ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Videojuego para Android

Realizado por:

NICOLÁS RODRÍGUEZ CAMPAYO

Dirigido por:
Dr. JOSÉ RAMÓN PORTILLO FERNÁNDEZ

Departamento
MATEMÁTICA APLICADA I

INDICE

3 Diseño de la aplicación. Ideas previas y evolución

4. Manual usuario

Nadal´s Wall

Nadal and Federer

5. Manual para desarrolladores

INTRODUCION

Con el paso del tiempo, el teléfono móvil paso de ser un instrumento básico de comunicación, a transformarse en una herramienta utilizada por la práctica totalidad de los sectores de la población, gracias a la reducción del tamaño de sus componentes y al aumento de sus prestaciones.

Años más tarde, las empresas de telecomunicaciones se plantean un nuevo reto, el de unir dos grandes tecnologías, Internet y los teléfonos móviles. El boom que supuso Internet en su época, junto con el furor de la telefonía móvil, dio lugar a teléfonos móviles más avanzados que no sólo servían para hablar, mandar mensajes cortos y consultar el calendario o la hora, sino también para acceder a Internet, hacer fotos y vídeos y jugar.

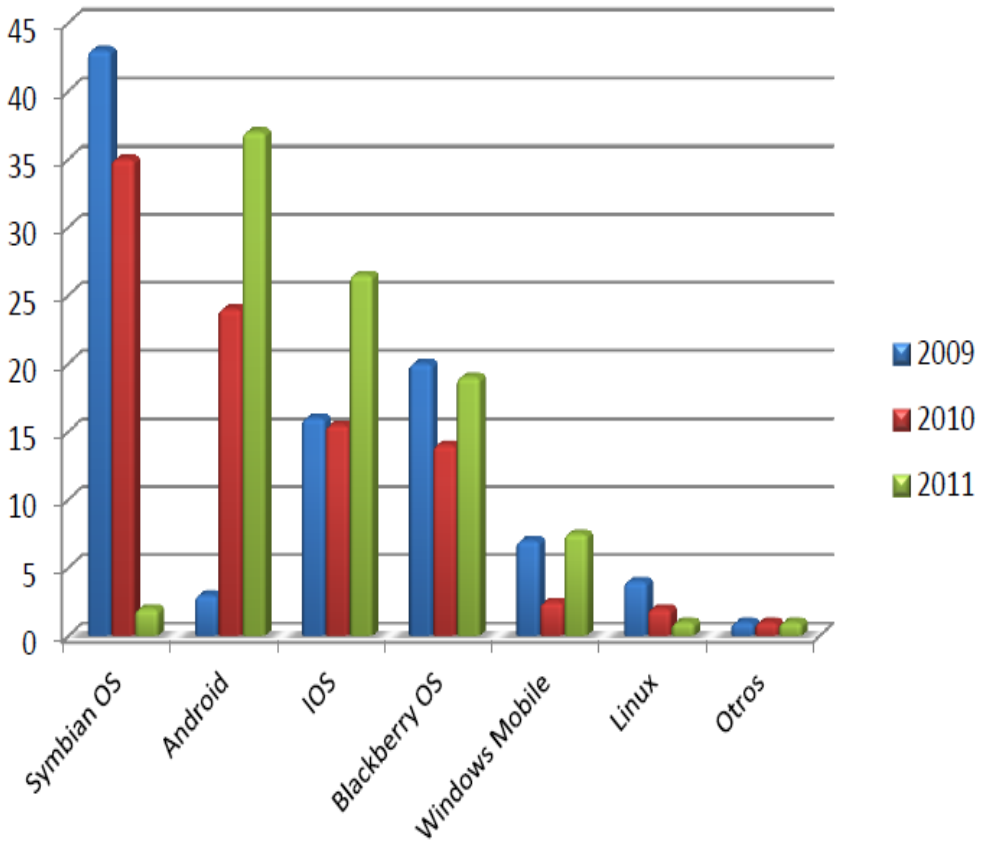
El avance de estos teléfonos, dio lugar a la tercera generación y con ella, al desarrollo de los teléfonos móviles más evolucionados, los teléfonos inteligentes o más comúnmente conocidos como, Smartphone. Los servicios asociados con la tercera generación, proporcionaron la capacidad de transferir tanto voz, una llamada telefónica, como datos, descarga de programas y juegos, intercambio de email y mensajería instantánea, como ambas cosas simultáneamente, una video llamada.

Todos estos aparatos cuentan con un sistema operativo que consta de varias partes; un kernel o núcleo, que ofrece servicios como la gestión de procesos y el acceso y gestión de memoria, un Middleware, que hace posible la existencia de aplicaciones para móviles, un entorno de ejecución de aplicaciones y un interfaz de usuario.

A medida que los distintos modelos de teléfonos móviles van adquiriendo popularidad, los sistemas operativos con los que funcionan van consolidán-

dose en el mercado. Los sistemas operativos más utilizados son: Android, desarrollado por Google; Symbian OS, producto de la alianza de varias empresas de telefonía móvil; iOS, creado por Apple; BlackBerry OS, desarrollado por la empresa canadiense RIM y Windows Phone, creado por Microsoft.

ESTUDIO DEL MERCADO



Son las estadísticas que se desprenden de un estudio realizado por una consultora.

Android aparece como sistema operativo en más del doble de los smartphones que se venden en España, por delante de Symbian, BlackBerry e iOS.

Concretamente, Google domina el sector en nuestro país con un 63% de cuota, más visible entre el público joven.

El desarrollo de Android en el mercado de los smartphones ha sido espectacular. Su escalada es imparable y hoy por hoy pocos son los fabricantes que no sucumben a su éxito y se unen a sus filas. Los grandes

perjudicados han sido, sobre todo, RIM con su BlackBerry y Symbian, aunque éste último agravado por otras circunstancias.

En el otro lado está iPhone con su iOS que, hoy por hoy, es la alternativa de peso al robot verde, aunque en este caso no se vea reflejado en las estadísticas.

Los Smartphone son los más deseados

Los datos provienen de la consultora Kantal Worldpanel, que ha analizado los hábitos de los usuarios en España durante el trimestre que engloba diciembre de 2011 hasta últimos de febrero de 2012. La primera gran conclusión del estudio es que los smartphones, o teléfonos inteligentes, empiezan a imponerse a los modelos tradicionales en seis de cada diez ventas. Actualmente, en nuestro país, ese nivel de ventas ha propiciado que la presencia de estos terminales aumente hasta un 35%, lo cual nos dice que en apenas un año ha subido 13 puntos porcentuales. Un crecimiento muy destacado que también refleja las preferencias de los consumidores.

%Cuota Sistemas Operativos sobre nuevos terminales	6 dic'10- 27 feb'11	5 dic'11- 26 feb'12
Android	17%	63%
Symbian	55%	17%
RIM	14%	8%
iOS	9%	7%
Windows	1%	2%
Otros	4%	3%
Total Smartphones	100%	100%
Fuente: Worldpanel ComTech		

Los teléfonos con Android han tenido mucho que ver en del resultado de estas estadísticas. Según recoge la consultora, un 63% de las ventas de smartphones llevan impreso el sello de Google y su sistema operativo Android. El crecimiento respecto al año pasado es destacado pues en el mismo periodo de 2011 el porcentaje era del 17%. Symbian es uno de los sistemas más perjudicados, que pasa de un 55% al 17%. En parte esto se ha debido no solo a su menor evolución respecto a otras plataformas, sino a la estrategia de Nokia al optar por una alianza con Microsoft. Los canadienses de RIM también se tambalean con un escaso 8% de cuota de mercado en España. El descenso en un año ha sido de seis puntos. Apple le sigue con un 7%, que baja respecto al 9% de cuota del curso anterior. Esta posición llama especialmente la atención, más si tenemos en cuenta el éxito comercial del iPhone 4S, presentado en octubre de 2011 por los de Cupertino. No obstante es la cifra que ha registrado la consultora en España, que no en EEUU.. En el último escalón, con otros sistemas en minoría, está Windows Phone. Aunque logra subir de un 1% a un 2%, su presencia en el mercado por ahora es escasa. Tal vez la presentación de nuevos terminales

Nokia Lumia y próxima generación de teléfonos de compañías como HTC, LG o Samsung logren animar al usuario.

Móviles y videojuegos

La llegada de los teléfonos inteligentes ha supuesto una revolución para el mercado de videojuegos portátiles debido a que ya no es necesario llevar una consola portátil contigo para jugar, puedes hacerlo en tu propio móvil.

Las ventas de videojuegos en teléfonos inteligentes**,** sobre todo Iphone y dispositivos Android, supusieron un 34% del total de ventas de videojuegos portátiles y las previsiones dicen que en muy poco tiempo el juego en teléfonos superará al de consolas portátiles.

El éxito se debe a la facilidad de adquirir los juegos mediante las tiendas de aplicaciones que tienen los teléfonos, el AppStore de Apple o el Market de Android, y al menor precio de éstos respecto a videojuegos para consolas. El precio de la mayoría de juegos para móviles es inferior a 3 euros, y un gran porcentaje de juegos pueden ser adquiridos gratuitamente.

El mercado de videojuegos para Android se encuentra en un momento inmejorable y todo parece indicar que esta tendencia solo acaba de comenzar.

Aspectos económicos

Si nos centramos ahora en los beneficios, podemos sacar unas conclusiones bastante valiosas:

En primer lugar, se puede apreciar que la diferencia entre el número de descargas cuando un juego es gratis o cuando es de pago es enorme.

Pero es que además se ha observado que es importantísimo que el juego tenga al menos una demo gratuita, donde los usuarios prueben la mecánica del juego y se puedan "*enganchar*". Si no existe esa demo es mucho más difícil que los usuarios, ante tanta oferta gratuita y de pago opten por comprarlo.

De hecho podemos apreciar cómo juegos que parte con la ventaja de haber sido promocionado en muchísimos medios, habiendo causado mucho revuelo entre los usuarios, con un precio muy reducido y con una mecánica enfocada a un conjunto más general de usuario, tiene menos ventas que el parchís para android.

Las estadísticas expuestas por Google muestran conclusiones parecidas. En una conferencia de Mayo de 2010, se indica que el 80% de los juegos de pago más descargados del Market (observando los 10 primeros) tienen demo o versión gratuita.

