ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Aplicación Android para el seguimiento de Cetáceos

AUTOR: María Fernanda Suárez Vélez

DIRECTOR: José Luis Gómez Tornero

CODIRECTOR: Simon Pomeroy

Julio **/** 2013

I. INDICE GENERAL

II. ÍNDICE DE IMÁGENES

Introducción

El propósito del proyecto es desarrollar una aplicación basada en el sistema operativo Android que ofrezca ayuda en el seguimiento de delfines en el mar, mediante el cálculo de la distancia donde aparece el animal. Esto se consigue haciendo uso de la Realidad Aumentada y con el empleo del acelerómetro, la brújula, la cámara y el GPS de la tablet, junto con su altura sobre la superficie del mar y el ángulo de inclinación del dispositivo, que es usado para calcular la distancia donde emerge el delfín.

Se comenzará con el estudio del sistema operativo Android, la instalación del entorno de programación y la realización de las primeras aplicaciones para afianzar conocimientos. Más tarde se pasará al estudio del acelerómetro, brújula, GPS, y cámara de la tablet para la utilización en la aplicación, junto con el estudio de Realidad Aumentada.

Una vez estudiados los sensores, con los datos de estos se hará el algoritmo que calcule la distancia dónde ha aparecido el delfín.

Finalmente se hará la codificación y diseño de la aplicación.

1. Sistema Operativo Android

1.1 Introducción

Android es una plataforma software basada en Linux para dispositivos móviles que incluye un sistema operativo, middleware y aplicaciones clave.

Está basado en software libre, lo que quiere decir que todo el mundo puede ver el código de las aplicaciones. Por este mismo motivo, todos los usuarios que quieran pueden desarrollar sus propias aplicaciones, ya que Google tiene a su disposición las herramientas necesarias para programar las aplicaciones.

Es aconsejable definir una serie de conceptos para entender la definición de que es y en que se basa Android:

•Un sistema operativo es un conjunto de programas de ordenador destinados a permitir una administración eficaz de sus recursos. Su finalidad es la de gestionar el hardware de la máquina desde los niveles más básicos, permitiendo también la interacción con el usuario.

•Middleware, es un tipo de software o programa de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de las aplicaciones distribuidas sobre plataformas heterogéneas. Se sitúa entre el sistema operativo y las funciones de red del dispositivo. El middleware proporciona una Interfaz de Programación de Aplicaciones, API, que facilita la programación y el manejo de las aplicaciones distribuidas.

Habiendo definido estos dos conceptos entendemos que Android es una plataforma de desarrollo y programación de aplicaciones para dispositivos móviles que incorpora un conjunto de programas, que administran eficazmente sus recursos y le permiten interactuar fácilmente con el usuario, ofreciéndole una serie de servicios de conectividad que el permiten manejar y programar aplicaciones de forma sencilla.

1.2 Historia

Android Inc. fue fundada en Palo Alto, California, en octubre de 2003, con el objetivo de desarrollar un nuevo sistema operativo para teléfonos inteligentes.

Google adquirió la compañía Android Inc. en 2005, por lo que es una subsidiaria de la propiedad absoluta de Google. De esta forma Google se lanzó al mercado de la telefonía móvil con este sistema. De manera que Google comercializa la plataforma con la finalidad de ofertar su producto Android a todos los operadores y fabricantes de teléfonos móviles.

A partir de ese momento, Google inició un proceso de búsqueda de acuerdos con diferentes operadores de comunicaciones y fabricantes de dispositivos móviles. La compara por parte de Google de los derechos de varias patentes relacionadas con tecnologías móviles hizo saltar la liebre y los mercados comenzaron a especular con la inminente entrada de la compañía en el negocio de las comunicaciones móviles.

El 5 de noviembre de 2007, nació la Open Handset Alliance, un consorcio de empresas de tecnología como Google, acompañadas de los mayores fabricantes de dispositivos como HTC y Samsung, los operadores inalámbricos, como Sprint Nextel y T-Mobile y los fabricantes de chips como Qualcomm y Texas Instruments. Este consorcio, formado por más de 70 empresas, es liderado por Google y su objetivo es el desarrollo de estándares abiertos para dispositivos móviles. Bajo esta premisa, Google liberó la mayor parte del código fuente del nuevo sistema operativo Android usando la licencia Apache, una licencia libre, gratuita y de código abierto.

Android fue presentado como una plataforma móvil basada en la versión del kernel de Linux 2.6, siendo un móvil de la marca HTC el primero en ser lanzado con este sistema operativo.

1.2.1 Versiones Android:

Android 1.0 Nivel de API 1 (septiembre 2008)

Primera versión de Android. Nunca se utilizó comercialmente, por lo que no tiene mucho sentido desarrollar para esta plataforma.

Android 1.1 Nivel de API 2 (febrero 2009)

No se añadieron apenas funcionalidades simplemente se fijaron algunos errores de la versión anterior. Es la opción a escoger si queremos desarrollar una aplicación compatible con todos los dispositivos Android. No obstante apenas existen usuarios con esta versión.

Cupcake

Android 1.5 Nivel de API 3 (abril 2009)

Es la primera versión con algún usuario (aunque apenas la usa un 0,1% en enero de 2013). Como novedades, se incorpora la posibilidad de teclado en pantalla con predicción de texto, los terminales ya no tienen que tener un teclado físico, así como la capacidad de grabación avanzada de audio y vídeo. También aparecen los *widgets* de escritorio y *live folders*. Incorpora soporte para *bluetooth* estéreo, por lo que permite conectarse automáticamente a auriculares *bluetooth*. Las transiciones entre ventanas se realizan mediante animaciones.

Donut

Android 1.6 Nivel de API 4 (septiembre 2009)

Permite capacidades de búsqueda avanzada en todo el dispositivo. También se incorpora *gestures* y *multi-touch*. Permite la síntesis de texto a voz. También se facilita que una aplicación pueda trabajar con diferentes densidades de pantalla. Soporte para resolución de pantallas WVGA. Aparece un nuevo atributo XML,

onClick, que puede especificarse en una vista. Play Store antes, Android Market se mejora permitiendo una búsqueda más sencilla de aplicaciones. Soporte para CDMA/EVDO, 802.1x y VPNs. Mejoras en la aplicación de la cámara.

Éclair

Android 2.0 Nivel de API 5 (octubre 2009)

Esta versión de API apenas cuenta con usuarios, dado que la mayoría de fabricantes pasaron directamente de la versión 1.6 a la 2.1. Como novedades cabría destacar que incorpora un API para manejar el *bluetooth* 2.1. Nueva funcionalidad que permite sincronizar adaptadores para conectarlo a cualquier dispositivo. Ofrece un servicio centralizado de manejo de cuentas. Mejora la gestión de contactos y ofrece más ajustes en la cámara. Se ha optimizado la velocidad de *hardware*. Se aumenta el número de tamaños de ventana y resoluciones soportadas. Nueva interfaz del navegador y soporte para HTML5. Mejoras en el calendario y soporte para Microsoft Exchange. La clase MotionEvent ahora soporta eventos en pantallas multitáctil.

Android 2.1 Nivel de API 7 (enero 2010)

Se considera una actualización menor, por lo que le siguieron llamando Éclair. Destacamos el reconocimiento de voz que permite introducir un campo de texto dictando sin necesidad de utilizar el teclado. También permite desarrollar fondos de pantalla animados. Se puede obtener información sobre la señal de la red actual que posea el dispositivo. En el paquete WebKit se incluyen nuevos métodos para manipular bases de datos almacenadas en Web. También se permite obtener permisos de geolocalización, y modificarlos en WebView. Se incorporan mecanismos para administrar la configuración de la caché de aplicaciones,

almacenamiento web, y modificar la resolución de la pantalla. También se puede manejar vídeo, historial de navegación, vistas personalizadas…

Froyo

Android 2.2 Nivel de API 8 (mayo 2010)

Como característica más destacada se puede indicar la mejora de velocidad de ejecución de las aplicaciones (ejecución del código de la CPU de 2 a 5 veces más rápido que en la versión 2.1 de acuerdo a varios *benchmarks*). Esto se consigue con la introducción de un nuevo compilador JIT de la máquina Dalvik.

