TRABAJO DE FIN DE CARRERA

TÍTULO DEL PFC: Estudio del desarrollo de aplicaciones RA para Android

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTOR: Alejandro Andreu Ropero

DIRECTOR: Sergio Machado Sánchez

FECHA: 18 de enero de 2011

Título: Estudio del desarrollo de aplicaciones RA para Android

Autor: Alejandro Andreu Ropero

Director: Sergio Machado Sánchez

Fecha: 18 de enero de 2011

Resumen

Las tecnologías audiovisuales evolucionan día a día. Actualmente tecnologías como el 3D o la alta definición pueden dejar a más de uno sorprendido, pero el siguiente paso ya está aquí.

Existen tecnologías que a través de un dispositivo, nos introducen visualmente en realidades alternativas, viendo un mundo digital paralelo, o como la realidad aumentada, que combina la visión del mundo real con una capa superpuesta virtual. Como su propio nombre indica esta tecnología intenta mejorar nuestra visión del mundo real con información o con elementos 3D.

El desarrollo de esta tecnología que se ha vivido en los últimos años en gran parte es debido a la existencia de un dispositivo con las características necesarias para hacer uso de ella. Ese dispositivo es el teléfono móvil, que en la actualidad no solo nos ofrece internet, GPS, pantallas táctiles, cámara o incluso sistema operativo, si no que nos ofrece un mercado enorme y que parece no tener techo.

En este proyecto haremos una introducción a la realidad aumentada, explicando los diferentes métodos para localizar los objetos que existen hoy en día.

También desarrollaremos una aplicación para teléfonos móviles con Sistema Operativo Android que haga uso de la realidad aumentada, que mediante nuestra posición GPS situé un objeto 3D en los puntos de interés de el Parc Mediterrani del Llobregat.

Por último haremos un estudio de Layar, una aplicación para Android que consiste en un buscador que hace uso de la realidad aumentada para situar puntos de interés.

Title: Study of development for RA applications for Android

Author: Alejandro Andreu Ropero

Director: Sergio Machado Sánchez

Date: January, 18th 2011

Overview

Audiovisual technologies evolving day by day, as the 3D technologies currently or high definition may leave more of a surprise, but the next step is here.

There are technologies through a device, we introduce visually alternative realities, seeing a parallel digital world, or as augmented reality, which combines real-world view with a virtual overlay. As its name suggests, the technology aims to improve our view of the real world with information or with 3D elements.

The development of this technology that has been experienced in recent years is largely due to the existence of a device with the features necessary to make use of it. That device is the mobile phone, which now offers not only internet, GPS, touch screen, camera, or even SO, but it offers a huge market and that seems to have no ceiling.

In this project we will make an introduction to augmented reality explaining the different methods for locating objects that exist today.

We will also develop one application for mobile phones with Android OS making use of augmented reality, that through our GPS to position 3D object at point of interest Mediterrani Parc del Llobregat.

Finally we will study Layar, an application for Android which is a search engine that uses augmented reality to locate points of interest.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En esta última década se han desarrollado aplicaciones las cuales a través de un dispositivo ofrecen una visión del mundo real, a través de las imágenes capturadas por la cámara, que se combinan con una capa virtual en la que se puede ver desde información sobre nuestro entorno real hasta gráficos 3D. Esta tecnología se llama realidad aumentada y poco a poco está llegando a nuestra vida diaria.

Como hemos comentado para mostrar la realidad aumentada hace falta un dispositivo y como mínimo este dispositivo tiene que contar con una cámara y con un cierto nivel computacional. El teléfono móvil, además de estas características, nos da acceso a un gran número de usuarios.

Los teléfonos móviles de hoy en día están más cerca de los ordenadores que tenemos en casa que del primer teléfono móvil, que solo permitía llamar. No sólo llevan cámara, sino que también nos ofrecen sistema operativo, posibilidad de usar aplicaciones con alto rendimiento gráfico, movilidad, servicio de posicionamiento GPS y acceso a internet.

Este proyecto tiene como objetivo crear una aplicación que utilice la Realidad Aumentada, de manera que podamos observar lo que captura la cámara de vídeo con una capa virtual superpuesta y donde al aproximarse a los puntos de interés del Campus, como por ejemplo la Biblioteca o la Cafetería, aparezca un objeto 3D.

La plataforma elegida para su desarrollo serán los teléfonos móviles con Sistema Operativo Android. Los motivos por los que elegimos Android son:

- Es libre, así que cualquier desarrollador puede bajarse su código fuente y crear aplicaciones.

- Está en pleno auge y cuenta con una gran comunidad de desarrolladores.

El proyecto está dividido en 5 capítulos que se estructuran de la siguiente manera:

En el primer capítulo haremos una breve introducción a la Realidad Aumentada y los tipos que hay, dando un pequeño repaso a su reciente historia.

En el segundo, explicaremos qué es Android, su historia, y explicaremos cómo crear un entorno de desarrollo para aplicaciones Android mediante el IDE Eclipse y cómo probar nuestras aplicaciones en el ordenador mediante un emulador de Android.

En el tercero explicaremos el objetivo de nuestra aplicación, a la cual hemos llamado “RACampusCastelldefels”, y como acceder a los recursos necesarios para que funcione.

En el cuarto haremos un estudio de Layar, una aplicación para Android que hace uso de la Realidad Aumentada para localizar puntos de interés de los usuarios. Explicaremos como hacer una capa y como testarla.

En el último capítulo daremos las conclusiones que han surgido de elaborar este proyecto, la ambientalización y las posibles mejoras que se nos ocurren para mejorar la aplicación.

CAPITULO 1. REALIDAD AUMENTADA

1.1. Introducción

La Realidad Aumentada (RA) consiste en añadir información o imágenes virtuales a la visión del mundo real. Así, podríamos decir que no sustituye a la realidad física, si no que le añade información ya sea en formato de datos o gráficos. Para ello la tecnología hace uso de un dispositivo que, mediante una aplicación, captura imágenes del mundo real y le sobreimprime la capa virtual, presentándonos el resultado por pantalla, y permitiéndonos o no interactuar con la información obtenida.

Ilustración 1 Muestra de Realidad Aumentada mediante Markers

El problema de la RA es situar los objetos 3D, como veremos a continuación existen diferentes métodos.

1.1.1. RA usando markers

El mark, o marker file, es un patrón en forma de imagen que se imprime para poder situar un objeto 3D en la Realidad Aumentada.

Ilustración 2 Ejemplo de dos markers

Como podemos observar está compuesto por un marco y un símbolo en su interior, de tamaño y forma característica. El software se basa en el reconocimiento de estos objetos para determinar que objeto 3D se representa y su posición.

La dificultad que surge al utilizar markers es que necesitas tener la imagen impresa en alguna superficie para poder ver el objeto 3D, y si la imagen no se reconoce, ya sea porque esté oculta una parte o porque esté deteriorada, no se mostrará nada. Por otro lado el uso de markers nos permite situar con mucha más exactitud el objeto 3D.

Una aplicación que utiliza este método es NayarToolkit, que se puede descargar gratuitamente para Android y nos permite mostrar algunos personajes animados en 3D usando la RA por markers.

Otro ejemplo de RA mediante markers es el videojuego que ha sacado la compañía PlayStation para su consola PSP: Invizimals, el cual nos muestra algún personaje cuando enfocamos con la cámara de la PSP el marcador que viene con el videojuego.

Ilustración 3 Imagen del videojuego de la PSP Invizimals

1.1.2. RA usando la posición GPS

Otra opción que existe es situar los objetos en una cierta posición GPS, y que al aproximarte a ella puedas ver los objetos a través del dispositivo.

El problema de este modo es la poca precisión de los GPS existentes hoy en día, de manera que si nuestra posición varía, aunque sea un poco, podemos ver saltar los objetos de un punto a otro. Por otro lado el uso del posicionamiento mediante GPS nos permitiría evitar las complicaciones de los marker.

Un ejemplo de este modo seria Layar, el cual explicamos en el capítulo 4.

1.1.3. RA usando reconocimiento de objetos

Esta opción es la más moderna y se basa en el reconocimiento de objetos conocidos, por ejemplo, el de la cara de una persona o el de la forma de un edificio u objeto.