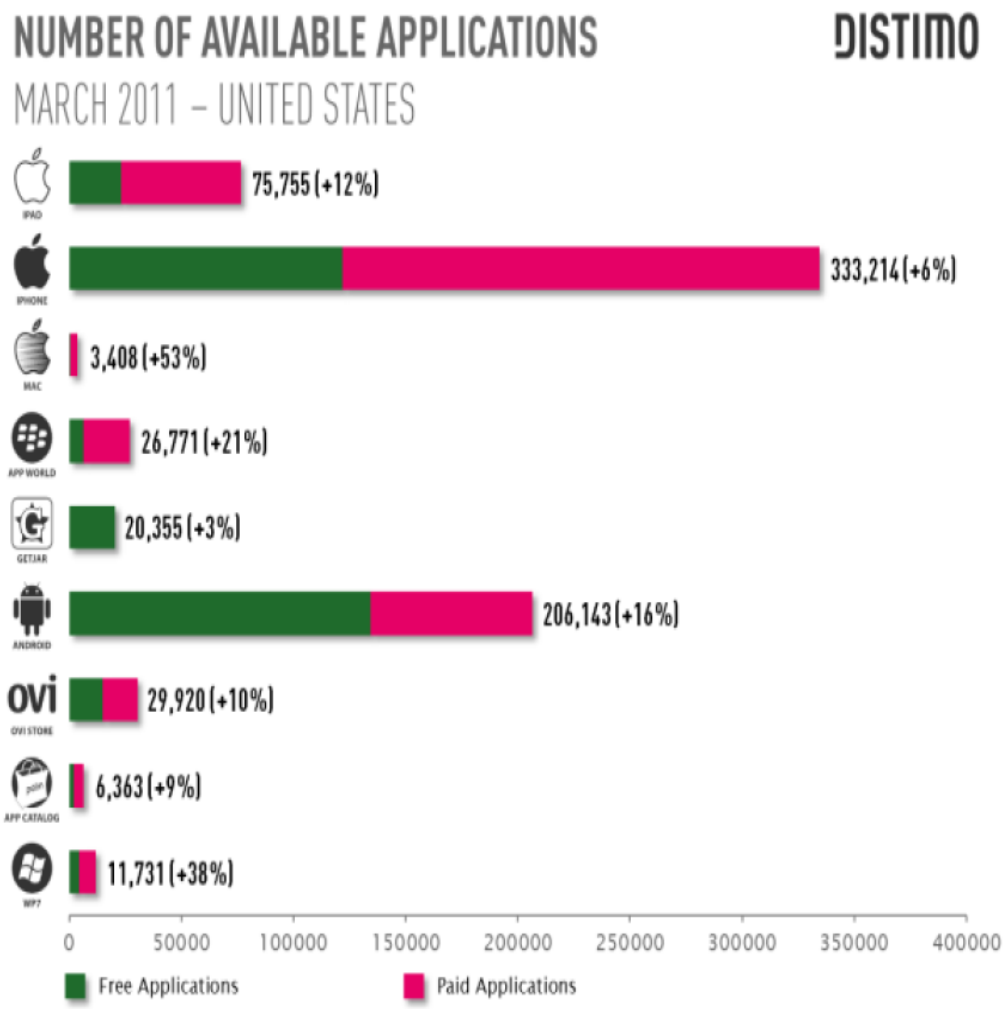
En lo que al precio respecta, el precio promedio de las aplicaciones de pago más vendidas es de 1,80€ (cogiendo como muestra los 10 juegos más populares del Market a día 16 de Mayo de 2011).

Por otro lado, no hay que olvidar la opción de recibir dinero por un juego basándonos únicamente en la publicidad. A pesar de que podamos poner demo gratuita o no, las versiones de pago siempre son descargadas muchas menos veces que una versión gratuita.

De hecho los desarrolladores de Angry Birds reconocen en diversas entrevistas que en Android es más difícil vender aplicaciones que en otros medios como la tienda digital de Apple. Siendo a través de la publicidad como se puede ganar más dinero con un juego para Android.

Esto tiene mucho sentido si tenemos en cuenta el medio a través del cual gana dinero Google con sus usuarios, que normalmente pasa por ofrecer todo gratuitamente y recibir ingresos por publicidad. Por tanto, es comprensible que los usuarios de Android prefieran las aplicaciones gratuitas y no les importe tener una pequeña publicidad en la pantalla de juego a cambio.

Estadísticas extraídas por la empresa Distimo nos ayudan a ponernos en situación y nos muestran como más de la mitad de las aplicaciones del Android Market son gratuitas. Siendo el porcentaje de aplicaciones gratuitas (65%) el más grande de entre todas las tiendas de aplicaciones analizadas. Y, además, observamos como el Android Market tiene más aplicaciones gratuitas que cualquiera de las tiendas de la competencia.



Estadísticas proporcionadas por la empresa Distimo correspondientes a Marzo de 2011 en Estados Unidos. En ellas podemos ver el número de aplicaciones total de cada una de las tiendas virtuales de los diferentes sistemas operativos móviles, junto con el porcentaje de crecimiento del último mes.

Además podemos ver la proporción entre aplicaciones gratuitas y de pago de cada sistema.

Nótese que aunque la estadística se haya llevado a cabo con el Android Market de Estados Unidos, la diferencia de las tiendas virtuales entre países suele estar en las aplicaciones que tienen restricción regional y no se distribuyen en todos los países. Así que puede haber una pequeña diferencia en-

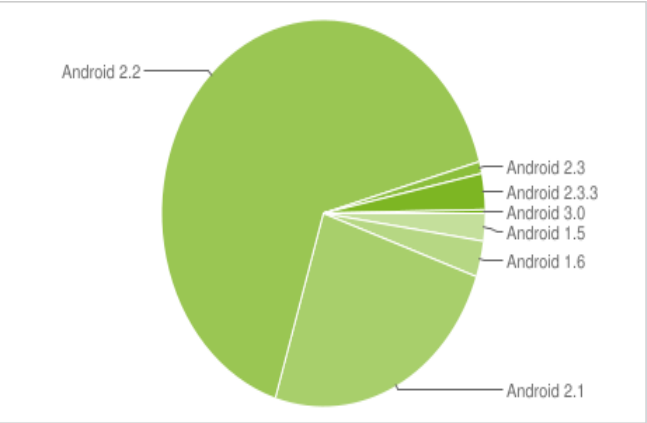
tre las estadísticas en EEUU y en España, pero no la suficiente como para alterar las conclusiones extraídas.

Compatibilidad entre versiones de Android

Podemos observar como la mayoría de los juegos son compatibles con la versión 1.6 y superiores de Android. Vamos a intentar analizar si el motivo se debe a una mayoría de usuarios con esta versión en el momento del desarrollo o a que, a pesar de que a día de hoy tan solo un 3% de los usuarios de Android estén utilizando la versión 1.6 sigue siendo la más recomendable para el desarrollo de videojuegos.

Porcentaje de uso de cada versión del S.O. Android, sobre el total de usuarios de Android, a día 2 de Mayo de 2011. Este porcentaje se mide monitorizando los accesos al Android Market.

En caso de que a día de hoy desarrolláramos un videojuego para la versión 2.2 o superior estaríamos dejando fuera a una gran cantidad de usuarios que tienen versiones antiguas (especialmente la versión 2.1 es muy utilizada con un 24,5%). Aún así, la versión 2.2 puede ser una opción a tener en cuenta si vamos a desarrollar un videojuego en 3D. Esto es debido a que la compatibilidad con OpenGL ES 2.0 permite el uso de *shaders* y, por tanto, facilita algunas cosas mientras que posibilita otras nuevas. Estamos llegando a un 70% de los usuarios de Android y conseguimos cosas que no podemos conseguir de otra forma.



Platform	API Level	Distribution
Android 1.5	3	2.3%
Android 1.6	4	3.0%
Android 2.1	7	24.5%
Android 2.2	8	65.9%
Android 2.3	9	1.0%
Android 2.3.3	10	3.0%
Android 3.0	11	0.3%

Porcentaje de uso de cada versión del S.O. Android, sobre el total de usuarios de Android, a día 2 de Mayo de 2011. Este porcentaje se mide monitorizando los accesos al Android Market.

Si, por el contrario, quisiéramos trabajar con una versión más moderna y nos diera por escoger la versión 2.3 como requisito mínimo, estamos dirigiéndonos a poco más de un 4% de los usuarios. Por tanto, parece que a día de hoy sacar un juego para la versión 2.3 de Android o superior no vale la pena aunque las mejoras que esta versión del sistema operativo aporta puedan facilitar bastante la tarea de desarrollo de videojuegos.

Podríamos deducir, por tanto, que lo más inteligente si no necesitamos *shaders* es desarrollar para la versión 2.1 (nótese que cuando desarrollamos una aplicación para una versión esta funciona en ésta versión y en las superiores), que por otro lado, no aporta casi mejoras en el desarrollo de videojuegos con respecto a la versión 1.6, con lo cual la mayoría de desarrolladores se decantan por la versión 1.6 aún a día de hoy como requisito mínimo. De esta forma consiguen llegar prácticamente a todos los usuarios.

Esto supone tener que añadir un selector para distinguir entre controles de pantalla *singletouch* y controles de pantalla *multitouch*. Puesto que el soporte para pantallas *multitouch* es la mejora más notoria, de cara al desarrollo de videojuegos, que se añade en la versión 2.1 de Android.

Por tanto, habría que hacer un inciso, puesto que si los controles del videojuego no se prestan a ser implementados mediante una pantalla mono táctil, ya que se limita la *jugabilidad* haciendo que la experiencia de usuario se vea excesivamente afectada, también sería una buena elección trabajar para la versión 2.1 de Android. De esta forma, ahorraríamos el tiempo necesario para implementar y probar un control mono táctil que va a ofrecer una mala experiencia de usuario.

Tampoco tenemos que olvidar que, cuando finalicemos el desarrollo del videojuego, el grafico habrá cambiado, y las versiones 1.5 y 1.6 serán, con toda probabilidad, más minoritarias de lo que ya lo son actualmente, cuando solo representan al 5,3% de los usuarios de Android.

OBJETIVOS

Visto el estudio de mercado la opción mas interesante para el desarrollo de un videojuego con el uso de los smartphones es hacerlo para Android ya que supone un sector de la industria del software bastante importante actualmente, con un crecimiento cada vez mayor. El número de aplicaciones que son lanzadas para este tipo de dispositivos crece día a día, y dentro de éstas las lúdicas cobran una importancia notoria desde el punto de vista comercial.

Ya que Android es uno de los sistemas operativos que ha experimentado un mayor crecimiento en el mercado, tenía una mayor preferencia por programar una aplicación o videojuego en este sistema operativo. De esta forma conseguiría familiarizarme con la plataforma, adquirir una gran experiencia y ser capaz de crear mis propias aplicaciones.

Para ello tendré que investigar y estudiar el funcionamiento de Android ya que este tipo de sistemas operativos no está incluido en el temario de la ETSII.

En primer lugar, quería hacer uso de algunas de las funciones propias de este tipo de teléfonos, sin limitarnos únicamente a ofrecer un producto que también pudiese disfrutarse en cualquier otro tipo de plataforma. Para ello, partimos de la idea de otorgar una gran importancia al sensor de movimiento que incluyen la mayoría de smartphones actualmente a la venta, así como hacer uso de la pantalla táctil, etc.

También parto de la idea de ofrecer un producto simple y entretenido, que mediante el uso de herramientas gratuitas de libre adquisición nos permitan realizar una aplicación interesante desde el punto de vista del público.

Demostrar que no hace falta un gran presupuesto para ofrecer contenidos de calidad. Así mismo, quiero aportar mi granito de arena a la comunidad de Android, ya que explicare paso a paso cada una de las herramientas que he utilizado y la forma en la que las he usado, para que en el futuro otros miembros interesados en este tipo de aplicaciones tengan parte del camino recorrido.