Se añaden varias mejoras relacionadas con el navegador Web, como el soporte de Adobe Flash 10.1 y la incorporación del motor Javascript V8 utilizado en Chrome o la incorporación del campo de “subir fichero” en un formulario.

El desarrollo de aplicaciones permite las siguientes novedades: se puede preguntar al usuario si desea instalar una aplicación en un medio de almacenamiento externo (como una tarjeta SD), como alternativa a la instalación en la memoria interna del dispositivo. Las aplicaciones se actualizan de forma automática cuando aparece una nueva versión. Proporciona un servicio para la copia de seguridad de datos que se puede realizar desde la propia aplicación para garantizar al usuario el mantenimiento de sus datos. Por último, se facilita que las aplicaciones interaccionen con el reconocimiento de voz y que terceras partes proporcionen nuevos motores de reconocimiento.

Se mejora la conectividad: ahora podemos utilizar nuestro teléfono para dar acceso a Internet a otros dispositivos (tethering), tanto por USB como por Wi-Fi. También se añade el soporte a Wi-Fi IEEE 802.11n y notificaciones push.

Se añaden varias mejoras en diferentes componentes: En el API gráfica OpenGL ES se pasa a soportar la versión 2.0. También se puede realizar fotos o vídeos en

cualquier orientación (incluso vertical) y configurar otros ajustes de la cámara. Para finalizar, permite definir modos de interfaz de usuario (“automóvil” y “noche”) para que las aplicaciones se configuren según el modo seleccionado por el usuario.

Gingerbread

Android 2.3 Nivel de API 9 (diciembre 2010)

Debido al éxito de Android en las nuevas tabletas ahora soporta mayores tamaños de pantalla y resoluciones (WXGA y superiores).

Incorpora un nuevo interfaz de usuario con un diseño actualizado. Dentro de las mejoras de la interfaz de usuario destacamos la mejora de la funcionalidad de “cortar, copiar y pegar” y un teclado en pantalla con capacidad multitáctil.

Se incluye soporte nativo para varias cámaras, pensado en la segunda cámara usada en videoconferencia. La incorporación de esta segunda cámara ha propiciado la inclusión de reconocimiento facial para identificar el usuario del terminal.

La máquina virtual de Dalvik para Android introduce un nuevo recolector de basura que minimiza las pausas de la aplicación, ayudando a garantizar una mejor animación y el aumento de la capacidad de respuesta en juegos y aplicaciones similares. Se trata de corregir así una de las lacras de este sistema operativo móvil, que en versiones previas no ha sido capaz de cerrar bien las aplicaciones en desuso. Se dispone de mayor apoyo para el desarrollo de código nativo (NDK).También se mejora la gestión de energía y control de aplicaciones. Y se cambia el sistema de ficheros, que pasa de YAFFS a ext4.

Entre otras novedades destacamos en soporte nativo para telefonía sobre Internet VoIP/SIP. El soporte para reproducción de vídeo WebM/VP8 y codificación de audio AAC. El soporte para la tecnología NFC. Las facilidades en el audio, gráficos y entradas para los desarrolladores de juegos. El soporte nativo para más sensores

(como giroscopios y barómetros). Un gestor de descargas para las descargas largas.

Honeycomb

Android 3.0 Nivel de API 11 (febrero 2011)

Para mejorar la experiencia de Android en las nuevas tabletas se lanza la versión 3.0 optimizada para dispositivos con pantallas grandes. La nueva interfaz de usuario ha sido completamente rediseñada con paradigmas nuevos para la interacción, navegación y personalización. La nueva interfaz se pone a disposición de todas las aplicaciones, incluso las construidas para versiones anteriores de la plataforma.

Las principales novedades de este SDK son:

Con el objetivo de adaptar la interfaz de usuario a pantallas más grandes se incorporan las siguientes características: resolución por defecto WXGA (1280×800), escritorio 3D con widgets rediseñados, nuevos componentes y vistas, notificaciones mejoradas, arrastrar y soltar, nuevo cortar y pegar, barra de acciones para que las aplicaciones dispongan de un menú contextual siempre presente y otras características para aprovechar las pantallas más grandes.

Se mejora la reproducción de animaciones 2D/3D gracias al renderizador OpenGL acelerado por hardware. El nuevo motor de gráficos Rederscript saca un gran rendimiento de los gráficos en Android e incorpora su propia API.

Primera versión de la plataforma que soporta procesadores multinúcleo. La máquina virtual Dalvik ha sido optimizada para permitir multiprocesado, lo que permite una ejecución más rápida de las aplicaciones, incluso aquellas que son de hilo único.

Se incorporan varias mejoras multimedia, como listas de reproducción M3U a través de HTTP Live Sreaming, soporte a la protección de derechos musicales (DRM) y

soporte para la transferencia de archivos multimedia a través de USB con los protocolos MTP y PTP.

En esta versión se añaden nuevas alternativas de conectividad, como las nuevas APIS de Bluetooth A2DP y HSP con streaming de audio. También, se permite conectar teclados completos por USB o Bluetooth.

El uso de los dispositivos en un entorno empresarial es mejorado. Entre las novedades introducidas destacamos las nuevas políticas administrativas con encriptación del almacenamiento, caducidad de contraseña y mejoras para administrar los dispositivos de empresa de forma eficaz.

A pesar de la nueva interfaz gráfica optimizada para tabletas, Android 3.0 es compatible con las aplicaciones creadas para versiones anteriores. La tecla de menú, inexistente en las nuevas tabletas, es reemplazada por un menú que aparece en la barra de acción.

Android 3.1 Nivel de API 12 (mayo 2011)

Se permite manejar dispositivos conectados por USB (tanto host como dispositivo). Protocolo de transferencia de fotos y vídeo (PTP/MTP) y de tiempo real (RTP).

Android 3.2 Nivel de API 13 (julio 2011)

Optimizaciones para distintos tipos de tableta. Zoom compatible para aplicaciones de tamaño fijo. Sincronización multimedia desde SD.

Ice Cream Sandwich

Android 4.0 Nivel de API 14 (octubre 2011)

La característica más importante es que se unifican las dos versiones anteriores (2.x para teléfonos y 3.x para tabletas) en una sola compatible con cualquier tipo de dispositivo. Entre las características más interesantes destacamos:

Se introduce un nuevo interfaz de usuario totalmente renovado. Por ejmplo, se reemplazan los botones físicos por botones en pantalla (como ocurria en las versiones 3.x).

Nuevo API de reconocedor facial, permite entre otras muchas aplicaciones desbloquear el teléfono a su propietario. También se mejora en el reconocimiento de voz. Por ejemplo se puede empezar a hablar en cuanto pulsamos el botón.

Aparece un nuevo gestor de tráfico de datos por Internet, donde podremos ver el consumo de forma gráfica y donde podemos definir los límites a ese consumo para evitar cargos inesperados con la operadora. Incorpora herramientas para la edición de imágenes en tiempo real, con herramientas para distorsionar, manipular e interactuar con la imagen al momento de ser capturada. Se mejora el API para comunicaciones por NFC y la integración con redes sociales.

En diciembre del 2011 aparece una actualización de mantenimiento (versión 4.0.2) que no aumenta el nivel de API.

Android 4.0.3 Nivel de API 15 (diciembre 2011)

Se introducen ligeras mejoras en algunas APIs incluyendo el de redes sociales, calendario, revisor ortográfico, texto a voz y bases de datos entre otros. En marzo de 2012 aparece la actualización 4.0.4.

Jelly Bean

Android 4.1 Nivel de API 16 (julio 2012)

En esta versión se hace hincapié en mejorar un punto débil de Android: la fluidez del interfaz de usuario. Con este propósito se incorporan varias técnicas, como: sincronismo vertical, triple búfer y aumentar la velocidad del procesador al tocar la pantalla.

Se mejoran las notificaciones con un sistema de información expandible personalizada. Los Widgets de escritorio pueden ajustar su tamaño y hacerse sitio de forma automática al situarlos en el escritorio. El dictado por voz puede realizarse sin conexión a Internet (de momento en ingles).