El problema de este método es la dificultad de reconocer la forma del objeto, ya que en el proceso se tendría que cotejar las formas de la imagen con una base de datos, y contra más objetos se quieran reconocer, más grande será la base de datos.

Un ejemplo sería que, al reconocer la cara de alguna persona, nos apareciese un icono para poder enlazar a las redes sociales a las que pertenece o su información personal, o por ejemplo que al mirar a la Sagrada Familia reconociese su perfil y nos informe de su historia y enlaces de interés.

Ejemplo de RA en un vídeo: http://www.youtube.com/watch?v=fSfKlCmYcLc

1.2. Historia

La Realidad Aumentada es una tecnología relativamente joven, prueba de ello es que el término se creó por primera vez en 1992 por Tom Caudell, año en el que se creó el primero prototipo, KARMA.

En el año 1999 se creó ARToolKit en Java, que es una librería que nos facilita el uso de la Realidad Aumentada mediante markers.

No es hasta 2009 cuando se crea una comunidad para desarrolladores de Realidad Aumentada bajo este Logo:

Ilustración 4 Logotipo de RA

En la actualidad podemos ver como la Realidad Aumentada empieza a hacer acto de presencia en nuestra vida. Su uso va, desde aplicaciones en la cirugía, mostrando en los visores del cirujano las constantes del paciente, hasta dispositivos que permiten que un usuario se pruebe gafas sin tener que ponérselas físicamente.

En cuanto al futuro de la Realidad Aumentada podemos vislumbrar que es muy prometedor, existiendo plataformas como el teléfono móvil y ofreciendo un gran abanico de posibilidades en diversos aspectos de la vida diaria.

CAPITULO 2. SISTEMA OPERATIVO ANDROID

2.1. Introducción

Desde que en 1983 apareció el primer teléfono móvil (el Motorola DynaTAC) hasta el día de hoy la telefonía móvil ha cambiado mucho. El móvil ha pasado de ser una simple herramienta para hacer llamadas, a sofisticados dispositivos informatizados que incorporan cámara fotográfica, agenda, acceso a Internet, reproducción de vídeo e incluso GPS y reproductor mp3, entre otros, y cada día nos sorprenden nuevas tecnologías que se van introduciendo como por ejemplo la Realidad Aumentada.

Y es que el desarrollo tanto de hardware como de software para móviles se hace día a día más interesante, puesto que es un mercado que a pesar de tener una cantidad de usuarios antes nunca conocida (en el 2010 se habla de más de 5000 millones de móviles para 6500 millones de personas en el mundo) sigue en auge. Los motivos son la propagación de la cobertura, el abaratamiento de los dispositivos y el cambio de la sociedad tanto como el del modo de comunicación.

Como dispositivo informático, el teléfono móvil requiere sistema operativo. Algunos de esos sistemas son iPhone OS, Windows Mobile, Symbian, MeeGo, BlackBerry y Android.

En este proyecto trabajaremos con Android. Según wikipedia: “Android es *un sistema operativo orientado a dispositivos móviles y que usa una versión modificada del núcleo Linux. Es desarrollado por la Open Handset Alliance, que aglutina a fabricantes de software y hardware, entre los que destacan Google, T-Mobile, HTC, Qualcomm y Motorola entre otros. Android hace parte de los sistemas operativos con interfaz natural de usuario”.*

El porqué de la selección de Android es que se trata de software libre, y eso significa que se puede descargar su código y trabajar o modificarlo para crear tus propias aplicaciones.

Y bien, ¿cuáles son las pegas de Android? La más importante son sus diferentes versiones, que a pesar de tener pocos años de existencia ya tiene varias. Las aplicaciones se crean para una sola versión, que puede ser o no soportado por versiones anteriores y posteriores de Android, así que hoy por hoy tienes que saber muy bien para que versión trabajas, teniendo en cuenta número de clientes (no trabajan todos con la ultima) y soporte a la necesidad de tu aplicación.

2.2. Historia

En julio de 2005, Google compra Android Inc., una pequeña empresa basada en Palo Alto, California, y entonces empezaron a trascender rumores acerca de que Google estaba planeando construir su propio teléfono móvil libre y hasta gratis, enfocándose en ganancias de publicidad en las búsquedas de las personas para mover un poco el status quo del mercado móvil. Obviamente, esos rumores de un móvil gratis fueron falsos pero al final Android resultó ser algo mucho más interesante y revolucionario: un sistema operativo móvil open source propulsado nada más y nada menos que por Google… así que repasemos la historia de Android por más breve que sea:

El lanzamiento inicial del Android Software Development Kit apareció en noviembre de 2007 y algún tiempo después -mediados de agosto de 2008- apareció el Android 0.9 SDK en beta. Al otro mes -fines de septiembre 2008-, finalmente lanzaron Android 1.0 SDK (Release 1). Seis meses después -principios de marzo 2009-, Google presentó la versión 1.1 de Android para el "dev phone" y la actualización incluía algunos cambios estéticos menores, además de soporte para "búsquedas por voz", aplicaciones de pago en Android Market, arreglos en el reloj alarma, mejoras en Gmail y demás.

A mediados de mayo 2009, Google lanza la versión 1.5 de Android OS (llamada Cupcake) con su respectivo SDK que incluía: grabación de video, soporte para Estéreo Bluetooth, sistema de teclado personalizable en pantalla, reconocimiento de voz y el AppWidget framework que permitió que los desarrolladores puedan crear sus propios widgets para la página principal. Android 1.5 fue la versión que más personas usaron para iniciarse en Android (con el T-Mobile G1 y HTC Dream en USA) y sigue siendo actualmente una versión que se encuentra disponible en muchos móviles Android como el HTC Hero o varios de los nuevos MOTOBLUR como el Motorola Backflip o Motorola Dext.

Luego apareció Android 1.6 "Donut" en septiembre de 2009 con mejoras en las búsquedas, indicador de uso de batería y hasta el VPN control applet. De hecho, esta versión fue tan buena que todos los Android que no tienen una interfaz personalizada como HTC Sense o Motoblur ahora corren 1.6, incluyendo el T-Mobile G1, y en la actualidad sigue siendo la versión más popular.

Para llevar las cosas más allá, el Motorola Droid (Motorola Milestone) fue lanzado con Android 2.0 "Eclair" en el que se podían encontrar aplicaciones precargadas que requerían un hardware mucho más rápido que la generación anterior de teléfonos móviles con Android (un mes luego, salió 2.0.1, una pequeña actualización).

Poco después, el Google Nexus One (el cuál marcó un antes y un después ya que Google trató de venderlo por su cuenta y liberado, además de en algunas operadoras) llegó con Android 2.1 (el cual algunos llamaron "Flan" pero Google sigue considerándolo parte de "Eclair") con nuevas capacidades 3D, live

wallpapers y lo que significó la gran mejora de la plataforma desde 1.6. De hecho, todos están pidiendo que les actualicen a 2.1 sus propios dispositivos con Android pero es probable que muchas de las novedades no funcionasen bien en versiones anteriores.

Si miramos al futuro de Android podemos ver como Android Market es la tienda de aplicaciones que más crece, llegando a las 40 mil aplicaciones. Android es el sistema operativo que más está creciendo en Estados Unidos rivalizando con iPhone. Motorola junto con algunos otros fabricantes están propulsando el desembarco en América Latina de Android con equipos económicos, y por el otro lado algunos se quejan de la fragmentación de la plataforma debido a las diferentes versiones. Pero lo cierto es que ya se está empezando a desarrollar el know how para ofrecer las actualizaciones de manera que sean compatibles para la versión 2.1 en adelante.

2.3. Creación de un entorno de desarrollo para Android

Así pues vamos a pasar a ver que tenemos que hacer para desarrollar aplicaciones en Android. Lo primero será la configuración del entorno de desarrollo para Android mediante el IDE Eclipse. En este proyecto cabe aclarar que trabajaremos con Windows.

2.3.1. Instalación de java

Es muy probable que nuestro PC ya cuente con Java, pero por si acaso descargaremos una versión actualizada desde su web oficial:

http://www.java.com/es/download/

El archivo descargado lo ejecutaremos y de esa manera instalaremos Java.