Por otro lado, tenemos al videojuego, cada vez más presenten en nuestra sociedad. Una forma relativamente nueva de expresarnos y enviar nuestros conocimientos o emociones a los jugadores, que los recibirán y percibirán de una forma diferente a como lo hacen cuando ven una película o leen un libro.

Los videojuegos son tan importantes para un dispositivo móvil que gran parte del éxito de un S.O. depende de las facilidades que este dé para desarrollar videojuegos y, por tanto, del catálogo que este ofrezca a sus usuarios.

El desarrollo de videojuegos para dispositivos móviles siempre ha despertado mi curiosidad, y que las personas puedan llegar a divertirse, disfrutar o aprender con ello, así que desde el principio pensé en combinar ambas cosas y de ahí salió la idea del proyecto.

Los métodos de control como pantallas táctiles o acelerómetros dan unas posibilidades que no tienen las consolas de sobremesa. Además, son el medio perfecto para desarrolladores independientes debido a que no es necesario un juego demasiado complejo para tener éxito

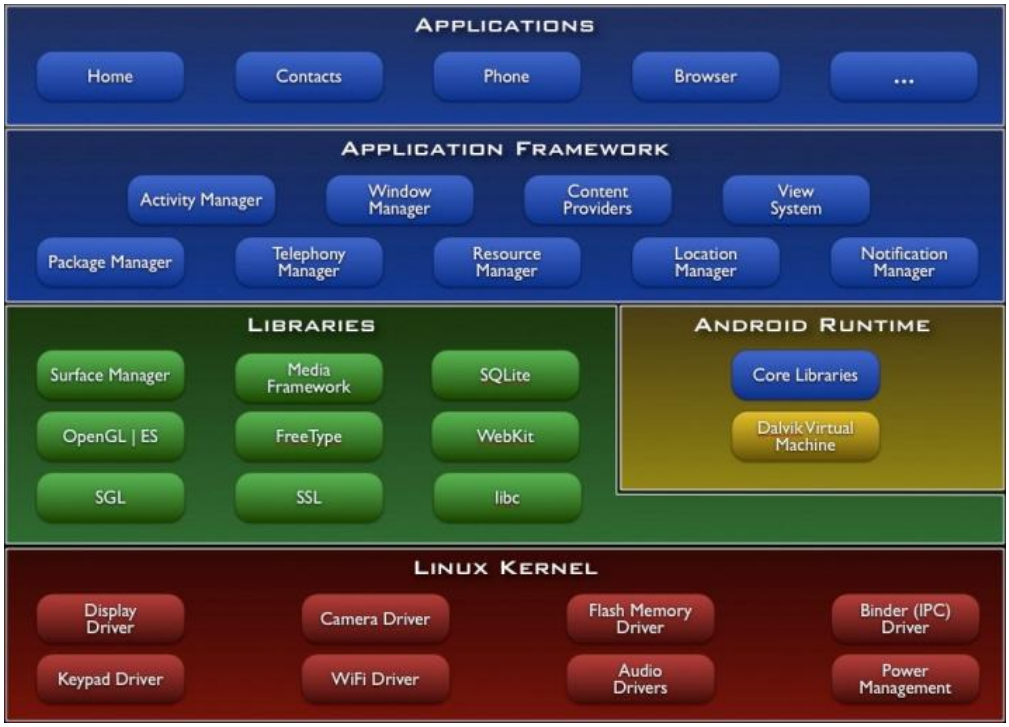
Android

Android es un conjunto de software que constituye un ecosistema para las aplicaciones móviles. Dentro de este conjunto se incluye un sistema operativo móvil, lo que significa que Android está dirigido principalmente a teléfonos inteligentes (o *smartphone*) y a *tablets*.

Y detrás de este software se encuentra la empresa Google Inc., archiconocida multinacional dedicada a ofrecer servicios y productos basados generalmente en Internet.

Estructura de Android

El conjunto de software denominado Android incluye: un sistema operativo, software intermedio que trabaja al servicio de las aplicaciones que se encuentran por encima y algunas aplicaciones claves que vienen incluidas desde el principio con el sistema operativo. Un ejemplo de aplicación por defecto es el Android Market, la tienda desde donde podemos comprar o descargar gratuitamente las aplicaciones que los desarrolladores ofrecen a través de ella. Para ver la estructura de todo el paquete Android tenemos la siguiente figura


APPLICATIONS
Home
Contacts
Phone
Browser
...
APPLICATION FRAMEWORK
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
LIBRARIES
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
ANDROID RUNTIME
Core Libraries
Dalvik Virtual Machine
LINUX KERNEL
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Como podemos ver en la figura, el núcleo del sistema operativo es una modificación del núcleo de Linux. En él se encuentran los drivers que permiten comunicarse con el hardware específico de cada dispositivo que implementa Android. Por encima tenemos una capa con todas las librerías, accesibles a la hora de programar aplicaciones, las cuales hacen uso de los drivers implementados. Entre estas librerías encontramos OpenGL ES, SQLite, SSL, etc.

Dentro del sistema operativo también encontramos la famosa maquina Dalvik. Y es que, en Android, las aplicaciones se ejecutan en una instancia de la máquina Dalvik. Cada instancia es independiente y, por tanto, ejecuta una aplicación de forma cerrada. Este es un buen mecanismo de seguridad, pues nadie puede llegar a entrometerse en la ejecución de una aplicación. De igual forma, los recursos de cada aplicación se encuentran en un fragmento de memoria privada e inaccesible desde fuera de la aplicación.

Y hasta aquí el sistema operativo, ya que en la siguiente capa, encontramos una serie de componentes utilizados por las aplicaciones para realizar funciones determinadas. Entre estos componentes, por ejemplo, se encuentra el Notification Manager, el cual recibe notificaciones de las aplicaciones y las presenta al usuario a través de la barra de notificaciones. Otro ejemplo es el Activity Manger, el cual se encarga de la gestión de las actividades de cada aplicación (veremos que es una actividad en el segundo apartado de este mismo capítulo).

Como estos componentes que conforman el *framework* manejan una información sensible para el usuario, pues en un dispositivo móvil normalmente el usuario introduce bastante información personal, se ha desarrollado un mecanismo de permisos por tal de mantener la información de forma segura.

De esta forma, cuando la aplicación se instala, esta solicita al usuario permisos para acceder a los componentes del *framework* que sean necesarios por

tal de llevar a cabo las acciones pertinentes. El usuario deberá aceptar estos permisos si quiere instalar la aplicación, ya que, en caso contrario, esta aplicación no podrá utilizar los poderes que el permiso solicitado otorga.

Si seguimos con la estructura de Android, un nivel por encima tenemos las aplicaciones, que pueden estar creadas por desarrolladores externos o por la propia Google. De entre las creadas por la empresa Google destacan unas cuantas que, como hemos dicho antes, son clave y vienen incluidas en el paquete.

Análisis de requisitos, diseño e implementación

Herramientas para el desarrollo

Equipos de desarrollo:

Sony VAIO E Series VPC-EH2K1E

Características:

- Procesador: Intel® Core™ i3-2330M a 2,20 GHz
- Memoria RAM: 4 GB DDR3 SDRAM a 1333 MHz
- Disco duro: 640 GB Serial ATA de 5400rpm)
- Unidad óptica: DVD SuperMulti Drive
- Pantalla VAIO Display de 39,5 cm (15,5 pulgadas), WXGA 1366x768
- Tarjeta gráfica: NVIDIA GeForce 410M GPU con 1 GB VRAM
- El cliente listo sabe: redcoon.com es mejor!
- Cámara Web 640 x 480
- Ranuras de expansión: SD Card (SDHC, SDXC), Memory Stick Duo
- Batería de iones de lítio (VGP-BPS26), Autonomía 4 horas

Sistema operativo :

- Windows 7 Home Premium (64-bit) Original

Conexión de redes :

- Gigabit Ethernet 10/100/1000
- WLAN: 802.11 b/g/n
- Bluetooth 3.0 + HS con alcance máx. 10m y velocidad máx. de datos de 24 Mbps

Interfaces :

- 4 x USB 2.0
- Toma de auriculares (minitoma estéreo)
- Toma de micrófono (minitoma estéreo)
- Salida HDMI
- Salida VGA

Dimensiones :

- 369,8 x 31,3 x 248,4 mm

Peso :

- 2,7 kg

Smartphone utilizado:

Htc desire

Físicas

Tamaño:

Peso: 135 gramos con batería

Velocidad de Proceso de la CPU: 1 GHz

Energía y Batería

Tipo de batería: Batería de Ion-Litio recargable

Capacidad: 1400 mAh

Tiempo de conversación:

WCDMA: Hasta 390 minutos

GSM: Hasta 400 minutos

Tiempo en espera:

WCDMA: Hasta 360 horas

GSM: Hasta 340 horas

Cámara

Cámara a color de 5 megapíxeles

Capacidad de detección de caras

Auto focus y flash

Captura de fotos en modo panorámico

Geo-etiquetado

Conectores

Jack de audio estéreo de 3.5 mm

Micro-USB estándar
(micro-USB 2.0 de 5-pins)

Sensores

Acelerómetro (G-Sensor)

Brújula digital

Sensor de proximidad

Sensor de luz de ambiente

Widgets de HTC

Marcadores, Calendario, Reloj, Footprints, Friend Stream, Correo, Mensajes, Música, Noticias, Gente, Álbum de Fotos, Marco de Fotos, Búsqueda, Ajustes, Bolsa, Twitter,

El Tiempo

Widgets descargables

Redes Sociales

Integración con Facebook™

Friend Stream

Compartir Fotos en Facebook, Flickr, y Twitter

Compartir vídeos en YouTube™

HTC Peep para twittear

Requisitos Recomendados para su Sistema Windows

Windows® 7, Windows Vista®, o Windows XP

HTC Sync

Características Especiales

Baja el volumen del tono de llamada según se coge el teléfono

El teléfono se silencia si se deja boca abajo

Copia de seguridad automática a la tarjeta microSD de ciertos datos, como mensajes SMS/MMS, marcadores, contraseñas de Wi-Fi, y más

Pantalla

Tipo: Pantalla táctil AMOLED capacitiva con capacidad de zoom por "pellizco"
Tamaño: 3.7 pulgadas
Resolución: 480 X 800 WVGA

Plataforma

Android™ 2.1 (Éclair) con HTC Sense™

Almacenamiento

ROM: 512 MB

RAM: 576 MB

Ranura de Expansión:

Tarjetade memoria microSD™ (compatible con SD 2.0)

Soporta hasta 32 GB

Bandas de Red

Europa:

HSPA/WCDMA: 900/2100 MHz

GSM: 850/900/1800/1900 MHz

Asia del Este y Pacífico:

HSPA/WCDMA: 900/2100 MHz

GSM: 850/900/1800/1900 MHz

Internet 3G:

Velocidad de descarga de hasta 7.2 Mbps

Hasta 2 Mbps de velocidad de subida

GPRS: Hasta 114 kbps descargando

EDGE: Hasta 560 kbps descargando

Wi-Fi®: IEEE 802.11 b/g

Tethering

Conexión a Internet compartida por USB

Bluetooth

Bluetooth® 2.1 con Velocidad de Transferencia Mejorada (EDR)

A2DP para auriculares estéreo inalámbricos

FTP y OPP (object push) para transferir ficheros

Otros perfiles soportados: AVRCP, GAP, GOEP, HFP, HSP, PBAP, SPP, Service Discovery Application Profile

Multimedia

Galería para ver Fotos y Vídeos

Música

Radio FM

Formatos de audio soportados:
Reproducción: .aac, .amr, .ogg, .m4a, .mid, .mp3, .wav, .wma
Grabación: .amr

Formatos de video soportados:
Reproducción: .3gp, .3g2, .mp4, .wmv
Grabación: .3gp

Posicionamiento

Antena GPS interna

Google Maps

HTC Footprints™

Eclipse + Android SDK:

El conocido entorno de desarrollo supone la mejor herramienta para programar en Android, debido a su fuerte orientación JAVA. Incluye todas las librerías necesarias para el desarrollo de aplicaciones, así como una máquina virtual configurable para probar el funcionamiento. Si se emplean dispositivos físicos, el propio Eclipse puede instalar y ejecutar la aplicación mediante una simple conexión USB, ofreciendo además todo tipo de herramientas para facilitar la programación, localizar errores, etc. Además, es completamente gratuito y de código abierto.