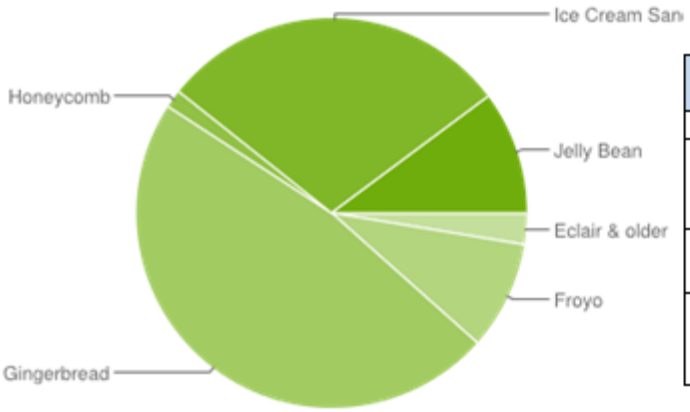
Se introducen varias mejoras en Google Search. Se potencia la búsqueda por voz con resultados en forma de ficha. La función Google Now permite utilizar información de posición, agenda y hora en las búsquedas.

Se incorporan nuevo soporte para usuarios internacionales: como texto bidireccional y teclados instalables. Para mejorar la seguridad las aplicaciones son cifradas. También se permite actualizaciones parciales de aplicaciones.

Android 4.2 Nivel de API 17 (noviembre 2012)

Una de las novedades más importantes es que podemos crear varias cuentas de usuario en el mismo dispositivo. Aunque, esta característica solo está disponible en tabletas. Cada cuenta tendrá sus propias aplicaciones y configuración.

Los Widgets de escritorio pueden aparecer en la pantalla de bloqueo. Se incorpora un nuevo teclado predictivo deslizante al estilo Swype. Posibilidad de conectar dispositivo y TVHD mediante wifi (Miracast). Mejoras menores en las notificaciones. Nueva aplicación de cámara que incorpora la funcionalidad Photo Sphere para hacer fotos panorámicas inmersivas (en 360º).



Plataforma	Nivel API	Porcent.
1.6 Donut	4	0.2%
2.1 Éclair	7	2.7%
2.2 Froyo	8	9.0%
2.3 Gingerbread	10	47.4%
3.1 Honeycomb	12	1.0%
3.2 Honeycomb	13	2.2%
4.0 Ice Cream S.	15	29.1%
4.1 Jelly Bean	16	9.0%
4.2 Jelly Bean	17	1.2%

Imagen 1: Dispositivos Android según plataforma instalada, que han accedido a Google Play Store durante dos semanas terminado el 3 de enero de 2013.

1.3. Creación del entorno de desarrollo para Android

Para el desarrollo de las aplicaciones vamos a poder utilizar un potente y moderno entorno de desarrollo. Al igual que Android, todas las herramientas están basadas en *software* libre. Aunque existen varias alternativas para desarrollar aplicaciones en Android. En este texto se supondrá que estamos trabajando con el *software* enumerado a continuación:

•Java Runtime Environment 5.0 o superior.

•Eclipse (Eclipse IDE for Java Developers)

•Android SDK (Google).

•Eclipse Plug-in (Android Development Toolkit- ADT).

Describiremos a continuación el proceso a seguir para instalar el *software* anterior. Si ya tienes instalado Eclipse en tu ordenador puedes completar la instalación añadiendo *Android SDK* y *Eclipse Plug-in*. Mantendrás tu configuración actual y simplemente añadirás nuevas funcionalidades. El este caso, sáltate los siguientes dos apartados.

Instalación de la máquina virtual Java

Este *software* va a permitir ejecutar código Java en tu equipo. A la máquina virtual Java también se la conoce como entorno de ejecución Java, Java Runtime Environment (JRE) o Java Virtual Machine (JVM).

Muy posiblemente ya tengas instalada la Máquina Virtual Java en tu equipo. Si es así puedes pasar directamente al punto siguiente. En caso de dudas, puedes pasar también al punto siguiente. Al concluirlo te indicará si la versión de la máquina virtual Java es incorrecta. En caso necesario, regresa a este punto para instalar una adecuada.

Para instalar la Máquina Virtual Java accede a http://java.com/es/download/ y descarga e instala el fichero correspondiente a tu sistema operativo.

Instalación de Eclipse

Eclipse resulta el entorno de desarrollo más recomendable para Android, es libre y además es soportado por Google (ha sido utilizado por los desarrolladores de Google para crear Android). Puedes utilizar cualquier versión de Eclipse a partir de la 3.3.1.

Para instalar Eclipse hay que seguir los siguientes pasos:

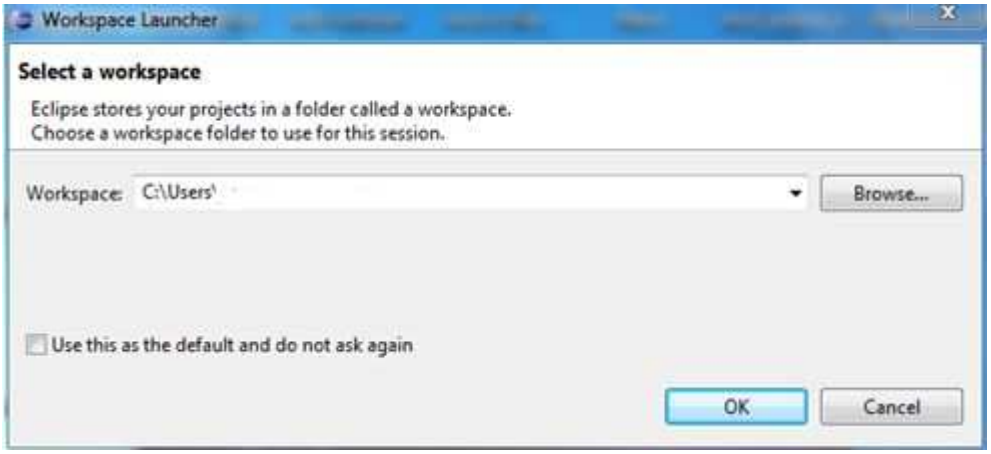
1. Accede a la página http://www.eclipse.org/downloads/ y descarga la última versión de “Eclipse IDE for Java EE Developers”. Verás que se encuentra disponible para los sistemas operativos más utilizados, como Windows, Linux y Mac OS.

2. Este *software* no requiere una instalación específica, simplemente descomprimir los ficheros en la carpeta que prefieras. Si así lo deseas puedes crear un acceso directo en el escritorio o en el menú inicio del fichero *eclipse.exe*.

***Nota**: Si al ejecutar Eclipse te aparece el siguiente mensaje:*

Nos indica que no tenemos instalada la máquina virtual Java (o la versión no es la adecuada). Para solucionarlo regresa al punto anterior.

3. Al arrancar Eclipse comenzará preguntándonos que carpeta queremos utilizar como *workspace*. En esta carpeta serán almacenados los proyectos que crees en Eclipse. Es importante que conozcas su ubicación para poder hacer copias de seguridad de tus proyectos.



4. Aparecerá una ventana de bienvenida.

Instalar Android SDK de Google

El siguiente paso va a consistir en instalar Android SDK de Google.

Pasos:

1. Accede a la siguiente página http://developer.android.com/sdk y descarga el fichero correspondiente a tu sistema operativo.

2. Este *software* no requiere una instalación específica, simplemente descomprimir los ficheros en la carpeta que prefieras.

Nota: *En algunos sistemas tendremos problemas cuando la ruta donde se descomprime los ficheros contiene un espacio en blanco.*

3. Ejecuta el programa SDK Manager.

4. Seleccionar los paquetes a instalar. Aparecerá una ventana donde podremos seleccionar los paquetes a instalar. Si lo deseas puedes instalar todos los paquetes (*Accept All*), en este caso el proceso de instalación puede tardar más de una hora. Si no dispones de tanto tiempo puedes seleccionar solo algunos paquetes. Siempre resulta interesante instalar la última versión de Android (incluyendo documentación, ejemplos y por supuesto la plataforma). Más adelante podrás instalar más paquetes si necesitas otras plataformas de desarrollo u otras máquinas virtuales.

Instalación del plug-in Android para Eclipse (ADT)

El último paso consiste en instalar el plug-in Android para Eclipse, también conocido como ADT. Este software desarrollado por Google, instala una serie de complementos en Eclipse, de forma que el entorno de desarrollo se adapte al desarrollo de aplicaciones para Android. Se crearán nuevos botones, tipos de aplicación, vistas,... para integrar Eclipse con el Android SDK que acabamos de instalar.