2.3.2. Descarga del SDK de Android

Como hemos comentado Android es un software libre, así que cualquier desarrollador puede bajarse el SDK, que contiene su API, y desarrollar aplicaciones. En nuestro caso nos descargaremos la versión para Windows de este enlace:

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

El archivo descargado es un ZIP, lo descomprimiremos y lo dejaremos en la ubicación deseada, que no es importante ahora pero más tarde la utilizaremos. En nuestro caso la dejaremos en “c:”.

2.3.3. Descarga y configuración de eclipse

Eclipse es una plataforma de desarrollo open source basada en Java. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los usuarios. En si mismo Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados (plug-in).

Hay plug-ins para el desarrollo de Java (JDT Java Development Tools) así como para el desarrollo en C/C++, COBOL o Android, entre muchos otros.

Para descargar Eclipse visitaremos su página y descargaremos la versión "Classic", en nuestro caso es la versión 3.6.1:

http://www.eclipse.org/downloads/

Lo que se descargará será un ZIP, lo descomprimiremos y dejaremos el contenido en la ubicación que mejor nos parezca, en nuestro caso será en "c:". Dentro de la carpeta descomprimida habrá un exe, recomiendo crear un acceso directo en el escritorio para no tener que acceder siempre a la carpeta.

Si ejecutamos el eclipse.exe nos preguntará que workspace utiliza. El workspace es el directorio donde eclipse guardará el código de las aplicaciones que desarrollemos, y en nuestro caso crearemos uno nuevo en la misma carpeta del eclipse "c:/eclipse/WorkSpace".

Al entrar podremos ver la interfaz de Eclipse:

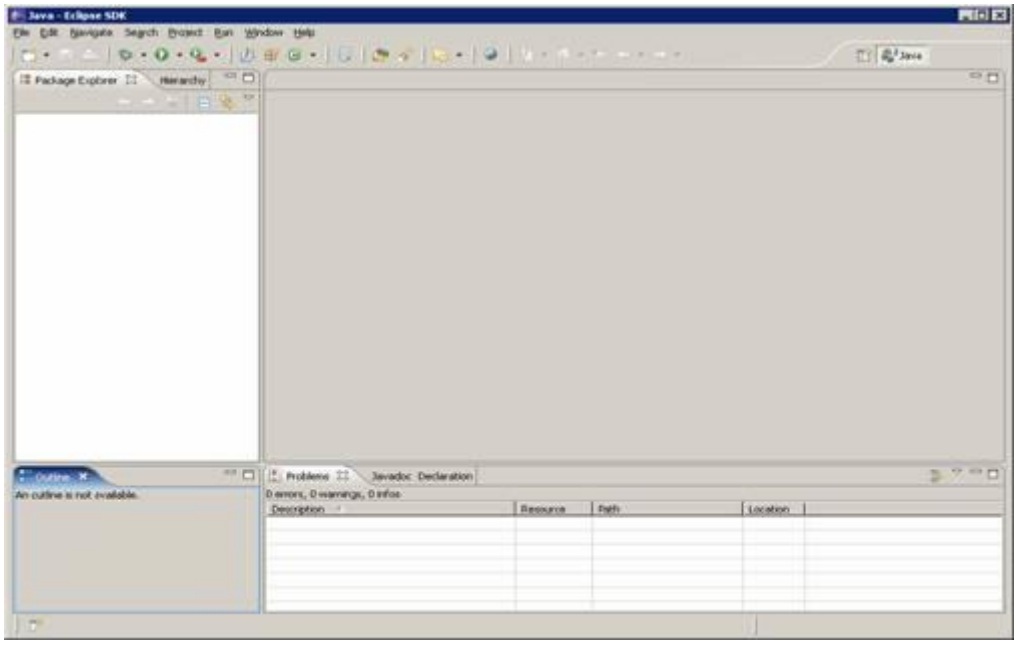

Ilustración 5 Interfaz de Eclipse

Ahora configuraremos el plugin de Android para poder programar en esa plataforma.

Accederemos en la interfaz de Eclipse al menú “Help->Install new Software…”, clicaremos en “Add…”. En la siguiente ventana en “Nombre” pondremos “Android” y en “Location” la URL:

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

Una vez hecho esto, en la ventana inferior habrán aparecido algunas opciones para seleccionar, las seleccionaremos todas y le daremos a “Next”.

Una vez descargado e instalado reiniciamos Eclipse como nos pide, y al volverlo al iniciar ya tendremos las funcionalidades del plugin disponibles.

Lo siguiente será referenciar SDK Android, para ello accederemos en el menú a: “Windows -> Preference”, le daremos a Android y en “SDK Location” pondremos el directorio donde hemos dejado el SDK (Apartado 2.3.2). Si le damos a Apply y todo lo anterior lo hemos esta seguido al pie de la letra nos reconocerá el SDK y nos aparecerá una ventana como la siguiente:

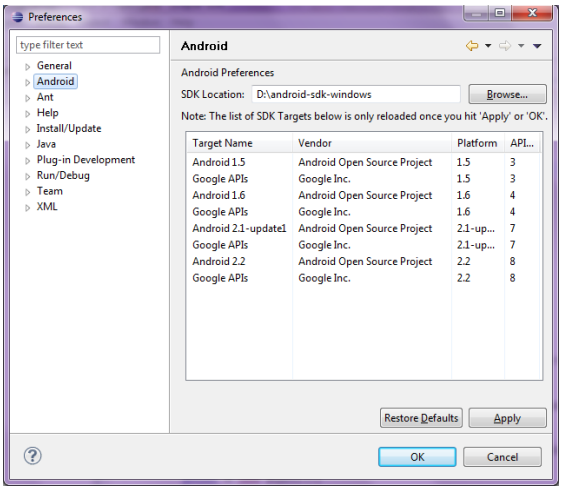

Ilustración 6 Ventana donde aparecen versiones disponibles de Android

Llegados a este punto ya estamos preparados para trabajar, pero antes intentaremos bajarnos las actualizaciones por si acaso alguna de las versiones descargadas ha sido mejorada, para ellos iremos al menú “Help -> Check for Update” y si nos aparece alguna opción la descargaremos.

Si seguimos estos pasos correctamente ya tendremos Eclipse preparado para crear aplicaciones Android.

CAPITULO 3. APLICACIÓN RACampusCastelldefels

3.1. Objetivo

Como se ha comentado en la introducción el objetivo de este proyecto es el desarrollo de una aplicación de Realidad Aumentada para teléfonos móviles con sistema operativo Android.

En la aplicación podremos observar por pantalla del dispositivo nuestro entorno real a través de las capturas de la cámara y una capa virtual superpuesta con varios objetos en 3D situados en posiciones de interés del Parc Mediterrani de la Tecnologia.

Los objetos 3D solo serán visibles cuando el dispositivo esté a menos de 100 metros de distancia y el dispositivo móvil esté enfocando en esa dirección.

De este modo podemos determinar que nuestra aplicación necesitará:

- Poder determinar la posición del dispositivo.
- Saber la dirección hacia la que está enfocado el dispositivo.
- Mostrar por pantalla las imágenes capturadas por la cámara del dispositivo.
- Crear una capa virtual superpuesta que en los puntos de Interés muestre un objeto 3D.

3.2. Determinar Posición

Para saber la posición del dispositivo utilizaremos un recurso del que disponen la mayoría de teléfonos móviles, el GPS.

El GPS nos proporcionará una latitud y una longitud con la que podremos situar el dispositivo móvil, el problema es la precisión de la medida que depende de algunos factores como el número de satélites disponibles o interferencias.

La posición GPS de los puntos de interés será fija, para facilitar nuestro trabajo en nuestra aplicación hemos creado la clase objeto Punto.java, en la que guardaremos la latitud, longitud entre otras propiedades.

3.2.1 Coordenadas GPS

El Sistema de posicionamiento global o GPS es un sistema global de navegación por satélite que permite determinar la posición del dispositivo por todo el mundo y que se determinan a partir de la distancia a los satélites que forman parte de este sistema.