Más información en:

http://www.eclipse.org/

http://developer.android.com/sdk/index.html

Características que debe tener un terminal Android

Un terminal Android a nivel de hardware debe cumplir los siguientes requisitos:

Debe tener una resolución de las siguientes:

240x320

240x400

240x432

320x480

480x800

480x854

Acelerómetro para cambiar la orientación de la pantalla entre landscape y portrait

Pantalla táctil con tecnología resistiva o capacitiva, prefiriendo las segundas que soportan tecnología multitáctil.

Puerto micro USB que implemente un cliente para almacenamiento masivo y un ADB (Android Debug Bridge).

Teclas de navegación (atrás, menú, home y buscar).

Soporte para tecnologías Wifi y 3G o HSPA.

Cámara de al menos 2 MegaPixels preferiblemente con autofocus.

Un acelerómetro de 3 ejes capaz de actualizarse a 50Hz o más.

Diferentes opciones

Para desarrollar en Android disponemos de varias opciones, cada una con sus pros y sus contras:

Eclipse: Es la principal opción apoyada por Google. Este IDE es Open-Source con lo que se pueden programar plugins y añadirlos sin problemas. Posee características como la auto compleción de código o la firma automática de aplicaciones para la posterior depuración en terminales. Por descontado al tener el plugin oficial integra todas las utilidades de Android Tools como DDMS (Dalvik Debug Monitor Server), gestor de AVD's (Android Virtual Devices) etc en la propia interfaz. Como desventajas cabe destacar su elevado consumo de recursos y pobre velocidad general al estar escrito en Java, lo que supone una sustancial perdida de rendimiento frente a otros IDE's escritos en C o C++.

NetBeans: Opción secundaria, el IDE en este caso no es Open-Source, es propiedad de Oracle, por lo que no es tan versátil como Eclipse en el sentido de poder acceder y modificar libremente el código. No existe ningún plugin oficial para esta plataforma aunque se puede instalar igualmente el de eclipse con un poco de esfuerzo con lo que tiene casi las mismas funcionalidades que el Eclipse. La gran desventaja de Netbeans es que no firma automáticamente las aplicaciones y además está a la par de Eclipse en cuanto a recursos y velocidad general.

Otro IDE más simple como Emacs: Google pone a disposición de los usuarios el android SDK que trae en el interior de la carpeta tools todo lo necesario para firmar, compilar, e instalar aplicaciones en un móvil a parte del ya nombrado DDMS, emuladores... La gran desventaja de este método es que todo se debe hacer a mano sin ninguna ayuda lo cual hace que el programador sea más propenso a errores y tarde más en detectarlos. La ventaja es que estos IDE's y consola son infinitamente más ligeros y no consumen casi recursos.

Kit de Desarrollo Nativo (NDK) Google también ha lanzado el NDK que utilizándolo de forma conjunta con el SDK permite acceder directamente al hardware del móvil y construir aplicaciones en código nativo C o C++. Con esta opción se pueden desarrollar aplicaciones que utilicen directamente las librerías propias del sistema sin tener que ir a través de la máquina virtual Dalvik.

Utilizare Eclipse, ya que este, es el que más hemos utilizado en la universidad y con el que estoy más familiarizado

Diseño de la aplicación. Ideas previas y evolución

Conceptos básicos

Hay varios conceptos esenciales que por lo menos te deben sonar para entender un poco de lo que pasa aquí:

- **View :** Son las clases básicas del interfaz de usuario que controlan el layout de la pantalla y la funcionalidad otorgada al usuario, por ejemplo botones, cajas de texto, etc…

- **Activity :** Se podría decir que es lo que “vemos” en la pantalla, aunque esto realmente no es así, ya que las Activities realmente son las diferentes acciones que podemos realizar en nuestra aplicación, ya sea mostrar un dialogo, una pantalla, operaciones ocultas para el usuario,etc… Estas acciones se pueden comunicar con otras a fin de hacer determinadas tareas u obtener diferentes resultados. Una aplicación de Android es realmente un conjunto de pequeñas activity enlazadas.

- **Intent :** Los Intends son la herramienta de desarrollo que permite comunicar unas actividades con otras y facilitan la transmisión de información entre las mismas. Los Intends también se utilizan para iniciar servicios o llamar a otras aplicaciones.

Podemos decir que los Intends son herramientas que permiten construir puentes entre los distintos procesos de una aplicación Android.

• **Content Provider :** Es la forma que tiene Android de compartir datos entre aplicaciones, puedes definir datos públicos para otras aplicaciones y obtener datos de otras aplicaciones del sistema (por ejemplo: lista de contactos)

• **Services :** Son servicios a la antigua usanza. Son procesos que corren en segundo plano y pueden estar corriendo un largo periodo de tiempo. Hay dos tipos de servicios: locales y remotos. Los servicios locales son componentes que solo son accesibles desde la misma aplicación. Los servicios remotos pueden ser accedidos por otras aplicaciones del terminal. Estos services pueden ser de dos tipos:

- **Started**: Son iniciados por algún componente y su ejecución es independiente del componente que lo ha iniciado, pudiendo continuar su ejecución aunque el componente que lo ha iniciado ya no exista, por ejemplo una actualización de una aplicación.
- **Bound**: Son iniciados cuando algún componente se "ata" a este servicie, y nos proporcionará una suerte de cliente-servidor que permite que los componentes se comuniquen con el service, etc… Es posible "atar" varios componentes a un mismo service, pero estos services se destruirán cuando el componente al que estaba "atado" se destruya.

• **AndroidManifest.xml :** Es el archivo de configuración de la aplicación, si vienes de J2EE es algo parecido al web.xml. Guarda el nombre de tu aplicación, los servicios que tiene, las actividades que tiene, mucha información en este archivo...

• **Android Virtual Devices :** No todo el mundo tiene un teléfono con Android y los AVD son máquinas emuladas con el SDK de Android que te permiten probar en diferentes versiones de Android el programa que estás haciendo.

- **Widget**: Son pequeños componentes que normalmente se muestran al usuario en pantalla y muestran cierta información, permitiendo al usuario realizar ciertas acciones. Por ejemplo un listado de mensajes enviados o de correos recibidos, el tiempo atmosférico, un reloj,etc… Pueden incluso ser embebidas dentro de otras aplicaciones y son susceptibles de actualizaciones.

Ideas previas

Motores Gráficos

Un motor gráfico es el entorno donde el programador diseña un juego ya sea 2D o 3D, éste tiene las herramientas necesarias que equipa al juego de las características físicas (como gravedad, fluidos, luz, sonido, texturas y muchas cosas mas).

Para desarrollar la aplicación el principal problema es la forma de realizar la animación. Si bien desde la API de Android el cargar una animación es muy fácil, hacerlo de forma que se pueda controlar el número de repeticiones y el frame en el que se encuentra la animación ya no lo es.

Comparativa motores gráficos

1. Rokon – Motor de desarrollo de código abierto para juegos en 2D sobre Android
Rokon es de código abierto (licencia BSD), para creación de juegos OpenGL en dos dimensiones, potente y flexible. Rokon ha sido reescrito desde cero. Muchas más características están integradas, y tiene vistas de hacer méritos en un futuro no muy lejano. Ejemplos, tutoriales y documentación están disponibles para todos. Con la ayuda de libgdx y Box2D los desarrolladores del proyecto te obsequian con un motor completo, detallado, escrito en código nativo.

Webs del proyecto: http://rokonandroid.com/ y http://code.google.com/p/rokon/

2. Libgdx – Marco de desarrollo (framework) de juegos Android
Libgdx es un proyecto de código abierto Android, basado en una librería de juegos multi-plataforma de desarrollo en Java. Permite crear prototipos y desarrollar una aplicación. Se puede empezar con tan sólo 6 líneas de código.

Web del proyecto: http://code.google.com/p/libgdx/

3. Android-2D-Engine – Motor de desarrollo para juegos 2D sobre Android en c + + / Java
Android-2D-Engine es un proyecto de código abierto para juegos 2D sobre Android todavía en construcción, el proyecto nacido para servir de base para los juegos. En realidad hay dos proyectos:
trunk / bullet: Contiene el código fuente C++. Este código no es realmente necesario ya que la muestra contiene la librería compilada.
trunk / androgine: Contiene el resto del código necesario para las comunicaciones, además de una muestra con fines de perfiles.

Web del proyecto: http://code.google.com/p/android-2d-engine/

4. AndEngine – Entorno de desarrollo Android para juegos OpenGL 2D
Las características principales incluyen:
* Android optimizado
* Compatibilidad con Android 1.6
* SplitScreen
* Red Multijugador
* Live Wallpapers
* MultiTouch
* Box2D

Web del proyecto: http://code.google.com/p/andengine/

5. Angle - Motor GL Android

Angle es otro motor de desarrollo para juegos 2D con OpenGL destinado a Android. El motor está completamente codificado en java, por lo que puede usar todos los objetos para su conveniencia. Con el motor se incluyen una serie de tutoriales que muestran cómo usarlo.

Web del proyecto: http://code.google.com/p/angle/

6. jPCT-AE: Un motor de 3D gratis para Android

jPCT es un motor 3D gratis, pequeño, rápido y fácil de aprender para Java. Ofrece soporte para el software y el uso de hardware. jPCT ofrece las características necesarias para escribir juegos en tres dimensiones o aplicación en Java en un corto período de tiempo.