Pasos:

Para instalar el plug-in Android sigue los siguientes pasos:

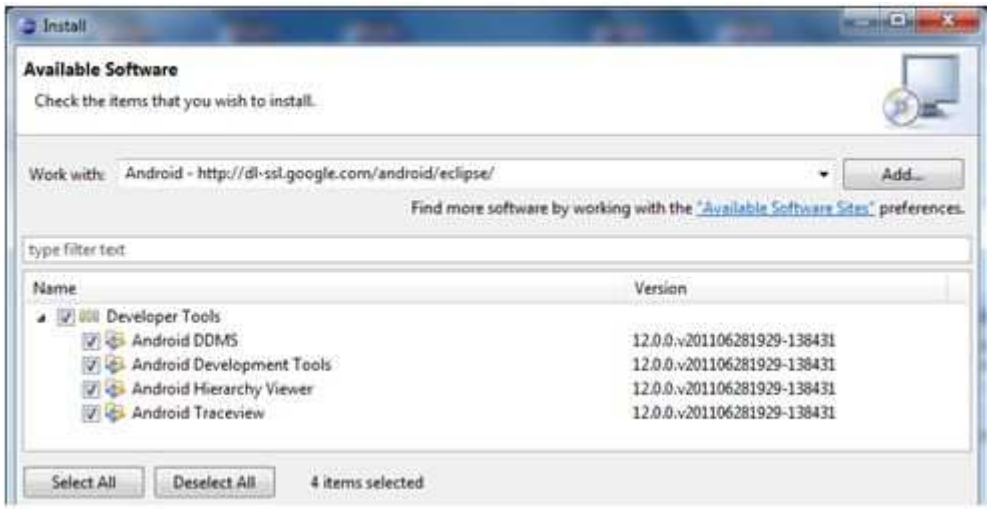
1. Arranca Eclipse y selecciona *Help>Install New Software*...

2. En el diálogo *Available Software* que aparece, haz clic en *Add…* En el cuadro de diálogo *Add Site* que sale introduce un nombre para el sitio remoto (por ejemplo, *Plug-in Android*) en el campo *Name*. En el campo *Location* introduce la siguiente URL:

http://dl-ssl.google.com/android/eclipse/

NOTA:*Si tienes algún problema en adquirir el plug-in, utiliza https en el URL en vez de http. Pulsa OK.*

Ahora, en el cuadro *Available Software,* debe aparecer *Developer Tools:*

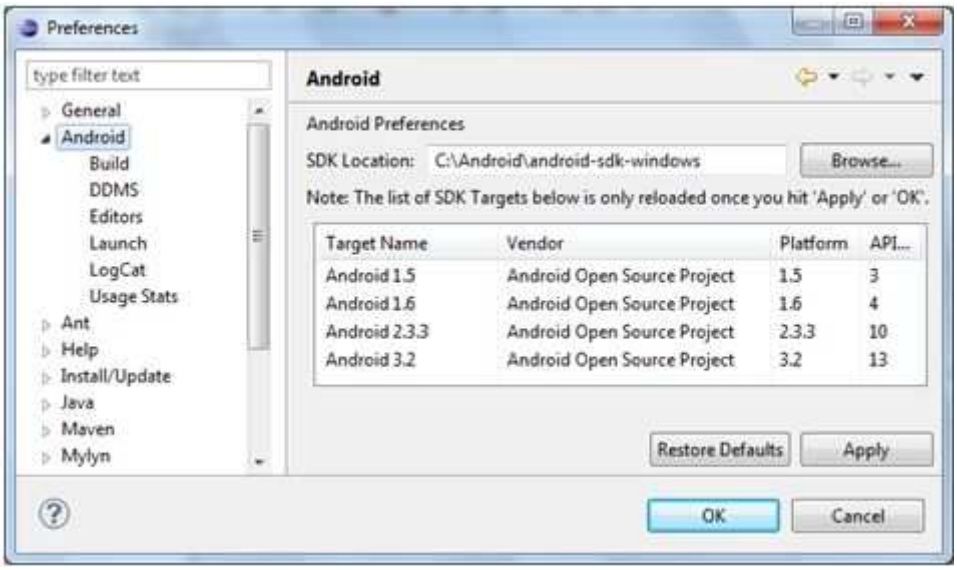


3. Selecciona los paquetes a instalar y pulsa Next. Aparecen listadas las características de Android DDMS y Android Development Tools.

4. Pulsa Next para leer y aceptar la licencia e instalar cualquier dependencia y pulsa Finish.

5. Reinicia Eclipse.

6. Configura Eclipse para que sepa donde se ha instalado Android SDK. Para ello entra en las preferencias en*Windows>Preferences…* y selecciona Android del panel de la izquierda. Ahora pulsa *Browse…* para seleccionar el *SDK Location* elige la ruta donde hayas descomprimido Android SDK. Aplica los cambios y pulsa *OK*.



2. Realidad Aumentada

2.1. Introducción

La **realidad aumentada** (RA) es el término que se usa para definir una visión directa o indirecta de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real. Consiste en un conjunto de dispositivos que añaden información virtual a la información física ya existente. Esta es la principal diferencia con la realidad virtual, puesto que no sustituye la realidad física, sino que sobreimprime los datos informáticos al mundo real.

Con la ayuda de la tecnología (por ejemplo, añadiendo la visión por computador y reconocimiento de objetos) la información sobre el mundo real alrededor del usuario se convierte en interactiva y digital. La información artificial sobre el medio ambiente y los objetos pueden ser almacenada y recuperada como una capa de información en la parte superior de la visión del mundo real.

La realidad aumentada de investigación explora la aplicación de imágenes generadas por ordenador en tiempo real a secuencias de vídeo como una forma de ampliar el mundo real. La investigación incluye el uso de pantallas colocadas en la cabeza, un display virtual colocado en la retina para mejorar la visualización, y la construcción de ambientes controlados a partir sensores y actuadores.

Imagen 2: Logotipo de la Realidad Aumentada

Hay dos definiciones aceptadas para la Realidad Aumentada. Una de ellas fue dada por Ronald Azuma en 1997. La definición de Azuma dice que la realidad aumentada:

•Combina elementos reales y virtuales.

•Es interactiva en tiempo real.

•Está registrada en 3D.

Además Paul Milgram y Fumio Kishino definen en 1994 la realidad de Milgram-Virtuality Continuum como un continuo que abarca desde el entorno real a un entorno virtual puro. Entre medio hay Realidad Aumentada (más cerca del entorno real) y Virtualidad Aumentada (más cerca del entorno virtual).

Imagen 3: Diagrama de niveles de realidad

Realidad Aumentada también es la incorporación de datos e información digital en un entorno real, por medio del reconocimiento de patrones que se realiza mediante un software, en otras palabras, es una herramienta interactiva que está dando sus primeros pasos alrededor del mundo y que en unos años, la veremos en todas partes, trayendo un mundo digital inimaginable a nuestro entorno real. Su gran diferencia con la realidad virtual, es que ésta nos extrae de nuestro entorno para llevarnos a una realidad.

Imagen 4: Ejemplo de Realidad Aumentada

2.2. Componentes de la realidad aumentada

•Monitor del computador: instrumento donde se verá reflejado la suma de lo real y lo virtual que conforman la realidad aumentada.

•Cámara Web: dispositivo que toma la información del mundo real y la transmite al software de realidad aumentada.

•Software: programa que toma los datos reales y los transforma en realidad aumentada.

•Marcadores: los marcadores básicamente son hojas de papel con símbolos que el software interpreta y de acuerdo a un marcador especifico realiza una respuesta especifica (mostrar una imagen 3D, hacerle cambios de movimiento al objeto 3D que ya este creado con un marcador)

Los dispositivos de Realidad aumentada normalmente constan de un **"headset"** y un sistema de display para mostrar al usuario la información virtual que se añade a la real. El "headset" lleva incorporado sistemas de GPS, necesarios para poder localizar con precisión la situación del usuario.

Los dos principales sistemas de "displays" empleados son la pantalla óptica transparente (Optical See-through Display) y la pantalla de mezcla de imágenes (Video-mixed Display). Tanto uno como el otro usan imágenes virtuales que se muestran al usuario mezcladas con la realidad o bien proyectadas directamente en la pantalla.