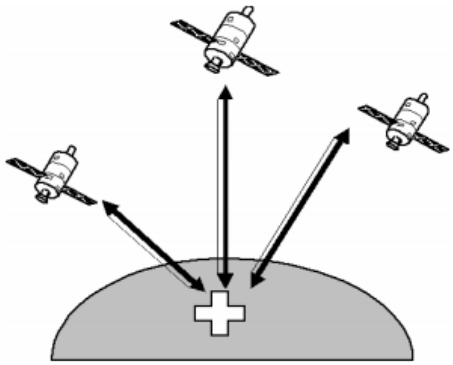

Ilustración 7 Intercambio de señales entre dispositivo y satélites en el GPS

El sistema GPS fue puesto en marcha por el ejército de los EUA y en 2009 se liberalizo. El sistema consta de 24 satélites geoestacionarios que cubren toda la superficie de la tierra, de las estaciones terrestres que controlan los satélites y se precisa de un terminal receptor con GPS para poder acceder a la señal.

La posición del GPS se basa en las coordenadas geográficas y vienen dada por las coordenadas angulares latitud, que mide el ángulo entre cualquier punto y el ecuador, y por la longitud, que mide el ángulo entre el punto y el meridiano de Greenwich.

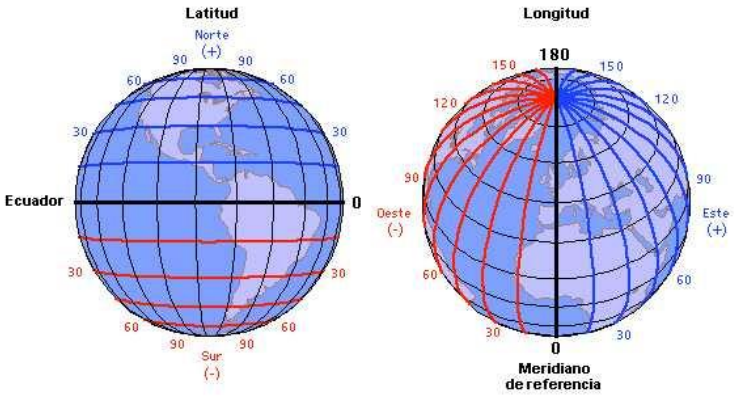

Ilustración 8 Representación de la tierra por latitudes y longitudes

Por ejemplo en el Parc Mediterrani de la Tecnologia podemos encontrar el punto con las coordenadas (41.27548027005814, 1.9875195622444153).

Uno de los problemas de estas coordenadas es que nos devuelven la posición que tenemos en la esfera terrestre, por lo que no nos proporcionará una posición (x,y). Eso es lo que nos interesaría para situar los objetos respecto al dispositivo y por eso decidimos pasar las coordenadas de GPS a el sistema UTM que sí las proporciona en ese formato.

El otro problema es la precisión, que depende tanto de la cobertura de los satélites con los que cuenta el sistema, como de otros factores que pueden hacer variar la posición en varios metros.

3.2.2 Acceso a las coordenadas GPS

El acceso a la posición GPS en Android se hace a través de la librería Android.Location mediante las clases LocationManager y LocationListener.

Primero que tenemos que hacer es añadir el permiso en el AndroidManifest.xml para que la aplicación pueda tener acceso al GPS. Para ello añadiremos la línea:

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

LocationManager proporciona acceso al servicio GPS y nos permite obtener actualizaciones periódicas de la situación geográfica del dispositivo. Se consigue a través de getSystemService(Context.LOCATION_SERVICE).

Utilizaremos la función requestLocationUpdates para asignar un LocationListener como oyente del proveedor y configurar el tiempo o distancia entre las actualizaciones.

LocationListener se utiliza para recibir las notificaciones de cambio del LocationManager. Es obligatorio declarar sus métodos onLocationChanged, onProvaiderDisabled, onStatusChanged y onProvaiderEnabled. De ellos sólo modificaremos el onLocationChanged.

En el onLocationChanged recibiremos un objeto location, que es una clase que contiene entre otros parámetros la latitud y longitud nueva del dispositivo.

3.2.3 Coordenadas UTM

El Sistema de Coordenadas Universal Tranversal de Mercator o UTM es un sistema de coordenadas que mediante una proyección, representa la superficie esférica terrestre sobre una superficie cilíndrica, tangente al meridiano, que al desplegarse genera un mapa terrestre plano. El problema es que de este modo las zonas más cercanas a los polos se hacen más grandes.

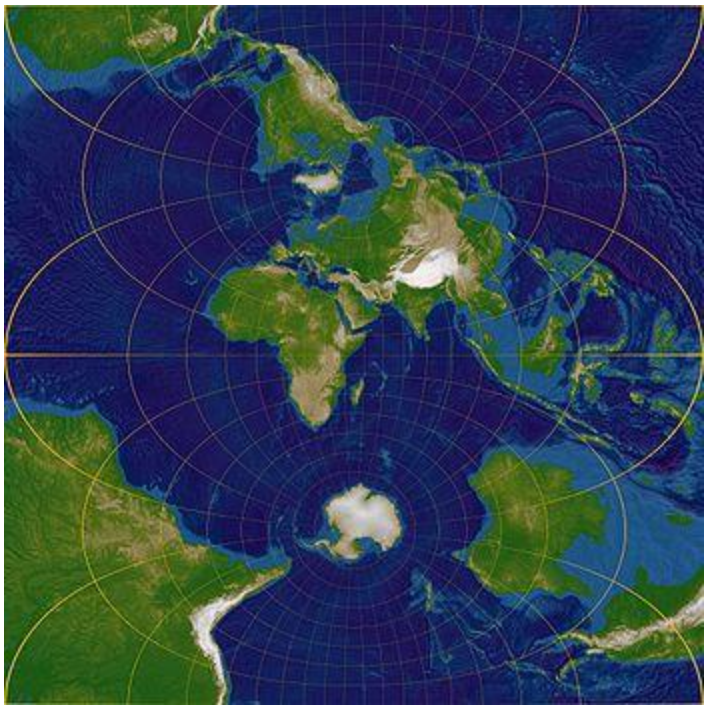

Ilustración 9 Proyección de la esfera terrestre en un cilindro

Es importante aclarar que el UTM localiza la posición dentro de una zona y con unas coordenadas (x, y), en lugar de devolverte un punto. De esta manera contra más pequeño sea el cuadrante más precisión tendremos.

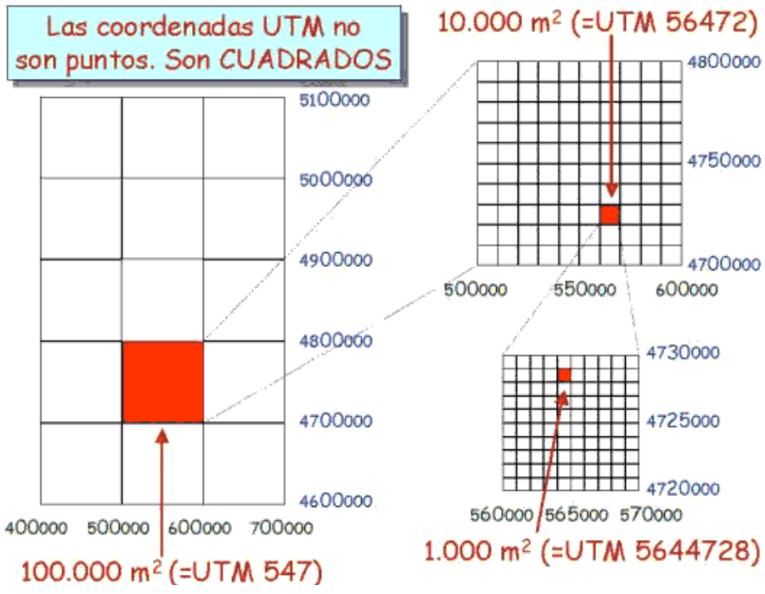

Ilustración 10 Ejemplo UTM

Como se puede observar en la ilustración 10 contra más precisión tenga la medida, más pequeña es la zona donde localizamos al dispositivo y más precisión tendrá. De este modo podemos localizar en dispositivo en un eje (x, y) en metros.

En las zonas limítrofes existe una extensión que se sobrepone unas con otras de ese modo algún punto puede tener su posición en 2 zonas distintas.

El punto que tenemos en el apartado 3.2.1, en UTM tiene las coordenadas 31T E 415203'78 N 4569832'99.