Web del proyecto: http://www.jpct.net/jpct-ae/

7. Dwarf-fw: Marco de desarrollo (framework) Android 3D

Dwarf-FW, de código abierto, es un marco / motor de desarrollo para juegos, con uso de OpenGL ES. Está desarrollado para funcionar con el teléfono Android Dev 1 que cuenta con un acelerómetro, magnetómetro, pantalla táctil y rueda de desplazamiento. Algunas de las características actuales:

* Animaciones de fotogramas clave (sin transformación)
* Picking
* OBJ importador
* Binario importador / exportador
* Iluminación
* Materiales
* Filtrado simple de sensores
* VBO soporte

Web del proyecto: http://code.google.com/p/dwarf-fw/

8. YoghurtGum – Multiplataforma C + + 2D, motor para dispositivos móviles
El proyecto de código abierto YoghurtGum es un motor 2Dque pretende hacer el desarrollo de juegos en dispositivos móviles sea fácil, divertido y rápido. Actualmente YoghurtGum está siendo desarrollado para Android, y pronto para Windows Mobile 6. En Android se utiliza OpenGL para hacer cosas en la pantalla mientras que en Windows Mobile 6 se usará DirectX.
Inicio del proyecto: http://code.google.com/p/yoghurtgum/

9. Forget3D: Un marco de OpenGL ES
Forget3D (OpenGL ES A Framework) es un marco de OpenGL ES (todavía no es un motor) para Android, Win32, WinCE plataforma, y simplifica su programa de desarrollo de OpenGL ES, gestión de escenas de apoyo, la textura, la cámara , Luz, modelo de cargador, fuente, etc.

Web del proyecto: http://code.google.com/p/forget3d/

10. Mages Game Engine
Mages Game Engine permite desarrollar eficaz multijugador cliente / servidor de juegos de internet para dispositivos móviles con el mínimo esfuerzo. Permite a los desarrolladores para crear juegos multijugador por internet mediante la aplicación de la lógica e interfaz gráfica de usuario mediante el uso de API del motor de gran alcance. Se puede realizar tareas comunes juego como acceso al servidor del juego, la recuperación de la lista de jugadores activos, la lista de sesiones de juego disponibles, crear nueva sesión de juego, unirse a juegos existentes, invitar a otros jugadores, charlar con los oponentes y muchas otras características.

11. JMonkey Engine - Motor de desarrollo Java, OpenGL
Realiza juegos en modernas plataformas. Programación en Java, en un tiempo record puedes tener tu juego en el mercado, desarrollado sin limitaciones.

Conclusiones

Tras observar las diferentes alternativas de motores gráficos he optado por no utilizar ninguno de ellos, ya que mi juego no utilizara unos gráficos de alta calidad y con las librerías graficas de android me serán suficientes para la realización del videojuego

Las librerías graficas en Android: OpenGL ES y Canvas

Android proporciona esencialmente dos librerías para tratar con gráficos, una orientada principalmente a gráficos 2D llamada Canvas y otra orientada tanto a 2D como a 3D con OpenGL ES 1.0.

Canvas:

Canvas es una librería que solamente va a permitir trabajar con gráficos 2D pero muy útil para proyectos pequeños a medianos. Canvas funciona como una capa que captura todas las llamadas a “draw” y pinta sobre un Bitmap que es la pantalla. Según las necesidades de rendimiento se puede elegir entre pintar creando una View modificada o en un nuevo hilo de ejecución y un SurfaceView.

SurfaceView: Esta es una forma que da especialmente buenos resultados en juegos que necesitan un nivel bastante alto de potencia a la hora de repintar la pantalla y que sirve para casi cualquier juego en 2D

Es la que utilizare para la creación del juego

Manual Usuario

Nadal´s wall

Introducción

Nadal´s Wall es un videojuego para terminales Android. En el manejamos un jugador de tenis, en nuestro caso Rafa Nadal, el cual dando golpes con su raqueta a una pelota ira destruyendo un muro hasta que llegue al codiciado ladrillo dorado, el cual, al ser golpeado por la pelota obtendrás el triunfo.

Pero ten cuidado porque algunos muros al ser golpeados se moverán obstruyendo el paso de la pelota

Los movimientos de Nadal serán movimientos laterales realizados con el acelerómetro de tu smartphone

El juego consiste en golpear el ladrillo dorado intentando conseguir la máxima puntuación

Pantalla de inicio

El juego tiene una pantalla de presentación con un botón para iniciar la partida y otro para salir

Entidades

- **Nadal:** Es el muñeco controlado por el jugador a través del acelerómetro

- **Ladrillo:** Son los objetos que Nadal debe ir destruyendo para obtener puntos

- **Ladrillo dorado:** Es el ladrillo que deberás destruir para superar el juego

- **Ladrillo solido:** Este no podrá ser destruido y además al ser golpeado por la pelota cambiara su posición verticalmente hacia abajo tantas veces como sea golpeado hasta llegar como máximo a la altura de Nadal obstaculizando así su golpeo

- **Pelota:** Es el objeto (pelota) que interactúa entre Nadal y los ladrillos y el que se encarga de destruir estos últimos

Controles

Para iniciar el saque lo único que hay que hacer es pulsar cualquier parte de la pantalla y la pelota se empezara a mover.

Tras el comienzo del movimiento de la pelota, deberás ir moviendo al Nadal usando el acelerómetro de tu terminal móvil de un lado a otro consiguiendo que la pelota no se caiga e intentando destruir el mayor número de ladrillos

El fin de la partida llegara cuando agotes tus tres vidas

Colisiones

En Nadal´s wall hay 3 tipos de colisiones:

- **Nadal-Pelota:**

Cuando colisiona nadal con la pelota esta sale disparada según la dirección aplicada.

- **Pelota-Ladrillo:**

La colisión de la pelota con el ladrillo produce la destrucción de este último y la pelota sigue otra trayectoria. Obteniendo 100 puntos por cada destrucción

- **Pelota-Ladrillo solido**

La colisión de la pelota con este tipo de ladrillos supone el cambio de posición del ladrillo verticalmente hacia abajo tantas veces como sea golpeado hasta llegar a la altura de nuestro personaje

- **Pelota-Ladrillo dorado**

La destrucción de este ladrillo generara la victoria del usuario

- **Pelota-Pantalla**

Cuando colisiona la pelota con los límites de la pantalla esta sale rebotada.

Manual Usuario

Nadal and Federer

Introducción

Nadal and Federer es un videojuego para terminales Android. En el cual podemos echar un partido de tenis entre los dos astros de la raqueta, Rafael Nadal y Roger Federer. Es un juego de entretenimiento en el que tendrás que manejar a estos dos tenistas para que la pelota no se salga de la pantalla

Maneja con habilidad a los dos jugadores pues tan solo tiene tres intentos

Los movimientos de Nadal y Federer serán movimientos laterales realizados con el acelerómetro de tu smartphone

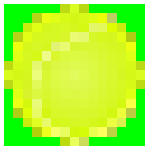
Mantén la pelota el mayor tiempo sin que se salga del terreno de juego

Entidades

- **Nadal:** Es uno de los muñecos controlado por el jugador a través del acelerómetro.

- **Federer:** Es el otro tenista controlado también con el acelerómetro

- **Pelota:** Es el objeto (pelota) que interactúa entre Nadal y los ladrillos y el que se encarga de destruir estos últimos



Controles

Para iniciar el saque lo único que hay que hacer es pulsar cualquier parte de la pantalla y la pelota se empezara a mover.

Tras el comienzo del movimiento de la pelota, deberás ir moviendo al Nadal y Federer usando el acelerómetro de tu terminal móvil de un lado a otro consiguiendo que la pelota no se salga del terreno de juego.

El fin de la partida llegara cuando agotes tus tres vidas

Manual para desarrolladores

Para comenzar a desarrollar aplicaciones para Android desde el PC es necesario instalar varios componentes que nos den el soporte necesario para llevarlo a cabo.

En un resumen rápido, debemos instalar Eclipse, el SDK de Android y descargar el plugin de Android para Eclipse.

• **Eclipse:** Es un IDE *(Integrated Development Environment)* que sirve como interfaz gráfica diseñada para facilitar la gestión y el desarrollo de aplicaciones.

• **SDK:** Es el kit de desarrollo de software y son el conjunto de herramientas que permiten al programador crear aplicaciones para un sistema concreto. En nuestro caso particular, empleamos el SDK de Java, que da soporte a Android.

• **Plugin de Android:** El ADT *(Android Development Tools)* es un plugin para el IDE Eclipse que está diseñado para proporcionar un entorno de desarrollo integrado con el que construir aplicaciones Android.

Es de libre distribución, por lo que no hay problemas de obtener las últimas versiones en las páginas de los fabricantes.

Configuración Entorno Desarrollo

1. Descargar e Instalar Eclipse.

a. Página Web de Eclipse. (Enlace)

b. La distribución Galileo o Indigo puede ser una muy buena opción.

c. Una vez descargado, descomprimimos el fichero y ejecutar “Eclipse.exe”.

d. Elegiremos un directorio para utilizarlo como entorno de trabajo y cuando se instale se podrá iniciar el programa.

2. Descargar el SDK

a. Se podrá obtener fácilmente desde la página para desarrolladores de Android.

b. Habrá que indicar la ruta donde instalarlo y ya estará listo para su uso.

3. Descargar el plugin de Android para Eclipse

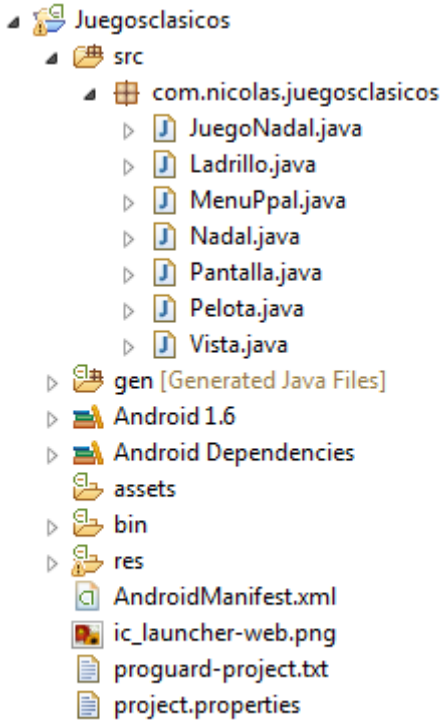
a. Dentro de Eclipse, iremos a Help -> Install New Software

b. Introducir https://dl-ssl.google.com/android/eclipse , seleccionamos e instalaremos**“Developer Tools”** e instalarlo.

4. En Eclipse, debemos indicarle donde se halla el SDK que hemos descargado. Para ello, hay que ir a “Window -> Preferences” y en la opción

Android, rellenar donde se encuentra el SDK Location instalado en el PC. Vuestra ruta será similar a esta: C:\eclipse\android\android-sdk-windows

Estructura del juego



Clases

•Ladrillo:

Son los objetos que tendremos que ir golpeando

En ella lo que hago es poner la pantalla como una matriz en la cual habrá 24 columnas por 16 filas, en la cual va a estar llena de 0´s excepto donde quiero poner un ladrillo en la cual habrá un 1, un ladrillo solido pondré un 2 y en el ladrillo dorado pondré un 3

En el constructor lo que hago es llamar a una función **cargarLadrillos ,** la cual es la función encargada de almacenar las imágenes de los ladrillos a través de una clase **BitmapFactory**

También llama a la función **setEstiloTexto** en la cual, podremos modificar las propiedades de la clase **Paint** anteriormente creada con su constructor, que será la encargada de mostrarnos el texto por pantalla

Un problema que se me produjo fue que dependiendo del tamaño de la pantalla de cada Smartphone Android el conjunto de ladrillo de no se vería bien ya que si le daba una altura y anchura determinada en otro dispositivo con diferentes dimensiones de pantalla no se vería bien por lo cual tuve crear un procedimiento **redimensionarLadrillos**

Este procedimiento **redimensionarLadrillos,** le paso dos parámetros que

son la altura y anchura de la pantalla y lo que hace es me redimensiona los ladrillos para ajustarlos a los diferentes tamaños de las pantallas.