Los Sistemas de realidad aumentada modernos utilizan una o más de las siguientes tecnologías: cámaras digitales, sensores ópticos, acelerómetros, GPS, giroscopios,

brújulas de estado sólido, etc. El hardware de procesamiento de sonido podría ser incluido en los sistemas de realidad aumentada. Los sistemas de cámaras basadas en realidad aumentada requieren de una unidad CPU potente y gran cantidad de memoria RAM para procesar imágenes de dichas cámaras. La combinación de todos estos elementos se dan a menudo en los smartphones modernos, que los convierten en una posible plataforma de realidad aumentada.

3. Sensores soportados

3.1. Introducción

Cada vez es más frecuente encontrar dispositivos móviles con distintos tipos de sensores. La razón fundamental es que nos proporcionan medidas de nuestro entorno, pudiendo adaptar las aplicaciones del terminal al mismo. Esto significa que el usuario puede adentrarse en una nueva experiencia, descubriendo que a su alrededor puede haber mucha más información, diversión o mucho más entretenimiento de lo que ve a simple vista. Android destaca por su soporte para múltiples sensores. En este apartado nos centraremos en el GPS, y para la realización del módulo de realidad aumentada, el acelerómetro y el sensor de orientación, los dos pertenecientes a la clase SensorManager, además de la cámara.

GPS

En la actualidad es bastante común encontrar un dispositivo GPS integrado en los terminales móviles de nueva generación. Este sensor nos proporciona nuestra información GPS, como las coordenadas de nuestra posición, su precisión, la altitud sobre el nivel del mar, los satélites que estamos escuchando... Es, por tanto el principal elemento para proporcionar servicios basados en datos de localización, como navegadores, redes sociales móviles o buscadores de recursos según su posición.

3.2. Sensores de la clase SensorManager

La clase SensorManager de Android SDK es la que engloba la abstracción de todos los sensores soportados por el sistema operativo, salvo el GPS y la cámara. Entre los sensores que se encuentran disponibles, podemos destacar los siguientes:

3.2.1. Acelerómetro

Los acelerómetros se presentan como unos sensores capaces de medir la aceleración instantánea de la gravedad terrestre. Las medidas son realizadas a lo largo de los tres ejes cartesianos, como se muestra en la imagen, por lo que tenemos tres componentes, que corresponden con la fuerza aplicada por el dispositivo sobre los ejes (x, y, z).

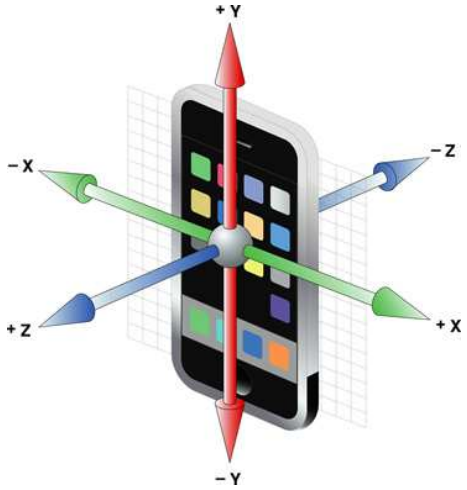


Imagen 5: Ejes cartesianos del acelerómetro

Como sabemos que el módulo de la aceleración de la gravedad es, aproximadamente, 9,8m/s^2, el módulo percibido por el terminal en reposo debe ser el mismo. Por tanto, valores mayores o menores indican que el portador del dispositivo comienza a caminar o se detiene. Como sabemos la dirección y el sentido del mayor cambio, podemos estimar la dirección hacia la que se mueve.

Por otro lado, como sabemos la magnitud de la gravedad en esas tres componentes, podemos calcular la inclinación del dispositivo en reposo. Por ejemplo, situando el terminal en posición horizontal, con la parte positiva del eje Z apuntando hacia el techo, tenemos que sólo ese eje se ve afectado por la gravedad. Si comenzamos a inclinarlo alrededor de cualquiera de los otros ejes, veremos que la magnitud del eje Z decae, incrementándose en los otros ejes. Haciendo una

medida de proporcionalidad entre estos valores, podemos saber el grado de inclinación del dispositivo.

3.2.2. Orientación

El sensor de orientación no es realmente un sensor, sino una abstracción de los valores capturados por los acelerómetros y el sensor de campo magnético, con el objetivo de construir una brújula digital, que puede determinar la dirección hacia la que está orientado el usuario. Sin embargo, no solo devuelve la orientación, también la inclinación del dispositivo.

Entre las magnitudes retornadas por este sensor, tenemos las siguientes:

●*Acimut*: mide en grados la rotación alrededor del eje Z. Este ángulo puede ser traducido en puntos cardinales, teniendo en cuenta las siguientes referencias:

•El Norte se corresponde con 0º/360º

•El Este se corresponde con 90º

•El Sur se corresponde con 180º

•El Este se corresponde con 270º

●*Pitch*: mide en grados la rotación alrededor del eje X. Su rango de medida es [-180º, 180º].

●*Roll*: mide en grados la rotación alrededor del eje Y. Su rango de medida es [-90º, 90º].

3.3. Cámara

La cámara es un sensor que nos proporciona imágenes de nuestro entorno. Como en el caso del GPS, podemos encontrar fácilmente cámaras en los dispositivos móviles desde hace varios años, más aún en los de nueva generación.

Este tipo de sensores son útiles tanto para tomar fotografías como para grabar vídeos, que podemos compartir con nuestros amigos o almacenarlos para realizar procesamientos posteriores. A partir del análisis digital de una imagen, podemos extraer una serie de parámetros que nos den información relevante sobre nuestro

entorno, lo que puede ser útil para una aplicación determinada (detección de objetos, cálculo de magnitudes visuales, búsqueda de patrones, etc.).

Por otro lado, también resultan útiles para la representación de ciertos objetos de manera mucho más intuitiva para el usuario. Por ejemplo, recursos como tiendas, restaurantes o cualquier otro objeto cuyas coordenadas geográficas sean conocidas, pueden ser presentados en la pantalla del dispositivo como un dibujo o imagen sobre una vista previa de la cámara, mostrando claramente en qué posicion se encuentran en relación a la localización del usuario.

3.4. GPS

El Sistema de Posicionamiento Global (NAVSTAR-GPS o más comúnmente GPS) es el más común y conocido a la hora de enfrentarse al problema de localizar un dispositivo móvil. Se trata de una constelación de 24 satélites que giran alrededor de la Tierra a una distancia de 20200 Km, con trayectorias sincronizadas para cubrir toda la superficie del planeta.

Para su funcionamiento, el terminal del usuario debe disponer de una antena y un módulo GPS. Con estos elementos, el dispositivo es capaz de recibir los datos que envían los satélites que tenga en línea de visión, que son fundamentalmente la posición del satélite y una marca de tiempo. Sincronizando esas marcas de tiempo se puede ver el retardo que han sufrido las señales hasta llegar al terminal, y por tanto la distancia a la que se encuentran los satélites. Ahora, triangulando según esas distancias se puede hallar la posición del dispositivo. El número de satélites necesarios para conseguir el punto en el que nos encontramos es variable, según la calidad del módulo GPS del terminal: teóricamente son necesarios 3, pero en la práctica pueden ser necesarios hasta 9 satélites para obtener una buena aproximación.

En cuanto a los datos que proporciona el sistema son varios, entre los que se encuentran las coordenadas geográficas (latitud y longitud) de nuestra posición, la altura sobre el nivel del mar a la que nos hallamos, marcas de tiempo, etc. La precisión de las coordenadas geográficas en condiciones óptimas tiene un error de aproximadamente 15 metros.

El sistema GPS es, por tanto, ideal para establecer la posición de un dispositivo, ya que prácticamente en cualquier espacio abierto tenemos cobertura. Sin embargo, el sistema GPS en situaciones en las que permanecemos rodeados de edificios u obstáculos altos en general, no disponemos de línea de visión directa con el número suficiente de satélites necesarios para fijar nuestra posición. Por tanto, el sistema no es suficiente para ello.