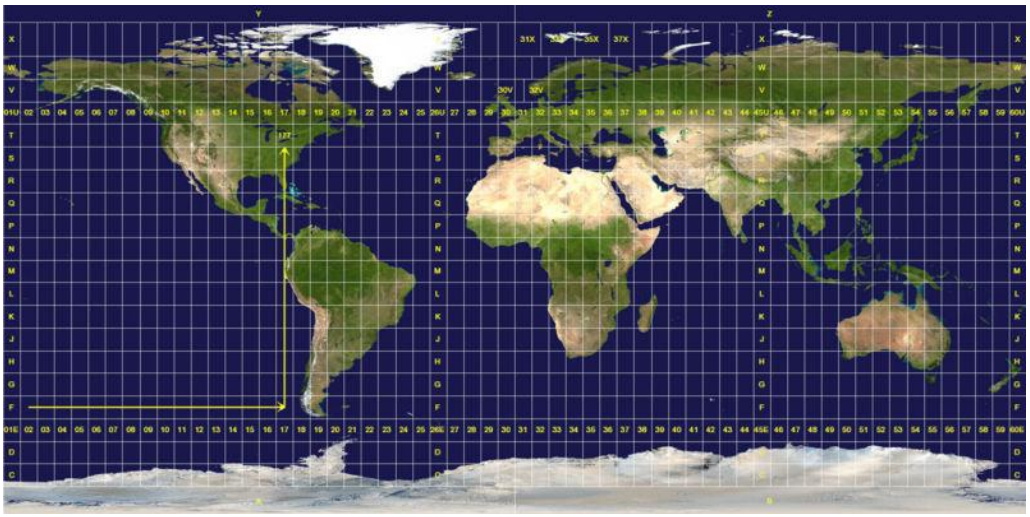

Ilustración 11 Zonas UTM

Para distancias grandes el UTM tiene el problema de que hay un eje diferente para cada zona y el cálculo se complica, pero en nuestro caso y para medidas no superiores a 100 metros, el objeto se da por defecto en nuestra misma zona.

De todos modos hemos decidido montarnos nuestro propio tablero, de manera que su eje de coordenadas está situado en el punto UTM 31T E 414920 N 4569708 como se puede ver en la siguiente imagen, de este modo las posiciones serán relativas a ese punto. También podemos observar los 4 puntos de interés que hemos colocado.

Ilustración 12 Origen de coordenadas y puntos en el Campus

Para trabajar con las coordenadas UTM utilizaremos la clase Coordinate Conversion.java que nos podemos descargar de:

http://www.ibm.com/developerworks/java/library/j-coordconvert/index.html

Gracias a esta clase podremos pasar coordenadas de UTM a GPS y a la inversa, para pasar de GPS a UTM utilizaremos la función latLon2UTM, a la que se le pasan como parámetros la longitud y latitud y te devuelve un String con la coordenada UTM.

3.2.4 Cálculo de la distancia entre dos puntos

Para determinar si la distancia entre los puntos es menor o superior a 100m utilizaremos las coordenadas GPS ya que es más fácil que con las UTM.

La misma librería Android.location nos permite calcular la distancia entre 2 puntos mediante la función distanceTo.

3.2.5 Simular el GPS para el emulador

Eclipse nos ofrece la herramienta DDMS para simular el envío de unas coordenadas GPS.

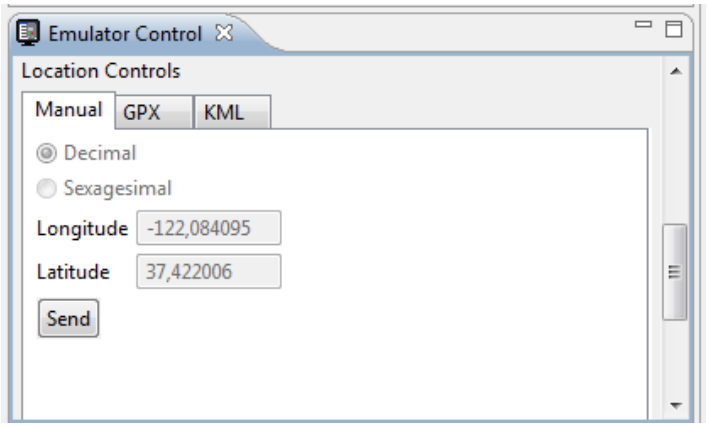

Ilustración 13 Ventana de control del emulador de Android

Debido a un error de Eclipse, si el Windows con el que trabaja el ordenador está en otro idioma que no sea Inglés, las coordenadas llegan con un error considerable. Esto se puede solucionar añadiendo unas líneas de código en el archivo eclipse.ini. Las líneas son:

```
-Duser.country=US
-Duser.language=en
```

Otro modo de enviar las coordenadas es establecer un Telnet con el emulador desde la consola mediante el comando *telnet localhost 5554* y después el comando *geo fix longitud latitud.*

3.2.6 Obtener las coordenadas desde un GPX

La precisión que puede llegar a tener el receptor GPS de un dispositivo móvil puede no ser suficientemente buena, y por ello hemos decidido hacer un segundo código utilizando como fuente de puntos GPS un archivo GPX, obtenido mediante un dispositivo el cual te proporciona una lista de puntos GPS a partir de una ruta realizada por el usuario con mayor precisión que el teléfono móvil.

El GPX es un formato para guardar trazas de puntos GPS que está basado en un XML. Aquí podemos observar un ejemplo con solo 2 puntos GPS:

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx
  version="1.0"
  creator="GPSBabel - http://www.gpsbabel.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.topografix.com/GPX/1/0"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/0
http://www.topografix.com/GPX/1/0/gpx.xsd">
<time>2010-11-15T10:03:59Z</time>
<bounds minlat="41.274829851" minlon="1.986018866"
```