La función **actualizar** es la encargada de llamar a la función **onDraw** de la clase **SurfaceView** . Esta función **actualizar** es la encargada de recorrer toda la matriz anteriormente definida y donde aparezca un 2 dibujara un ladrillo solido, donde haya un 3 un ladrillo dorado y donde este un 1 un ladrillo normal

También se encarga de redibujar el fondo de la pantalla, mostrar la puntuación y devuelve un tipo booleano por si queda algún ladrillo o no.

También tendremos que detectar las colisiones para ello nos creamos una función **colisiones** que será la encargada de determinar si el objeto que se le pasa como parámetro colisiona en las coordenadas (x,y) con un ladrillo, si esto ocurre devuelve true y borra de la matriz poniendo sus coordenadas (i , j) de la matriz a cero para que cuando redibuje la matriz ya no lo muestre

- **Nadal:**

Es el objeto donde tendremos que intentar que rebote la pelota para que no caiga al suelo

En su constructor lo único que hago es cargar la imagen de Nadal a través de **BitmapFactory** como hice en la clase **ladrillos**

Tras conocer las dimensiones de la pantalla llamo a la función **setPosition** (nos permite fijar la posición (x,y) de la pantalla) en la función **SurfaceChanged** para situar a Nadal en cualquier parte en la pantalla, lo pondré centrado en la anchura y a una decima parte de la altura de la pantalla

También tendré la función **actualizar** la cual es la encargada de pintar el objeto Nadal, comprueba que no salga de los límites de la pantalla así como de actualizar el estado de Nadal

También hay una función **colisión** que indica si el objeto que se encuentra en unas coordenadas (x,y) que se le pasan como argumentos produce una colisión con Nadal o no.

Es decir se encarga de cuando la pelota debe rebotar.

•Pelota

El objeto pelota que será el encargado de ir destruyendo los ladrillos y ser golpeado por Nadal

En el constructor cargo la imagen de la pelota y además le paso las instancias de los ladrillos y Nadal creadas en la clase **Pantalla** (**SurfaceView**) para que puedan llamar a los métodos de estas (ladrillo y Nadal) para saber cuándo debe haber un cambio de dirección o debe seguir Nadal porque el juego no ha empezado y no se ha producido ningún movimiento.

El procedimiento **actualizar** es el encargado de actualizar el estado de la pelota

En la cual se comprueba a partir de un tipo boolean si estamos jugando, luego compruebo si colisiona con los bloques del eje X y eje Y, llamando a la función colisiones de la clase **Ladrillos** anteriormente vista.

Luego compruebo si colisiona con el objeto Nadal, a través de la función colisión de la clase **Nadal** en el cual rebotara la pelota o no. Actualiza las posiciones y redibuja la pelota.

•Pantalla

Esta clase hereda de la clase SurfaceView e implementa SurfaceHolder. Callback. La clase SurfaceView es el punto de partida para la construcción de juegos que necesitan un nivel bastante alto de potencia a la hora de dibujar gráficos 2D o 3D.

Para hacer uso del objeto Surface, es necesario hacerlo a través de un contenedor o SurfaceHolder e indicar posteriormente que dicho contenedor va a recibir las llamadas del SurfaceHolder. Callback. Estas tareas se realizan en el constructor de dicha clase y en el método cambia.

Además de dicho método, se implementan tres métodos más, propios del interfaz SurfaceHolder. Callback:

public void surfaceCreated (SurfaceHolder holder)

Inicializa la superficie donde se va a dibujar el videojuego de plataformas.

public void surfaceDestroyed (SurfaceHolder holder)

Detiene el hilo principal y libera los recursos de la aplicación.

public void surfaceChanged (SurfaceHolder holder, int format, int w, int h)

Se invoca al cambiar el dispositivo de landascape a portraite. En este caso no se implementa debido a que la aplicación siempre será en un sentido de orientación.

public void run ()

Este método se encarga de manejar todo el hilo de ejecución del videojuego. En él se inicializan todos los objetos necesarios para el correcto funcionamiento del mismo, y se ejecuta un bucle, que se encarga de llamar a los métodos lógica y dibujar, y de controlar el número de frames por segundo que la aplicación tarda en realizar estas dos tareas, hasta que el usuario ordena parar el videojuego, es decir, configura la variable muerto a true.

Tratamientos de colisiones

Hay 2 tipos de colisiones que se pueden producir:

Colisión Ladrillo-Pelota

Función que determina si la pelota en la coordenadas x e y colisiona con algún objeto

public boolean colisiones(int x, int y)

Primero divido la coordenada del eje Y entre la altura de mi ladrillo calculada en el método redimensionar y la guardo en una variable entero

I=y/alto

Lo mismo hago para la coordenada de eje X

j=x/ancho

Luego veo si se sale de los límites de la pantalla y devuelvo el boolean a true

If (i<0 || i>9 || j<0 || j>15)

return true

Si colisiona contra un ladrillo normal lo borro poniendo un cero en su lugar para que cuando lo vuelva a redibujar aparezca un cero y no lo dibuje y también aumentamos la puntuación. Y devuelvo true

If(muro [i] [j] == 1)

muro [i] [j]=0

puntuación +=100

return true

Si colisiona con un ladrillo solido pueden pasar 2 cosas:

- Si el ladrillo está por encima de la fila 10 hago que baje una posición y devuelvo true .Si está en la fila 10 lo dejo ahí.

If(muro[i] [j] == 2)

If(i==10)

matriz [10] [j]=2

return true

}else{

muro[i] [j] == 0

muro[i+1] [j] == 2

Colisión Nadal-Pelota

Detecta si se ha producido una colisión entre Nadal y la pelota

Public boolean colision (int x, int y)

Tiene que cumplir 3 las condiciones para que haya colisión

- Que la posición X donde se encuentra Nadal menos el centro de la anchura del dibujo de Nadal sea menor que la posición x donde se encuentra la pelota

pos_x – (nadal.getWidth() / 2) < x

- Que la posición X donde se encuentra Nadal mas el centro de la anchura del dibujo de Nadal sea mayor que la posición x donde se encuentra la pelota

pos_x + (nadal.getWidth() / 2) > x

- Que la posición Y donde se encuentra Nadal menos el centro de la altura del dibujo de Nadal sea menor que la posición y donde se encuentra la pelota

pos_y – (nadal.getHeight() / 2) < y

Acelerómetro

Hasta ahora no tenía nada implementado con el acelerómetro y vamos a empezar en ello. Vamos a obligar que el objeto Nadal se muevo con los sensores del terminal móvil.

Lo primero que hacemos en la clase **pantalla** que extiende de la **SurfaceView** es que implemente a la interfaz **SensorEventListener** que nos obliga a implementar los métodos **onAccuaracyChanged** y **onSensorChanged**

Antes de implementar estos métodos primero voy a obtener todos los sensores del Smartphone

SensorManager sensores = (SensorManager) getSystemService (Activity.SENSOR_SERVICE)

Ahora obtengo solo los del acelerómetro

List<Sensor> listaSensores = sensores.getSensorList (Sensor.TYPE_ACCELEROMETER)

Y obtengo el objeto que representa a el acelerómetro

Sensor acelerómetro = listaSensores.get(0)

Ahora pongo que el objeto pantalla sea quien maneje los eventos

Sensores.registerListener((SensorEventListener)pantalla, acelerómetro, sensores.SENSOR_DELAY_UI)

Una vez obtenido el acelerómetro del Smartphone hay que implementar los métodos creados por la interfaz **SensorEventListener.** Son **onAccuaracyChanged** y **onSensorChanged**

onSensorChanged es llamada cuando cambia la medición del acelerómetro

Solo me interesa la función **onSensorChanged** donde obtengo un array de 3 posiciones de lo que está leyendo el acelerómetro en cada uno de sus ejes.

Cogemos las coordenadas Y las multiplico por 6 para obtener una respuesta mejor

public void onSensorChanged (SensorEvent event)

synchronized(this)

if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER)
acelerometro = (int)(event.values[1] * 6)

SONIDOS

Para la reproducción de los diferentes sonidos que tiene el juego utilizo el MediaPlayer, ya que es muy fácil de utilizar y puedo reproducir los mp3

Para la obtención de estos diferentes sonidos utilice una biblioteca de sonidos que me proporciono http://www.4shared.com/

En la cual encontré multitud de sonidos para implementarlos en mi juego

Los sonidos se guardan en la carpeta res (recursos) y dentro de ella en la subcarpeta raw donde almacenare todos los sonidos que iré usando a lo largo del juego

Una vez integrados en el juego el siguiente paso es reproducirlo en un momento determinado

Primero me creo una variable del tipo MediaPlayer en la clase donde quiera que se vaya a reproducir

prívate MediaPlayer reproductor

Luego en su constructor lo inicializo, en nuestro caso el sonido que tenemos guardado en la carpeta raw se llama aplausos

reproductor = MediaPlayer.create(context,R.raw.aplausos)

reproductor.setLooping(false)

La última línea es para decirle que no sea un bucle, ya lo tenemos preparado para reproducir cuando queramos el sonido. Lo único que habría que añadir es

reproductor.start(), en el momento que queramos reproducirlo

IMÁGENES

Una vez que tengo mis imágenes las guardo en la carpeta res/drawable para su posterior uso

 Obtengo un mapa de bits, a partir de una imagen localizada en la carpeta res/drawable, para transformarlo en una textura mediante la sentencia:

Bitmap bmp = BitmapFactory.*decodeResource* (Resources res, int id)

Donde **res** es el recurso de la aplicación que contiene los datos de la imagen deseada e **id** el identificador de dichos datos.

FASES DEL DISEÑO

Creación de los ladrillos en la pantalla y el texto para la puntuación

Añado la imagen de Nadal y el acelerómetro

Añado la pelota y la interacción con los demás objetos

Análisis temporal y de costes

Tiempo dedicado en cada sección

En esta sección vamos a hacer un repaso sobre el coste en tiempo de las diversas partes del proceso de elaboración de “Nadal´s Wall”. En primer lugar, vamos a elaborar estas secciones, detallando lo que se incluye en cada una.

En cuanto a los costes como me réferi en apartados anteriores, quería crear una aplicación ,la cual no me supusiese muchos gastos y así ha sido, ya que el Sony vaio y el htc desire ya los tenias adquiridos y las herramientas utilizadas son de libre adquisición por lo tanto el desembolso para hacer este proyecto ha sido 0 euros

Costes= 0 euros

● Investigación:

○ El aprendizaje sobre la estructura de un desarrollo en Android.

