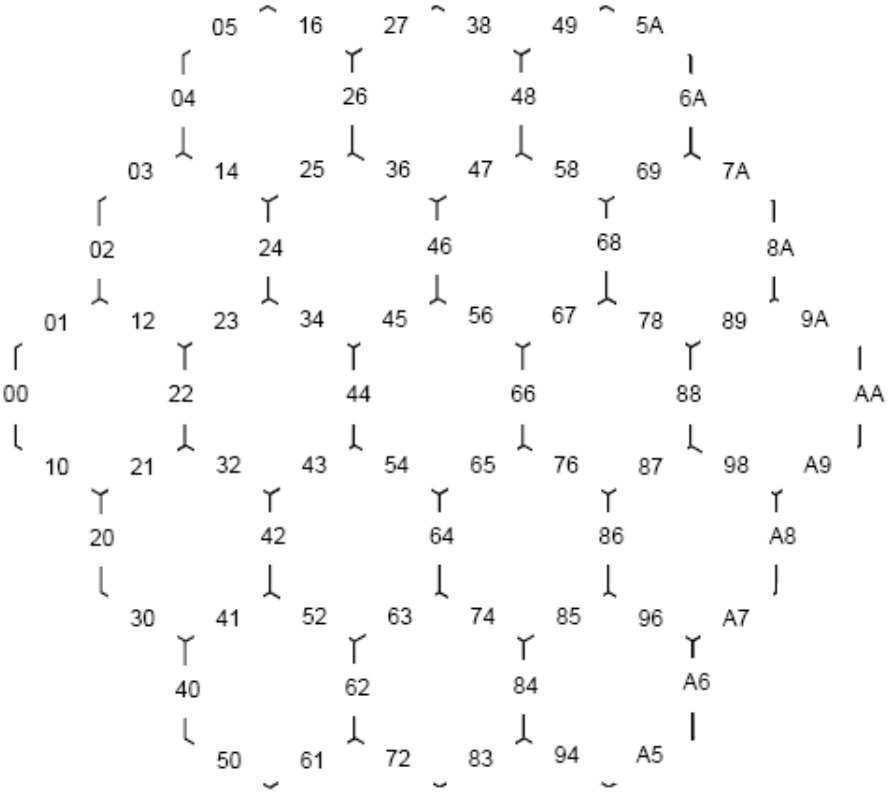


Figura B.3. Lados. Sistema de coordenadas.

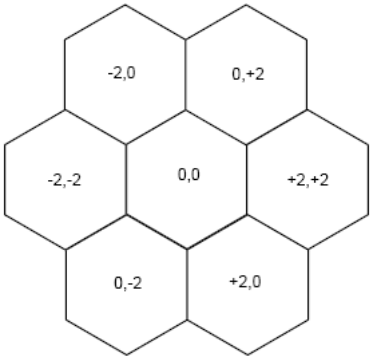


Figura B.4. Celdas adyacentes a una celda. Sistema de coordenadas.

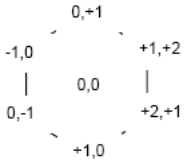


Figura B.5. Vértices adyacentes a una celda. Sistema de coordenadas.

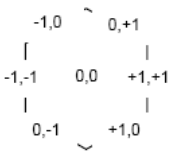


Figura B.6. Lados adyacentes a una celda. Sistema de coordenadas.

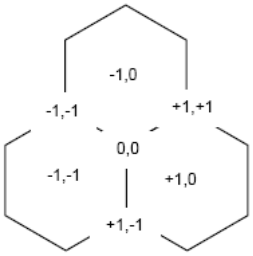


Figura B.7.]

Celdas y vértices adyacentes a un vértice [par,impar]. Sistema de coordenadas.

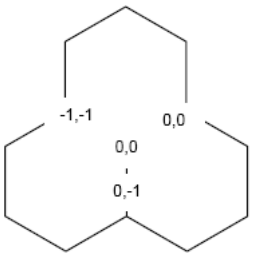


Figura B.8.]

Lados adyacentes a un vértice [par,impar]. Sistema de coordenadas.

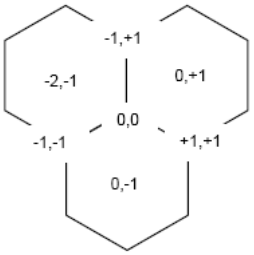


Figura B.9.]

Vértices y celdas adyacentes a un vértice [impar,par]. Sistema de coordenadas.

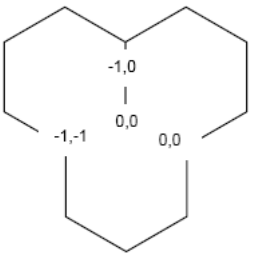


Figura B.10.]

Lados adyacentes a un vértice [impar,par]. Sistema de coordenadas.

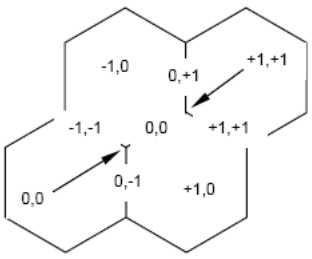


Figura B.11.]

Celdas, vértices y lados adyacentes a un lado [par,impar]. Sistema de coordenadas.

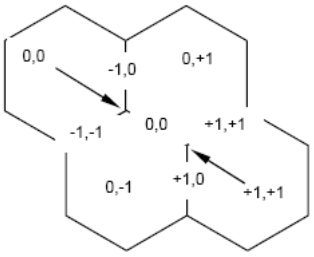


Figura B.12.]

Celdas, vértices y lados adyacentes a un lado [impar,par]. Sistema de coordenadas.