```
maxlat="41.275832914" maxlon="1.987823825"/>
<trk>
  <name>15-NOV-10 #2</name>
<number>1</number>
<trkseg>
<trkpt lat="41.275780443" lon="1.987174395">
  <ele>11.673218</ele>
</trkpt>
<trkpt lat="41.275782287" lon="1.987173641">
  <ele>11.673218</ele>
</trkpt>
</trkseg>
</trk>
</gpx>
```

Un XML se puede leer mediante SAX o DOM. La diferencia es que SAX lee todo el archivo y después continúa, y DOM lo va leyendo mientras trabaja. En este caso nos interesa SAX debido a que el GPX no contiene mucha información.

El acceso al archivo GPX lo haremos a través de internet, con lo que necesitaremos tener el archivo disponible en algún servidor.

3.3. Acceso a la cámara

Modificando el código de min3d hemos conseguido mostrar por pantalla los objetos 3d teniendo como fondo las imágenes capturadas por la cámara.

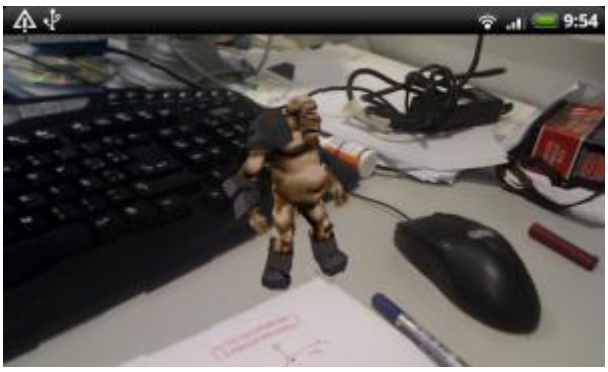

Ilustración 14 Combinación de un objeto 3D con las capturas de la cámara.

3.4. Mostrar objetos en 3D

Para mostrar los objetos 3D utilizaremos la librería min3d. El código de min3d está basado en OpenGL para Android y es un código apadrinado por Google.

Para poder trabajar con min3d lo primero que tenemos que hacer es que nuestra actividad extienda de RenderActivity. De esta manera podremos trabajar con la escena (scene), que consta de la cámara (camera) y de los objetos 3D (childs). Todas las modificaciones de la escena se harán en la clase principal.

3.4.1. Scene

Scene es algo así como el escenario en el que trabajamos. En él tendremos una cámara que tendrá el papel de observador, y que tendrá una posición y un target, ambas establecidas en un eje x, y, z.

El target es hacia el punto que mira la cámara. El cálculo de sus coordenadas será la suma de la posición y del módulo respecto el ángulo de visión.

La posición donde se muestran los objetos vendrá determinada por la distancia y la orientación hacia la que mira el dispositivo. De esta manera, moviéndonos, tendremos diferentes perspectivas de los cubos que tengamos a la vista.

3.4.1. Objetos 3D

Min3D nos proporciona algunos objetos 3D por defecto, en nuestro caso utilizaremos un cubo con las caras de diferentes colores para identificar los puntos de interés creado directamente por OpenGL, pero podríamos mostrar archivos con formato obj o md2 por ejemplo.

Cuando un punto entra dentro del radio de 100m desde el dispositivo, tendremos que crear y situar el objeto 3D en su posición relativa a la nuestra determinando sus coordenadas (x, y, z).

3.5. Sensores del teléfono móvil

En nuestra aplicación necesitaremos saber la dirección hacia la que el dispositivo móvil está enfocando. Eso lo conseguiremos gracias a los sensores de los que disponen los dispositivos móviles de hoy en día.

3.5.1. Acceso a los sensores del teléfono móvil

Para acceder al sensor del teléfono móvil Android nos proporciona las clases Android.hardware.SensorManager y Android.hardware.SensorListener.

Al implementar en nuestra aplicación el SensorListener tendremos que añadir la función onSensorChanged que es la que se encarga de recibir los datos enviados por los sensores que son solicitados y administrados por la clase SensorManager.

El cambiar el valor de los sensores cuando el móvil se gira se ejecutará el código que hay dentro de onSensorChanged.

Hemos decidido actuar sólo cuando el giro es mayor de 5 grados, ya que el sensor, al ser bastante sensible, se producen cambios casi sin moverlo.

3.5.2. Simulación de los sensores en el emulador

SensorSimulator de OpenIntents es una aplicación que nos permite simular físicamente el móvil de manera que cambien los sensores de orientación o movimiento en tiempo real.

Para poder usarlo primero tenemos que descargarnos la aplicación sensorsimulator y la librería openintents de las páginas oficiales:

http://code.google.com/p/openintents/downloads/detail?name=sensorsimulator-1.0.0-beta1.zip&can=2&q=

http://code.google.com/p/openintents/downloads/detail?name=openintents-binary-0.9.0.zip&can=1&q=

En el zip del sensorsimulator.zip podremos encontrar una carpeta llamada bin donde tenemos un jar y un apk. Los 2 nos ayudarán a configurar el emulador para que pueda extraer los datos de la aplicación jar. Primero tendremos que instalar el apk en el emulador de Android. Para ello seguiremos los siguientes pasos:

1- Lanzaremos el emulador de Android.
2- Colocaremos el archivo apk en la carpeta tolos del sdk de Android.