Por último, otro problema al que nos enfrentamos con esta tecnología es el del consumo de batería. Los dispositivos móviles se caracterizan por alimentarse de una batería. Los requisitos de ligereza hacen que las mismas deban ser del mínimo tamaño posible con un rendimiento adecuado. El contratiempo se produce porque el modulo GPS tiene un consumo de batería elevado, limitando la duración de operatividad del terminal, que, en algunos casos como en los teléfonos, es un factor crítico. Para paliar esto se han desarrollado algoritmos que ahorran batería, conectando el GPS sólo cuando sea estrictamente necesario.

3.5. Acelerómetro

Los acelerómetros se emplean para medir vibraciones y oscilaciones, así como para el desarrollo de productos (por ejemplo, componentes o herramientas). La medición proporciona los siguientes parámetros: aceleración de la vibración, velocidad de vibración y variación de vibración. De este modo se caracterizan las vibraciones con precisión.

Los acelerómetros se pueden utilizar para detectar la inclinación, la vibración, y el choque. Están cada vez más presentes en dispositivos electrónicos portables.

Otra característica que se puede extraer es la velocidad del portador del dispositivo, ya que podemos apreciar y medir la magnitud, la dirección y el sentido de los cambios en la aceleración.

¿Para qué se usan los acelerómetros?

Hasta ahora, para interactuar con cualquier máquina utilizábamos un teclado, un ratón, una pantalla táctil o, en algunos casos, una cámara. Todos ellos, utilizan diferentes tipos de sensores para convertir fenómenos físicos en señales eléctricas.

Los acelerómetros son un paso más, un tipo de sensor que permite medir la reacción de un objeto sometido a una fuerza, evaluando la dirección y la variación

de la velocidad, con el que se puede medir el eje y el grado de inclinación del aparato. Es decir, es un dispositivo capaz de convertir ciertos gestos en una señal eléctrica que puede ser o no interpretada por el sistema. Viendo la información que podemos obtener, surgen dos utilidades básicas:

Por un lado, podemos abordar el problema de la detección de movimiento, que puede servir para la monitorización de gente enferma o para localización en interiores.

Permiten realizar multitud de tareas, como por ejemplo girar una foto con solo dar vuelta el aparato, algo que era increíble cuando apareió por primera vez. Además de cosas que simplifican el uso del aparato, también se utiliza con fines más útiles, por ejemplo elegir de qué forma queremos navegar por una web, seleccionando la opción vertical o apaisada en función del diseño de la página web. También esta característica se ha comenzado a utilizar en el desarrollo de los juegos permitiéndole al jugador una experiencia de juego más real, como es el caso de la Nintendo Wii. También se pueden hacer cosas más originales como modificar el perfil activo o gobernar un vehículo mediante radio control, simplemente moviendo el teléfono.

4. Manual de usuario para desarrolladores

En este punto se introducirán los conocimientos básicos de Android para poder manejar una aplicación en este sistema operativo.

Teniendo el manual de instalación del entorno de trabajo del apartado 1,3, donde se explica con claridad como instalar el SDK y los demás plugins para poder trabajar con Android, vamos a hablar de los componentes básicos que tiene todo programa en Android.

4.1 Componentes básicos de una aplicación Android

Cada vez que creamos un nuevo proyecto Android en Eclipse, se genera automáticamente un sistema de archivos definido para todas las aplicaciones de dicho sistema operativo.

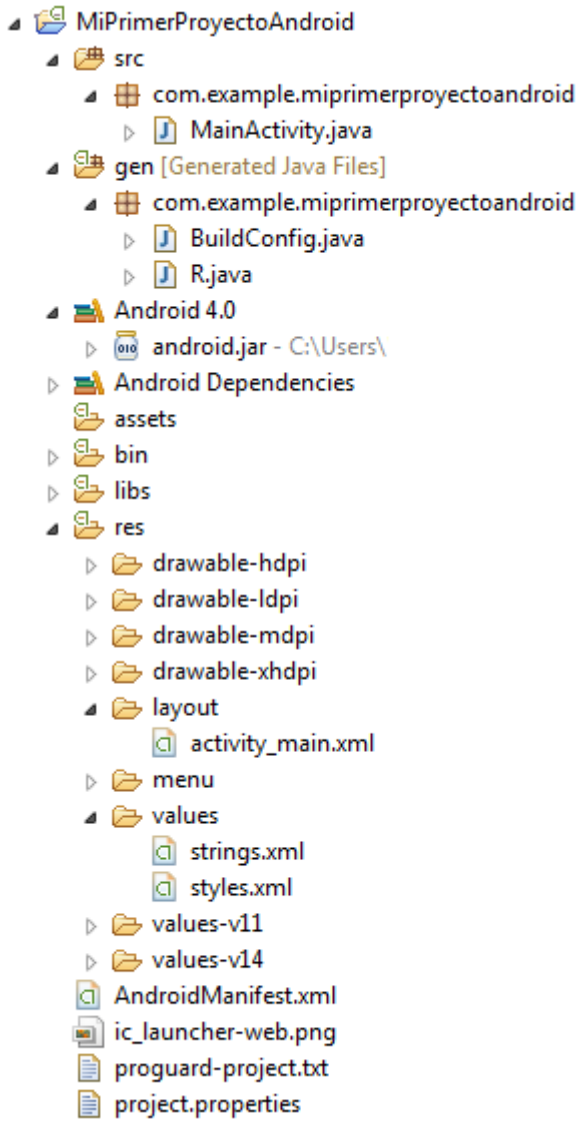


Imagen 6: Archivos generados en la creación de un proyecto Android

Según la versión del SDK, se han ido haciendo pequeñas modificaciones en la estructura de las aplicaciones en Android. La jerarquía de carpetas y archivos que se muestra en la imagen definen todos los proyectos en Android construidos a partir de la versión más reciente del SDK 4.0. Dentro de la carpeta principal, llamada *MiPrimerProyectoAndroid*, se encuentra la estructura de archivos que se definen a continuación:

▪En la carpeta ***src*** se guarda el paquete que contiene las clases necesarias para hacer funcionar la aplicación. Son los archivos de origen *.java* que una vez compilados se transformarán en archivos *.class*.

▪En la carpeta ***bin***, se guardan las clases compiladas que se han descrito en el punto anterior, los fichero *.class*.

▪En el directorio ***gen*** se ubica una clase muy peculiar de las aplicaciones en Android, la clase *R.java*. Esta clase se genera automáticamente cada vez que modificamos los ficheros de recursos y no podemos retocarla manualmente. Se podría considerar como un nexo de unión entre los recursos de la aplicación y los ficheros *.xml* que se encuentran en el proyecto.

▪Dentro del proyecto también se encuentran las **librerías** de Android que son las que dan funcionalidad al sistema. Las librerías también dependen de la versión del SDK que estemos utilizando.

▪En la carpeta ***assets***, inicialmente vacía y que en general no suele usarse, se pueden ubicar cualquier tipo de ficheros que represente un origen de datos para la aplicación. Para poder manejar esos datos, se tendría que crear un gestor de assets, o *AssetManager*, que se encargaría de asociar los datos dentro de la propia aplicación.

▪En el directorio ***res*** se genera automáticamente una estructura definida para gestionar los recursos de origen de la aplicación. En el subdirectorio ***drawable*** se almacenan todos los ficheros de imágenes necesarias para la aplicación. Dentro de ***layout*** se guardan los ficheros *.xml*, que componen la interfaz gráfica del usuario. En el subdirectorio ***values*** se almacena un fichero *string.xml* donde se asocian todas las cadenas de texto de la aplicación, y también se pueden tener valores de atributos o estilos de los elementos de la aplicación.

▪Finalmente, tenemos el fichero ***AndroidManifest.xml*** que es indispensable para todas las aplicaciones de Android. En este fichero se almacenan los parámetros de configuración de la aplicación, tal como las Activities principales (posteriormente serán descritas), los privilegios asignados, etc.