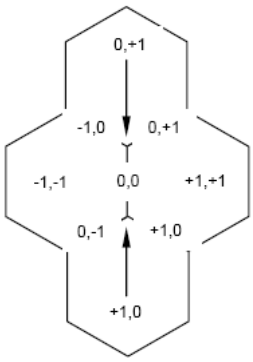


Figura B.13.]

Celdas, vértices y lados adyacentes a un lado [par,par]. Sistema de coordenadas.

Bibliografía

[1] M. Brownlow. (2011) Smartphones statistics and market share. [Online]. Available: http://www.email-marketing-reports.com/wireless-mobile/smartphone-statistics.htm#market

[2] I. Gartner. (2011) Gartner says worldwide mobile device sales to end users reached 1.6 billion units in 2010; smartphone sales grew 72 percent in 2010. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1543014

[3] A. Wells. (2010) Android growth, statistics and projections. [Online]. Available: http://www.androidtapp.com/android-growth-statistics-projections/

[4] C. GmbH. (2011) Catan - the first island for iphone and ipod touch. [Online]. Available: http://www.catan.com/electronic-games/catan-for-iphone.html

[5] B. Reed. (2010, Jun.) A brief history of smartphones. [Online]. Available: http://www.networkworld.com/slideshows/2010/061510-smartphone-history.html#slide1

[6] E. Pacheco. (2010) T-mobile g1, el primer terminal android llega al fin de su vida comercial. [Online]. Available: http://www.xatakamovil.com/mercado/t-mobile-g1-el-primer-terminal-android-llega-al-fin-de-su-vida-comercial

[7] symbian.nokia.com. (2011) Symbian at nokia. [Online]. Available: http://symbian.nokia.com/

[8] L. Foundation. (2011) Bienvenido a limo. [Online]. Available: http://www.limofoundation.org/es/bienvenido-a-limo.html

[9] P. Naik. (2011) Smartphone os comparison. [Online]. Available: http://www.techtree.com/India/Features/Smartphone_OS_Comparison/551-114322-899-1.html

[10] enbeeone3.com. (2011) Top 20 best android games. [Online]. Available: http://enbeeone3.com/top-20-best-android-games/

[11] A. F. Vasquez. (2010) Historia del sistema operativo android. [Online]. Available: http://kronox.org/2009/06/09/historia-de-android/

[12] O. H. Alliance. (2011) Open handset alliance. [Online]. Available: http://www.openhandsetalliance.com/

[13] Android.com. (2011) What is android? [Online]. Available: http://developer.android.com/guide/basics/what-is-android.html

[14] ——. (2011) Application fundamentals. [Online]. Available: http://developer.android.com/guide/topics/fundamentals.html

[15] H. Corporation. (2011) quietly brilliant. [Online]. Available: http://www.htc.com/la/quietlybrilliant/index.html

[16] Celularis.com. (2010) Htc llegando a la cima gracias a android. [Online]. Available: http://www.celularis.com/opinion/htc-llegando-a-la-cima-gracias-a-android.php

[17] HTC.com. (2011) Htc legend. [Online]. Available: http://www.htc.com/es/product/legend/overview.html?utm_source=google&utm_medium=cpc&utm_term=HTC%20legend&utm_campaign=Model+-+HTC+Legend&utm_content=sknWoZN20!pcrid!13485545385

[18] C. S. . S. E. S. Committee, *IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description for Software-Intensive Systems.* Luxembourg: IEEE Computer Society, 2000.

[19] S. Muthu. (2010) The structure of an android project. [Online]. Available: http://sudarmuthu.com/blog/the-structure-of-an-android-project

[20] Android.com. (2011) Technical resources - tutorials. [Online]. Available: http://developer.android.com/resources/browser.html?tag=tutorial

[21] ——. (2011) User interface - declaring layouts. [Online]. Available: http://developer.android.com/guide/topics/ui/declaring-layout.html

[22] ——. (2011) User interface - creating dialogs. [Online]. Available: http://developer.android.com/guide/topics/ui/dialogs.html

[23] ——. (2011) User interface - handling ui events. [Online]. Available: http://developer.android.com/guide/topics/ui/ui-events.html

[24] ——. (2011) User interface - menus. [Online]. Available: http://developer.android.com/guide/topics/ui/menus.html

[25] ——. (2011) Launcher icons. [Online]. Available: http://developer.android.com/guide/practices/ui_guidelines/icon_design_launcher.html

[26] R. S. Thomas, "Real-time decision making for adversarial environments using a plan-based heuristic," Jun. 2003.

[27] PacDV.com. (2011) Free sounds effects. [Online]. Available: http://www.pacdv.com/sounds/index.html

[28] Android.com. (2011) Android market. [Online]. Available: https://market.android.com/

[29] ——. (2011) Listas de aplicaciones destacadas. [Online]. Available: http://www.google.com/support/androidmarket/developer/bin/answer.py?hl=es&answer=1295940

[30] A. Pardo, *Hello, Android - Introducing Google's Mobile Development Platform.* Ed Burnette, 2008.

[31] M. L. Murphy, *The Busy Coder's Guide to Android Development.* Commonsware, 2008.

[32] flashkit.com. (2011) A flash developer resource site. [Online]. Available: http://www.flashkit.com/soundfx/Recreation/Casino/Dice_sou-Cochino_-8687/index.php

[33] Xacatamovil.com. (2011) Resultados financieros de htc. [Online]. Available: http://www.xatakamovil.com/mercado/htc-vendio-246-millones-de-smartphones-en-2010-resultados-financieros

[34] ——. (2010) Primer terminal android. [Online]. Available: http://www.xatakamovil.com/mercado/t-mobile-g1-el-primer-terminal-android-llega-al-fin-de-su-vida-comercial