3- Después accederemos desde la consola de Windows a la carpeta tolos de Android y ejecutaremos el siguiente comando:

adb install SensorSimulatorSetting.apk

Ilustración 15 Icono de SensorSimulator en el menú del emulador

4- Arrancaremos desde la misma consola de Windows el archivo jar que hemos descargado mediante el comando:

java –jar sensorsimlator.jar

Al hacerlo nos aparecerá la aplicación que se asemejará a esto:

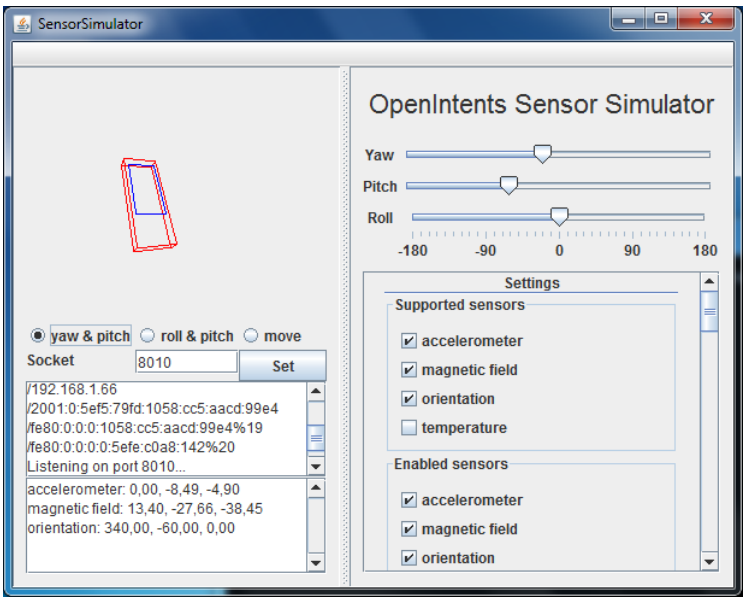

Ilustración 16 Aplicación SensorSimulator

Como podemos ver tenemos un objeto 3D que parece un móvil que podremos desplazar, y al rotarlo cambiarán los valores de los sensores. También podemos modificar el puerto e IP por el que transmitirá.

5- Dentro del emulador de Android ejecutaremos la aplicación que hemos instalado SensorSimulatorSetting, que tiene esta apariencia:

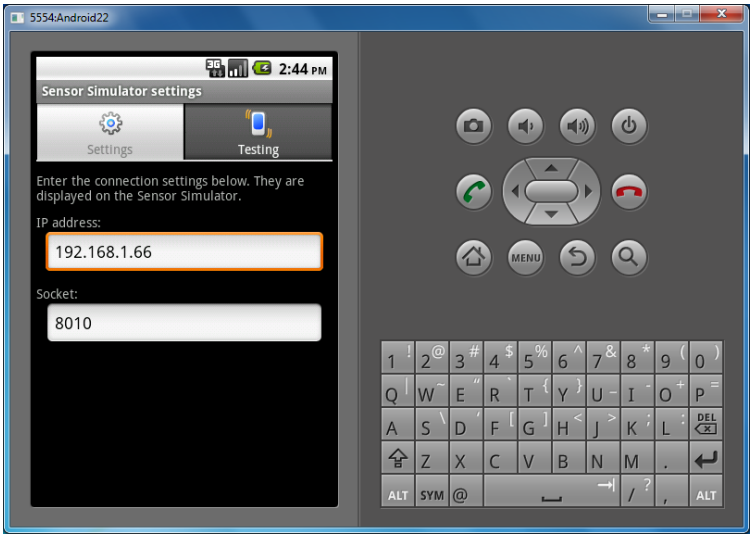

Ilustración 17 Pestaña Settings de SensorSimulator

Tendremos que cambiar la IP y el Puerto por los que se mostraban en la aplicación jar.

6- Para poder acceder a los datos enviados tendremos que apagar el cortafuegos. Después accedemos a la pestaña de Testing y si todo se ha hecho correctamente, al darle a connect podremos observar los valores de los sensores de la aplicación jar.

Ilustración 18 Pestaña Testing de SensorSimulator

A partir de ahora en las aplicaciones que queramos que usen el sensorsimulator deberán incluir la librería openintens antes nombrada y en lugar de conectarse al Hardware del sistema se conectarán al puerto que usa la aplicación sensorsimulator.jar, por lo que tendremos que tenerla encendida.

3.6. Ciclo de vida de una aplicación Android

Una aplicación Android funciona como un proceso Linux, por tanto, una característica fundamental de Android es que el tiempo y ciclo de vida de una aplicación no están controlados por la misma aplicación, sino que lo determina el sistema a partir de una combinación de estados como pueden ser qué aplicaciones están funcionando, qué prioridad tienen para el usuario y cuánta memoria queda disponible en el sistema.

A la hora de programar es muy importante controlar bien el ciclo de vida de una aplicación, y aún lo es más cuando estamos hablando de móviles. Toda aplicación que está funcionando en el móvil aumenta el consumo, con lo que si no controlas los procesos que están encendidos es fácil agotar rápidamente la batería del móvil, lo que puede ser muy molesto para el usuario. Así pues vamos a ver cómo podemos programar de manera que nuestra aplicación consuma el mínimo de recursos tanto si está encendida como pausada.

Lo correcto es que, mientras estas utilizando una aplicación, ésta pueda disponer del máximo de la CPU del móvil, pero al pausarla, la aplicación libere todos los recursos librando al móvil de una carga innecesaria.

El ciclo de vida del componente Activity es como el siguiente:

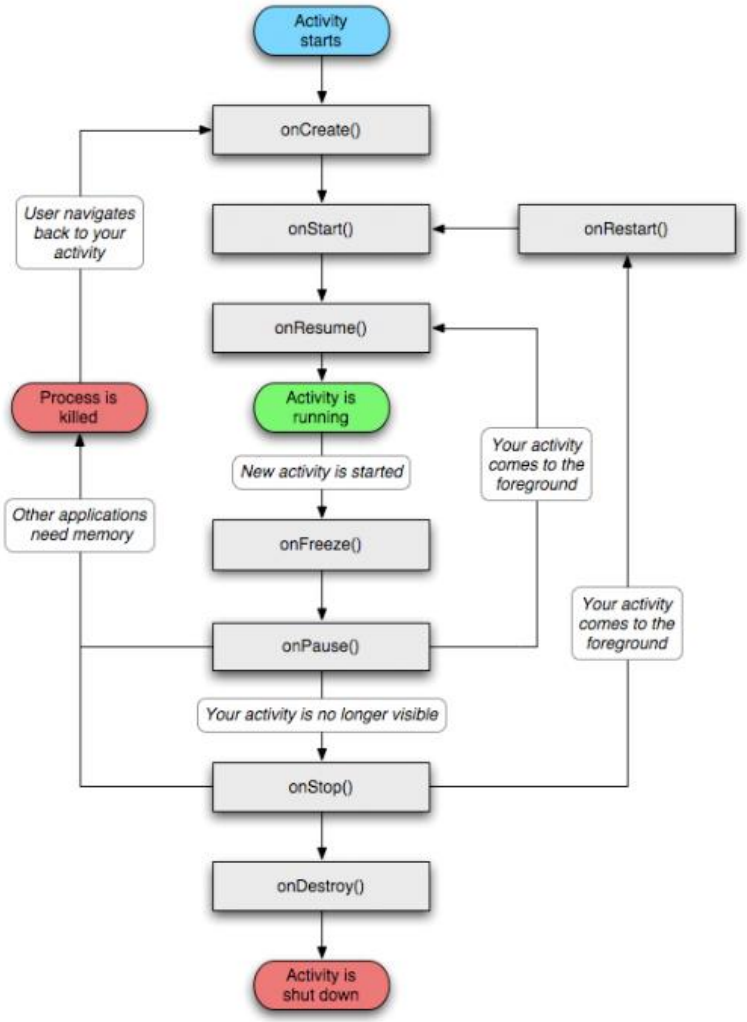

Ilustración 19 Ciclo de vida de una aplicacion Android

Como hemos podido observar onCreate y onDestroy se centran en el inicio o finalización de la aplicación. Después de ellos llegarían el onStart y el onStop, que delimitan los momentos en que la aplicación está visible o no (visible significa que está en segundo plano), y también están el onResume y el onPause que delimitan cuando la actividad está en el foco del usuario y cuando este puede interactuar con ella.

Visto esto en nuestra aplicación, los servicios que más nos harán consumir son el GPS y el sensor. Así pues los activaremos en el onResume() asegurándonos que solo están activados mientras el usuario interactúa con nuestra actividad como foco y los finalizaremos en el onPause(). De este modo evitaremos que la batería se acabe por el consumo del GPS de manera innecesaria.

CAPITULO 4. LAYAR

4.1. Introducción

Layar es una aplicación disponible para móviles con sistema operativo Android y para Iphone, que a partir de nuestro posicionamiento GPS, la brújula, la cámara y una conexión a internet nos permite mostrar por pantalla las capturas del mundo real, añadiendo una capa virtual con información y posicionamiento de unos puntos llamados puntos de interés (POI). Así pues podríamos decir que Layar es un buscador que hace uso de la Realidad Aumentada.