▪El fichero *project.properties* no influirá en la realización del proyecto, ya que sólo es un fichero que se genera automáticamente indicando las propiedades por defecto del SDK con el que estamos trabajando. En el punto anterior, se ha mencionado el término *Activity*, que se configura en el fichero AndroidManifest.xml. Una ***Activity*** es una de las componentes más importantes de una aplicación Android. Toda aplicación puede constar de varias componentes que interactúan entre sí, y cada una de ellas tiene una función determinada a realizar. Las componentes se implementan en las clases que tengamos creadas en la carpeta ***src*** de nuestro

proyecto. Las más importantes de toda aplicación las pasamos a definir a continuación:

●***Activity***: En general, corresponde a una pantalla específica de la aplicación, como por ejemplo la pantalla inicial que representa el punto de entrada de dicha aplicación. Desde ahí se pueden invocar a las principales tareas que el usuario puede llevar a cabo.

Las actividades pasan por una serie de estados, desde que se crean hasta que se destruyen, que conviene conocer para poderlos capturar y actuar en ellos en determinadas circunstancias.

Una actividad tiene esencialmente 4 estados:

►Si una actividad está en el primer plano de la pantalla (en la parte superior de la pila), está activa o en marcha (ejecutándose).

► Si una actividad ha perdido el foco, pero sigue siendo visible (es decir, una nueva actividad que no ocupe toda la pantalla o una transparente que tiene el foco en la parte superior de su actividad), está en pausa. Una actividad pausada está completamente viva (mantiene toda la información del estado y del miembro y permanece unido al gestor de ventanas), pero puede morir por el sistema en situaciones extremas de poca memoria.

►Si una actividad es completamente oscurecida por otra actividad, se detiene. Conserva toda la información del estado y miembros; sin embargo, ya no es visible para el usuario por lo que su ventana se oculta y a menudo esta actividad será eliminada por el sistema cuando la memoria sea necesaria.

►Si una actividad está en pausa o se detiene, el sistema puede dejar la actividad sin memoria, ya sea esperando a que termine, o simplemente matar el proceso. Cuando se vuelva a mostrar al usuario, ésta debe ser completamente renovada y restaurada a su estado anterior.

Por tanto, para terminar de entender claramente las actividades, destacaremos las tres fases de una actividad:

1. La vida entera de una actividad ocurre entre la primera llamada a **onCreate()** y una única llamada final a **onDestroy()**. Una actividad será

lanzada por primera vez y, además, establecerá toda la configuración general de su estado en onCreate(), y liberará todos los recursos que quedan en OnDestroy ().

2. El curso de la vida visible de una actividad ocurre entre una llamada al método **onStart()** y la correspondiente llamada a **onStop()**. Durante este tiempo el usuario puede ver la actividad en pantalla, aunque podría no ser en primer plano e interactuando con ella. Entre estos dos métodos se pueden mantener los recursos que se necesitan para mostrar el resultado de la actividad al usuario. Los métodos onStart() y onStop() se pueden llamar varias veces, de esta forma la actividad se hace visible y se oculta para el usuario.

3. El tiempo de vida en primer plano de una actividad ocurre entre una llamada a **onResume()** y a **onPause()**. Durante este tiempo la actividad se encuentra delante de todas las demás actividades y en interacción con el usuario. Una actividad con frecuencia puede estar en esta situación. El método onPause() es el estado en el que entra la actividad cuando se lanza otra. La actividad permanece en segundo plano hasta que se vuelva de la nueva que se acaba de lanzar. El método onResume() es el estado que indica que la actividad ha vuelto a entrar en primer plano tras estar en estado de pausa.

Hay que destacar que las actividades no terminan, a no ser que se haga de forma explícita. Normalmente permanecen en segundo plano (no en pausa, sino en stop) hasta que el sistema operativo las cierra para liberar la memoria que ocupan en caso de necesidad.

A continuación, un diagrama en el que se muestra de manera visual lo que se acaba de explicar:

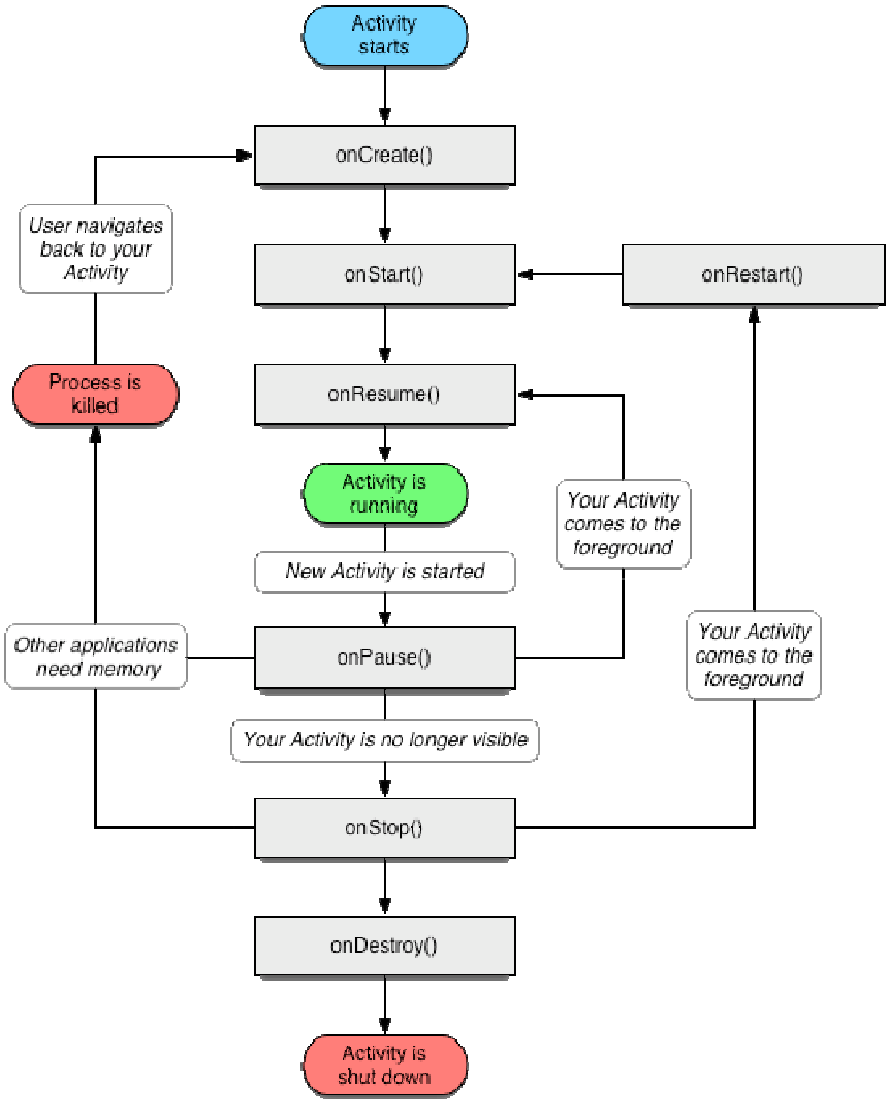


Imagen 7: Ciclo de vida de una actividad

• ***Service***: Representa una aplicación que corre internamente en el sistema y debe ejecutarse sin interacción con el usuario. Es una aplicación que se está ejecutando pero sin necesidad de ser vista, como puede ser el caso de reproducir una pista de audio. Es una aplicación que puede mandar los mensajes que sean necesarios a un determinado servicio que esté activo. Para utilizar un recurso, como por ejemplo el GPS, es necesario solicitarlo al manejador de servicios, que nos proporcione una instancia del mismo y así poder utilizarlo.

•***IntentReceiver***: Este componente permite a la aplicación realizar llamadas internas que responden a cambios de estado de nuestro terminal. Por ejemplo una llamada, un mensaje recibido, etc.

•***ContentProvider***: Representa una capa que permite compartir datos a las distintas aplicaciones de Android. Es necesario declarar este tipo de componentes para poner a disposición de los procesos los datos que se consideren oportunos.

5. Implementación de la aplicación

En este aparatado se explican los aspectos más importantes para la implementación de la aplicación.

Para dicha implementación se pueden distinguir 3 áreas de trabajo principales que se han llevado a cabo:

●**Visualización de la imagen de la cámara.**

Una de las partes más importantes en una aplicación de realidad aumentada, es acceder a la cámara del dispositivo para posteriormente poder mezclar esta imagen con los datos virtuales.

●**Sensores.**

Es otra parte importante de la realidad aumentada ya que serán estos datos los que se supondrán a la imagen de la cámara para poder verlos en tiempo real.