○ El aprendizaje sobre el manejo de las herramientas de desarrollo (Eclipse + SDK).

○ La búsqueda, análisis y comparativa de frameworks / motores, incluyendo la investigación tanto de posibilidades 2D o 3D, y el posterior aprendizaje sobre la alternativa escogida.

○ La búsqueda, análisis y comparativa de opciones para el diseño de escenarios y su posterior implementación en el juego, así como el aprendizaje sobre la herramienta escogida.

Total: 17 días

● Desarrollo de la aplicación:

○ El desarrollo de la idea inicial y los sucesivos refinamientos en el concepto del juego.

○ La implementación de la estructura de clases del proyecto.

○ La elaboración del código.

○ El diseño artístico, tanto en el aspecto gráfico como el musical.

Total: 38 días

● Elaboración del documento:

○ Recopilación de información relativa a aspectos relacionados con ciertas secciones (gráficas, datos objetivos, etc..)

○ Redactado del documento con síntesis de conceptos.

○ Toma de capturas de pantalla.

○ Realización de gráficas.

Total: 15dias

● Pruebas y testeo:

○ Análisis de la jugabilidad del título y el correcto funcionamiento

○ Recopilación de opiniones externas sobre aspectos técnicos y jugables.

○ Pruebas de rendimiento en distintos dispositivos Android.

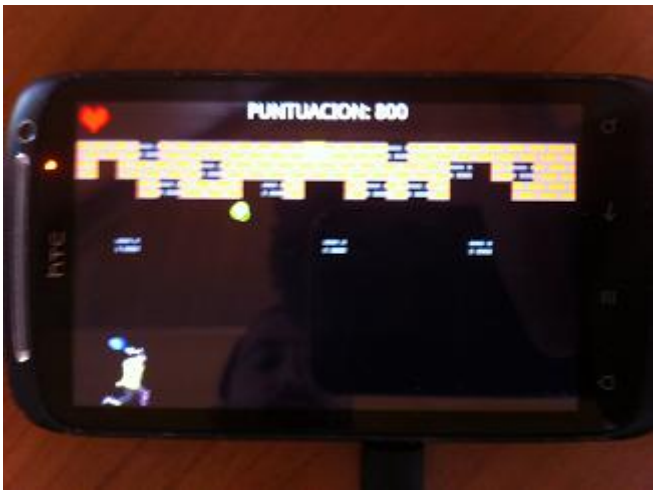
Total: 7 días

Total de días para la realización del proyecto: 67 días x 8 horas= 536h

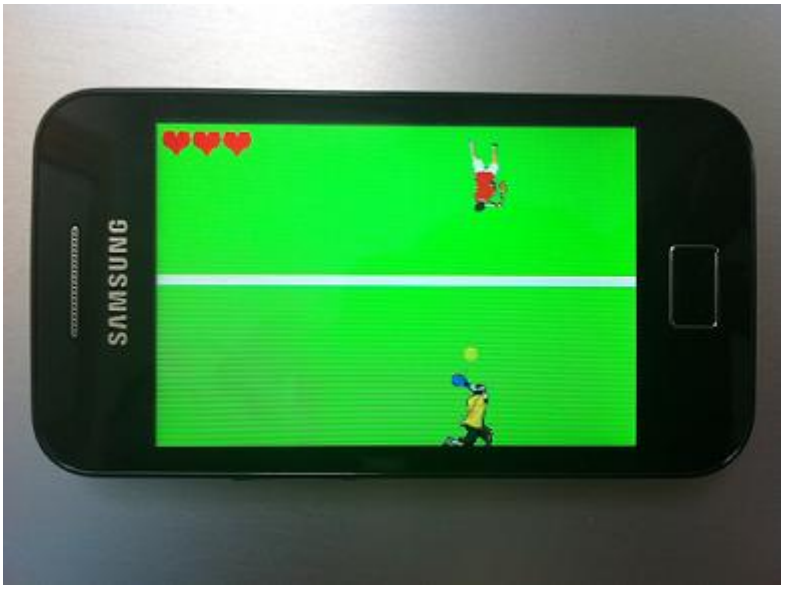
Pruebas de funcionamiento

Las pruebas fueron realizadas en un htc desire y en un Samsung galaxy sin ningún tipo de problemas

Aquí muestro imágenes de **Nadal´s wall** sobre una Htc Desire y sobre un Samsung Galaxy



Y aquí una imagen de la aplicación Nadal and Federer



Comparación con otros juegos similares

Arkanoid

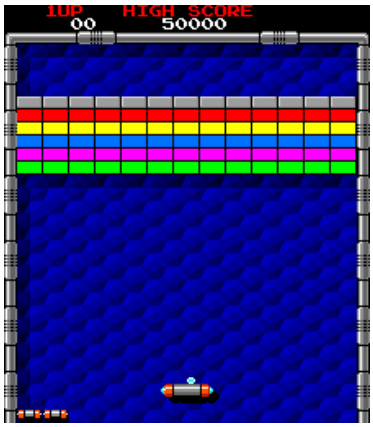
Un juego “sin trama”, pero no la necesita. La historia explica que una nave espacial llamada Arkanoid es destruida y solo sobrevive el Vaus que es como una baliza espacial la cual, es tu herramienta para jugar (la “tablita” o el “palito” en el que rebotan las pelotitas) y queda perdida en el espacio.

Tu objetivo es derrotar al, maligno DOH, que es una especia de estatua gigante de la isla de pascua (llamados moai) y tal vez así logres solucionar tu situación espacial precaria.

La mecánica del juego es también simple, puedes mover el Vaus de derecha a izquierda para lograr hacer rebotar sobre éste una pelotita que, al tocarlas, irá destruyendo las estructuras rectangulares que se encuentran en la parte superior. Cuando destruyes todas estas estructuras, pasas el nivel. Si dejas caer la pelotita, pierdes una vida.

El nivel de dificultad va aumentando con cada nivel, ya sea por la introducción de bloques indestructibles y su configuración (como están acomoda-

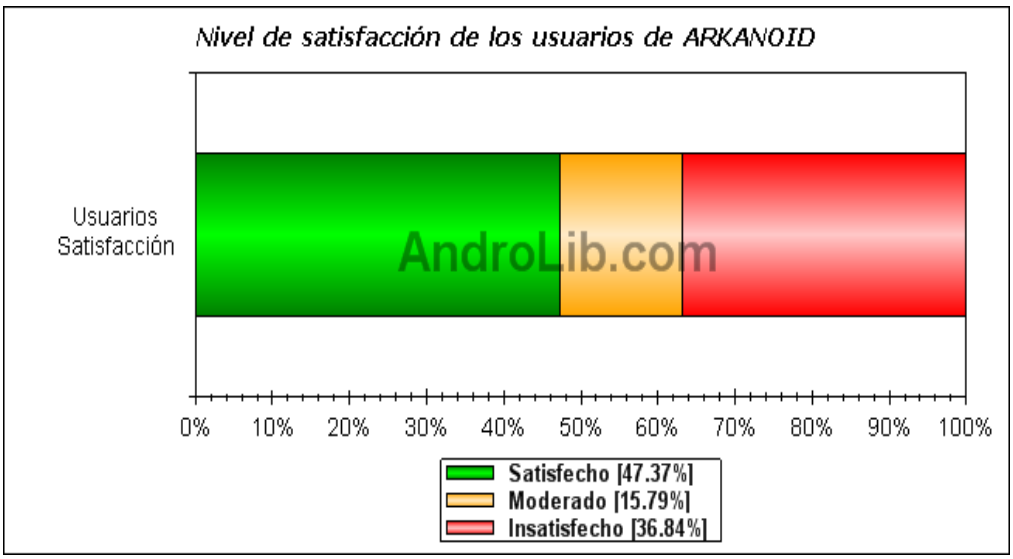
dos). O por la velocidad de las pelotitas que adquieren al rebotar varias veces en un área encerrada.



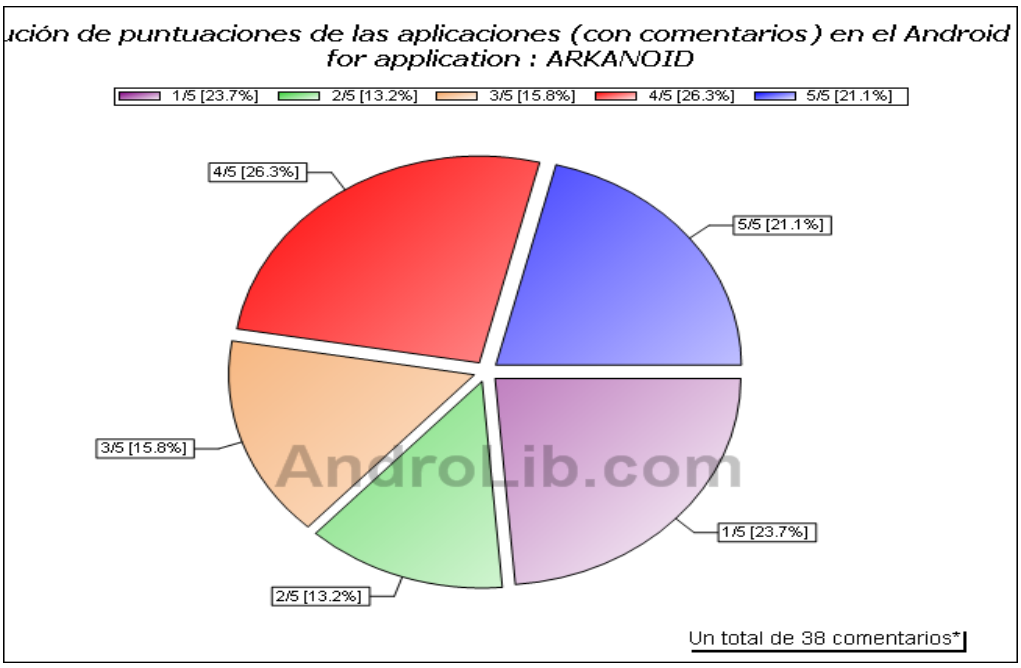
No estás solo, durante todo el juego al destruir algunas estructuras caerán ciertos “items” que te “beneficiarán”. Los efectos pueden ser:

- hacer el Vaus más largo
- otorgarte más pelotitas
- destruir los bloques fuertes con un solo golpe
- lanzar un laser desde el Vaus
- hacer más chico el Vaus
- incluso hacerte pasar al próximo nivel automáticamente.

En el original eran 33 niveles, y en el final te enfrentabas al temido DOH. Por supuesto que hubo muchas versiones para otras plataformas, en las que como siempre podían variar algunas funcionalidades o bonus, o configuración de los escenarios. Pero básicamente la mecánica se mantuvo fiel a la versión de arcadia.