●**Algoritmo para calcular la distancia.**

En este apartado será dónde se implementará el algoritmo que calcule la distancia al delfín. Para poder llevarlo a cabo es necesario implementar un *Listener* que es por dónde el usuario inserta la altura a la que se encuentra para que se pueda llevar a cabo el cálculo.

5.1. Visualización de la imagen de la cámara

La imagen que proporciona la cámara del dispositivo es usada como fondo dónde se superpondrán los datos de los sensores así como la altura y la distancia final calculada. Para la visualización de esta imagen se utiliza la variable *cameraPreview* que contendrá una referencia a la *SurfaceView* dónde se representará la imagen mostrada desde la cámara.

SurfaceView es un tipo de vista que contiene una superficie sobre la que dibujar, por lo que la utilizaremos para mostrar la imagen de la cámara. La clase *SurfaceHolder* da acceso a la superficie donde pintar y la interfaz *SurfaceHolder.Callback*, contiene tres métodos que notifican cuando la superficie se crea, se destruye o cambia de tamaño.

En primer lugar creamos un objeto de la clase *SurfaceHolder*, y otro la de la clase *Camera*, la cual accede a la cámara del dispositivo. Como se muestra en el siguiente código, le indicamos al *SurfaceHolder* que notifique los cambios que se produzcan en la superficie a nuestro objeto, e indicamos el tipo de superficie a usar. Para la imagen de la cámara debemos utilizar el tipo SURFACE_TYPE_PUSH_BUFFERS.

```
SurfaceView cameraPreview;
SurfaceHolder previewHolder;
Camera camera;
```

```
cameraPreview = (SurfaceView)findViewById(R.id.cameraPreview);
previewHolder = cameraPreview.getHolder();
previewHolder.addCallback(surfaceCallback);

previewHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
```

Se implementan los 3 métodos correspondientes a la interfaz:

• *public void surfaceCreated(SurfaceHolder holder):* es llamado cuando la superficie se crea. Le indicamos que inicialice el sistema de la cámara y que muestre la imagen utilizando la superficie asociada al holder.

• *public void surfaceDestroyed(SurfaceHolder holder):* es llamado cuando la superficie se destruye, es decir, cuando se cambia de actividad o se inicia otra nueva. Paramos la visualización de la cámara y liberamos la cámara para que pueda volver a ser utilizada si así se requiere. Si no se liberase y cambiase el foco principal de la aplicación a otra actividad, al volver a visualizar la imagen captada por la cámara provocaría excepciones de memoria y esta no podría volver a ser usada.

• *public void surfaceChanged(SurfaceHolder holder, int format, int width, int height):* se llama cuando la superficie cambia de tamaño, por ejemplo, cuando gira el dispositivo de posición horizontal a vertical y viceversa.

Para poder usar la cámara en la aplicación es necesario introducir los permisos oportunos en el Manifiesto:

```
<uses-feature android:name="android.hardware.camera" />
<uses-permission android:name="android.permission.CAMERA" />
```

5.2. Sensores

Los sensores se utilizan para poder saber hacia dónde está apuntando la cámara del dispositivo y las rotaciones que se producen. Para poder manejar los sensores de un dispositivo Android es necesario utilizar las clases *Sensor* y *sensorManager*. En esta aplicación se usarán dos sensores de la clase *sensorManager,* que serán:

● *Sensor.TYPE_ACCELEROMETER*: mide la aceleración de los 3 ejes en m/s2.

● *Sensor.TYPE_ORIENTATION*: devuelve la orientación de los 3 ejes en grados.

La tasa de actualización de los datos recogidos por los sensores, puede ser escogida entre cuatro valores predeterminados:

• *SensorManager.SENSOR_DELAY_FASTEST:* indica la mayor tasa posible de actualización.

• *SensorManager.SENSOR_DELAY_GAME*: indica una actualización adecuada para el control de videojuegos.

• *SensorManager.SENSOR_DELAY_NORMAL*: indica la tasa de actualización por defecto.

• *SensorManager.SENSOR_DELAY_UI*: indica una tasa adecuada para una interfaz de usuario.

Sensor orientación y acelerómetro

```
        //SENSOR ORIENTACION
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        orientationSensor = Sensor.TYPE_ORIENTATION;
        sensorManager.registerListener(sensorEventListener,
sensorManager.getDefaultSensor(orientationSensor),
SensorManager.SENSOR_DELAY_NORMAL);
        //ACELEROMETRO
        accelerometerSensor = Sensor.TYPE_ACCELEROMETER;
        sensorManager.registerListener(sensorEventListener,
sensorManager.getDefaultSensor(accelerometerSensor),
SensorManager.SENSOR_DELAY_NORMAL);
```

Con este código se registra el sensor orientación y el acelerómetro indicando la tasa de actualización que se desea. En este caso para los dos será la misma tasa que es la tasa por defecto.

```
final SensorEventListener sensorEventListener = new SensorEventListener() {

            public void onSensorChanged(SensorEvent sensorEvent) {

                  if (sensorEvent.sensor.getType() == Sensor.TYPE_ORIENTATION){

                  […]

                }else if (sensorEvent.sensor.getType() ==
Sensor.TYPE_ACCELEROMETER) {

                  […]

                }

          }
```

El método *onSensorChanged* recibe las actualizaciones cuando los datos de los sensores cambian filtrando según si es el sensor orientación o el acelerómetro.

5.2.1 GPS

```
        locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
//es el servicio de localización de Android
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
2000, 2, locationListener);
```

Utiliza el servicio de localización de Android y solicita las actualizaciones de ubicación que se notificarán mediante *locationListener.*

Dentro del método *LocationListener* se encuentra un método llamado *onLocationChanged* que se invoca cada vez que el intervalo de tiempo mínimo tiene lugar o el dispositivo se mueve la distancia mínima especificada o más.

5.3 Algoritmo para calcular la distancia.

La aplicación calcula la distancia al delfín. Para esto será necesaria la altura del usuario respecto a la superficie del mar, se introducirá mediante el teclado del dispositivo haciendo uso de un *editText* y un *Button* que al pulsarlo ejecutará el código necesario para el cálculo.

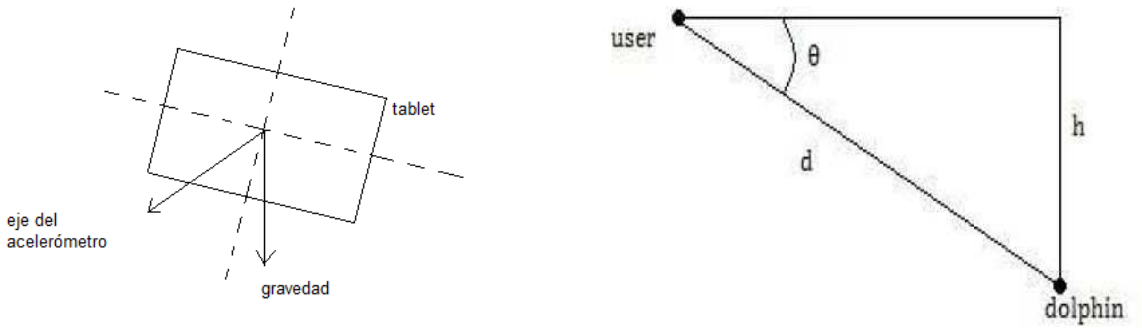


Imagen 8: Diagramas del cálculo de la distancia

Para calcular el ángulo del acelerómetro:

$$\theta = \operatorname{acos}\left(\frac{angulo\ acelerómetro}{gravedad}\right)$$

Una vez obtenido el ángulo se calcula la distancia con la siguiente ecuación:

$$d = \frac{h}{\sin(\theta)}$$

6. Bibliografía

Reto Meier: “Professional Android 2 Application Development”, Indianapolis, Indiana, John Wiley Publishing, Inc., 2010

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

Reto Meier: “Professional Android 4 Application Development”, Indianapolis, Indiana, John Wiley Publishing, Inc., 2010

Raghav Sood: “Pro Android Augmented Reality”, Apress, Paul Manning, 2012.

http://developer.android.com/reference/android/view/SurfaceView.html

http://www.belatrixsf.com/index.php/spdesarrollosmoviles

http://es.wikipedia.org/wiki/Realidad_aumentada