Nivel de satisfacción de los usuarios de ARKANOID
Usuarios
Satisfacción
AndroLib.com
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Satisfecho [47.37%]
Moderado [15.79%]
Insatisfecho [36.84%]


Nivel de satisfacción de los usuarios de ARKANOID
AndroLib.com
Nivel de satisfacción
Satisfecho
Moderado
Insatisfecho


ución de puntuaciones de las aplicaciones (con comentarios) en el Android
for application : ARKANOID
1/5 [23.7%]
2/5 [13.2%]
3/5 [15.8%]
4/5 [26.3%]
5/5 [21.1%]
4/5 [26.3%]
5/5 [21.1%]
3/5 [15.8%]
1/5 [23.7%]
2/5 [13.2%]
AndroLib.com
Un total de 38 comentarios*

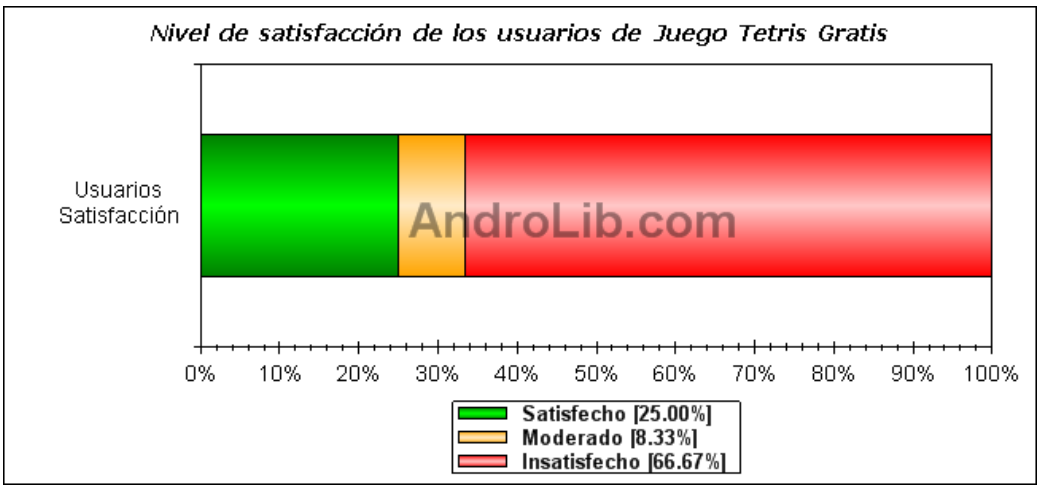
Classic Tetris

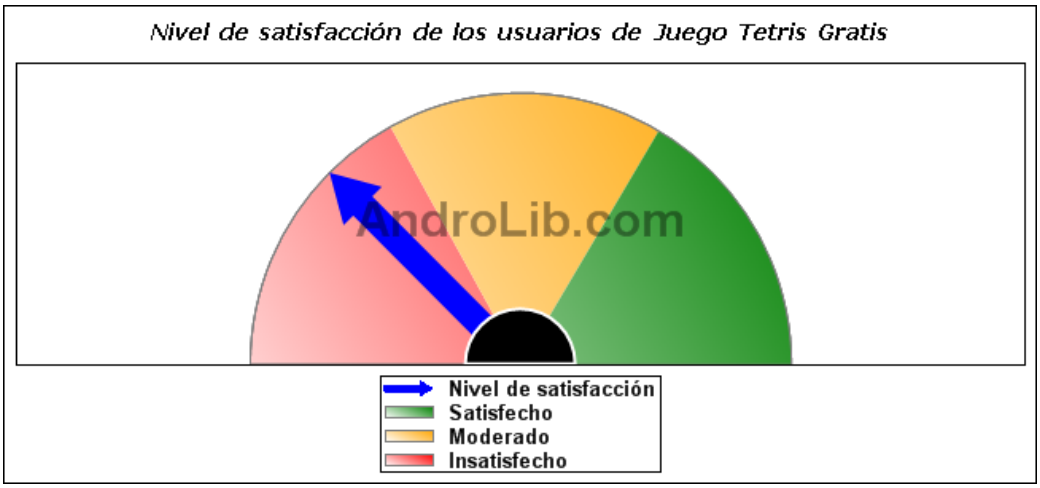
Classic Tetris es, como su mismo nombre indica, el Tetris de toda la vida en nuestro Android.

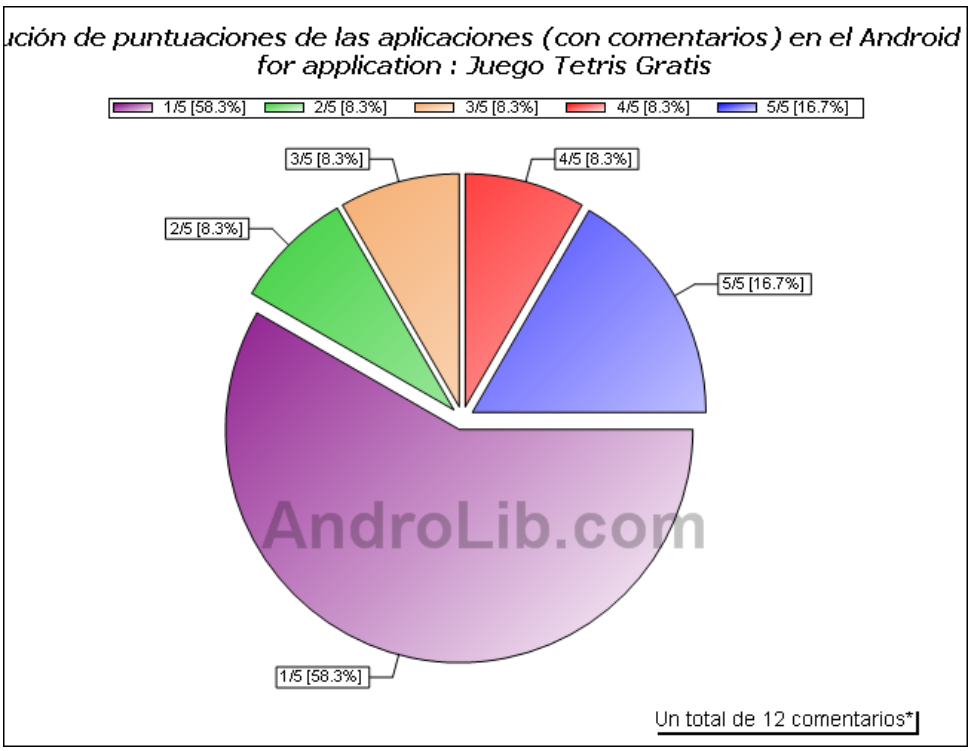
El juego ya lo conocemos todos; van cayendo bloques de distintas formas desde arriba y tenemos que colocarlos para completar líneas que desaparecen, dándonos puntos. El juego cada vez va más rápido y acaba cuando los bloques llegan hasta la línea superior. Lo único que aquí cambia es la forma de manejarlo, y es que no tenemos teclas hacia los lados para hacerlo.

Hay dos formas de jugar. La primera de ellas utilizando la pantalla táctil, con el dedo sobre la pieza que va hacia abajo podremos desplazarlo hacia un lado u otro, y la pieza nos seguirá; pulsaremos sobre ella para hacerla rotar o acompañarla hacia abajo para que caiga más rápido. Esta opción, hasta que no nos acostumbramos, dificulta un poco el juego ya que si se te va el dedo puedes perder toda la partida. Si no, podemos jugar con la trackball, con los mismos movimientos: hacia los lados moveremos la pieza, hacia abajo la aceleraremos, y moviendo hacia arriba la rotaremos. Una vez los controlamos, ambos métodos son muy válidos.

- Precio: Gratuita
- Espacio: 2.2 MB
- Compatibilidad: cualquiera
- Desarrollador: Gameone
- Código QR:


Nivel de satisfacción de los usuarios de Juego Tetris Gratis
Usuarios
Satisfacción
AndroLib.com
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Satisfecho [25.00%]
Moderado [8.33%]
Insatisfecho [66.67%]


Nivel de satisfacción de los usuarios de Juego Tetris Gratis
AndroLib.com
Nivel de satisfacción
Satisfecho
Moderado
Insatisfecho


ución de puntuaciones de las aplicaciones (con comentarios) en el Android
for application : Juego Tetris Gratis
1/5 [58.3%]
2/5 [8.3%]
3/5 [8.3%]
4/5 [8.3%]
5/5 [16.7%]
AndroLib.com
Un total de 12 comentarios*

Conclusiones

Una vez concluido el desarrollo del presente proyecto, se procederá a hacer un balance del resultado final obtenido. Para ello se repasará de manera breve los objetivos personales así como los requisitos técnicos propuestos en el inicio del mismo y se hará una comparación con los resultados finalmente obtenidos.

En cuanto a los objetivos personales se puede afirmar que se han cumplido de manera satisfactoria. A lo largo del proyecto se ha conseguido adquirir un amplio conocimiento de las herramientas necesarias para su construcción.

Entre dichas herramientas cabe destacar la librería grafica de Android de la cual se han adquirido conocimientos acerca de la construcción de figuras sencillas, del manejo avanzado de texturas, del uso de colores e iluminación de la escena y del manejo básico de la cámara y de sus perspectivas.

También se ha conseguido ampliar en gran medida el manejo de la plataforma Android, proporcionando un amplio conocimiento de su funcionamiento interno y de la gestión de recursos y de memoria, así como del entorno de desarrollo avanzado Eclipse, aprendiendo nuevas técnicas para la depuración del código y la validación de los requisitos técnicos mínimos necesarios para el correcto funcionamiento de la aplicación.

En cuanto a los requisitos técnicos se puede afirmar que la mayor parte de ellos se han cumplido de manera satisfactoria, aunque existen ciertos puntos que podrían mejorarse.

Por último, es necesario mencionar que la implementación de este proyecto no se podría haber llevado a cabo sin el enorme esfuerzo realizando estos últimos años, en los que cursé la carrera de Ingeniería Técnica de

Informática de Sistemas, y que me han permitido adquirir grandes conocimientos tanto técnicos como personales.

Bibliografía

Desarrollo de aplicaciones android para dispositivo móviles
http://e-archivo.uc3m.es/bitstream/10016/6506/1/PFC_Jaime_Aranaz_Tudela_20101116132629.pdf

Desarrollo android
http://arduinoyandroid.blogspot.com.es/2012/03/como-instalar-eclipse-para-desarrollar.html

Aprendiendo de android
http://droideando.blogspot.com.es/2011/03/introduccion-andengine-parte-i.html

Preparación del pc para el desarrollo de android
http://www.movilzona.es/android-preparacion-del-pc-para-usar-android/

Conocimientos de android
http://www.androidsis.com/programacion-android-iii-interfaz/

Blog sobre desarrollo de juegos
http://blog.vidasconcurrentes.com/android/creando-una-aplicacion-de-android-primeros-pasos/

Blog sobre android
http://www.androidstartup.com/

Obtención de sonidos
http://www.4shared.com/

Información sobre android
http://www.android.es/category/tutoriales#axzz25UhtfrRi

Manuales sobre android
http://www.android-spa.com/

Tutoriales android
http://www.demonhades.org/foro/

Tutoriales android
http://android-latino.blogspot.com.es/

Análisis juegos android
http://www.mundoandroides.com/juegos

Aplicaciones realizados por alumnos de la us
https://forja.rediris.es/