A nivel de usuario lo primero que se tiene que hacer cuando se conecta a Layar es elegir una capa (o layer en ingles) a la que conectarse. Antes de todo,¿qué es una capa? Una Capa es un grupo de POIs que pueden tener una temática en común o no; por ejemplo se puede hacer una capa sobre puntos de interés en el campus (cafeteria, parquing de alumnos, biblioteca …). Cada uno de estos lugares sería un POI, y se mostraría en la pantalla de Layar su posición respecto a la nuestra y información añadida o no, la información añadida puede llegar desde un teléfono de interés, un mail de contacto, enlaces a URL externas o imágenes 2d o 3d.

Una vez seleccionada la capa, la aplicación Layar obtiene la posición GPS del usuario y hace una petición de POI enviando al servidor de Layar los campos: Nombre de capa, posición GPS, rango para mostrar puntos y versión de Layar con la que se trabaja.

Cuando Layar recibe el mensaje, comprueba si en su base de datos existe esa capa y de ser así, en la base de datos (BD) de Layar ya tendrá asociada una URL de una página del autor que hace de proxy entre el servidor Layar y la BD del autor, reenvia la petición a dicha página y esta le devuelve los puntos que están en el rango tras consultar a la BD.

Una vez recibidos los puntos la aplicación se encarga de posicionarlos según la orientación del usuario y de printar por pantalla, si es necesario, las imágenes o los objetos 2-3d.

4.2. Creación de una capa para Layar

4.2.1. Registro de la capa

Para crear una capa en Layar lo primero que tenemos que hacer es darnos de alta como desarrollador de Layar, para ello rellenamos los datos en la página:

https://www.layar.com/accounts/register/

Mientras la cuenta se activa, lo que puede tardar unos días, podemos pasar a crear el hosting que necesitaremos para tener accesible nuestros POI (Apartado 4.2.2). Una vez aceptados, accedemos a www.layar.com e iniciaremos nuestra sesión. Una vez inicializada la sesión, en la parte derecha superior podremos acceder a “My layers” para empezar a crear capas.

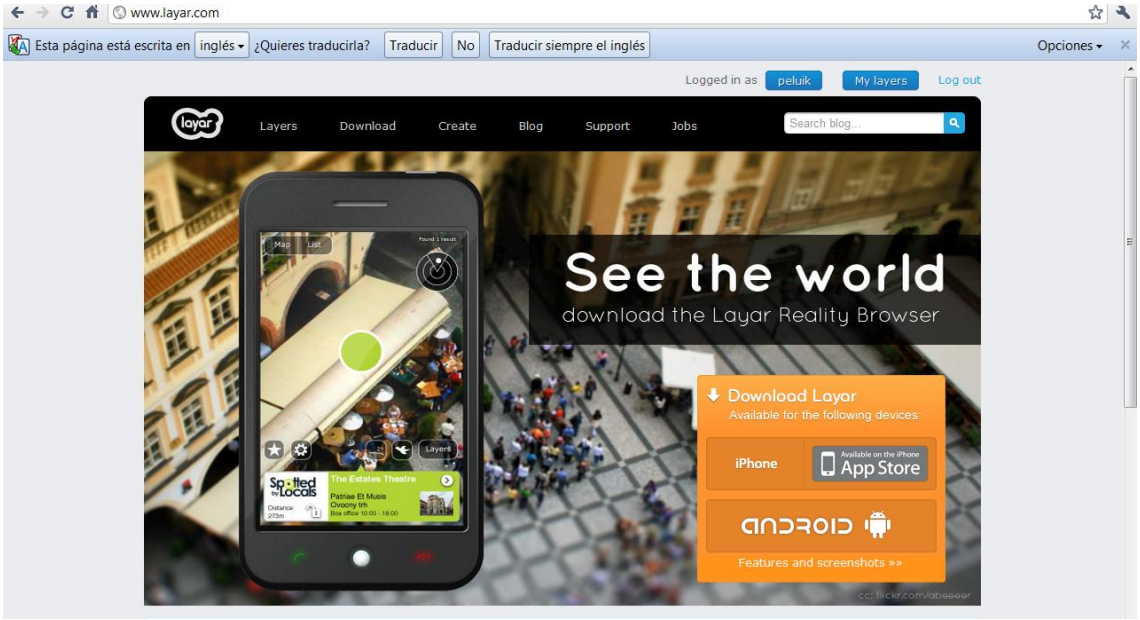

Ilustración 20 Pagina de Layar con usuario Logeado

En la página siguiente seleccionaremos “Create a layer”, y ya podremos rellenar los campos para crear la capa:

Campo	Función
Layer Name	El nombre de la capa que elijamos ponerle
Title	Título con el que queremos que aparezca
Publisher Name	Nuestro nombre o Nick de desarrollador
Layer type	2D o 2D-3D, la elección dependerá de si prevemos que nuestra capa será con objetos en 3D o no.
API endpoint URL	La dirección de la página que tiene acceso a nuestra BD, si en este momento aún el hosting preparado podemos poner cualquier URL aunque no funcione.
Short Descripcion	Breve descripción del contenido de la capa

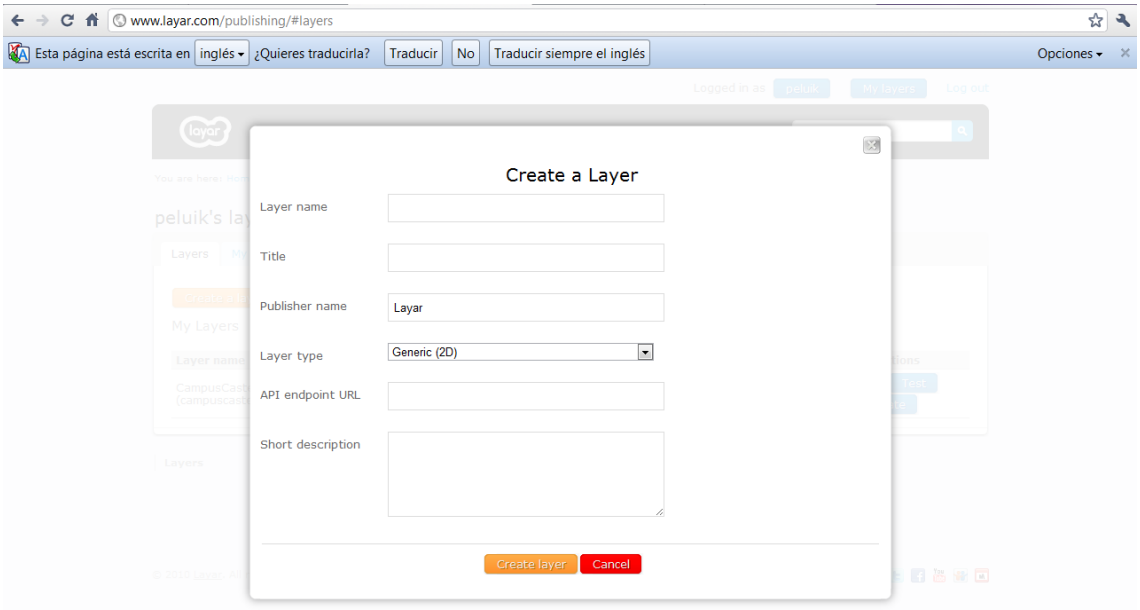

Ilustración 21 Pagina de creación de Layar

Si enviamos los datos nos pasa a una página en la cual podemos personalizar la capa, con campos como por ejemplo: los colores de interfaz del usuario en la aplicación o configurar la capa como de pago.

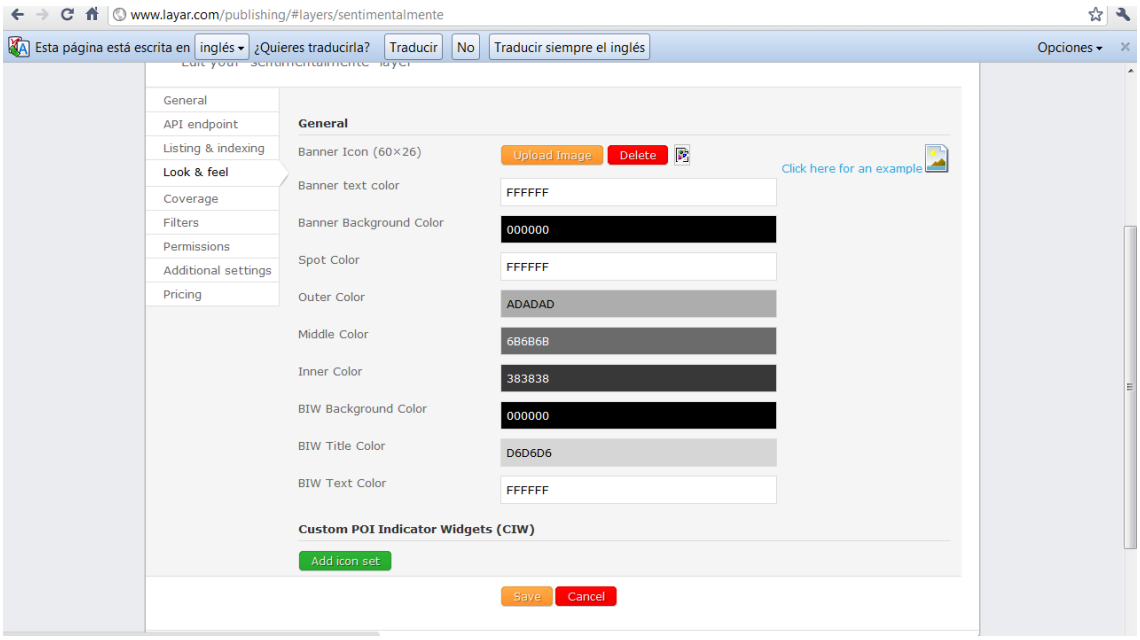

Ilustración 22 Pagina de configuración del BIW de Layar

4.2.2. Creación de un hosting de capas layar privado

Como hemos explicado al principio, para poder ofrecer el servicio de una capa necesitamos tener una página web que hará de proxy con las base de datos y

que será accesible desde la Internet. Para ello vamos a explicar cómo tener un hosting en nuestro ordenador con unos sencillos pasos mediante software libre.

Layar nos ofrece tanto una consulta para crear la BD como el código de la página que hará de proxy. Se pueden encontrar en los tutoriales oficiales de Layar:

http://layar.pbworks.com/w/page/7783228/FrontPage

Para crear el hosting utilizaremos XAMPP. XAMPP es un programa gratuito que nos instala Apache y MySQL de manera fácil y sencilla. Lo podemos descargar de:

http://www.apachefriends.org/download.php?xampp-win32-1.7.3.exe

Una vez descargado lo ejecutamos, lo instalaremos en algún directorio bien ubicado puesto que luego tendremos que acceder a él (en nuestro caso será **"c:\").**Una vez instalado lo ejecutaremos y podremos observar una ventana como la siguiente:

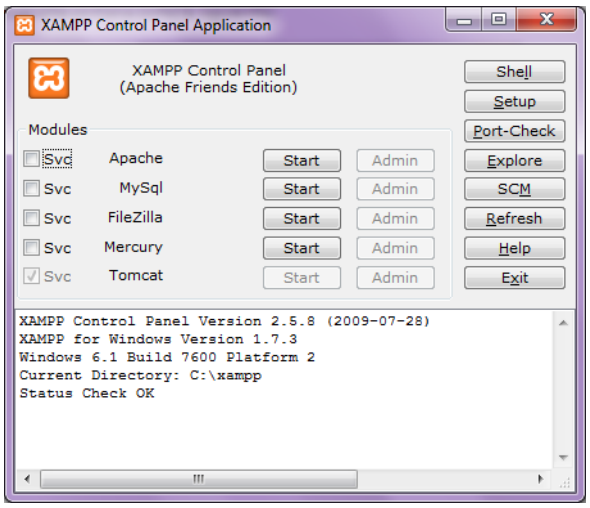

Ilustración 23 Ventana principal de XAMPP

Para iniciar el servicio clicaremos el Start de Apache y MySql que son los 2 servicios que necesitamos.

4.2.2.1. Configuración del a Base de Datos

Para configurar la BD lo primero que tendremos que hace es acceder al modo administrador. Para ello clicaremos en el Admin de MySql desde Xampp, la pagina web en el navegador que tengamos predeterminado.

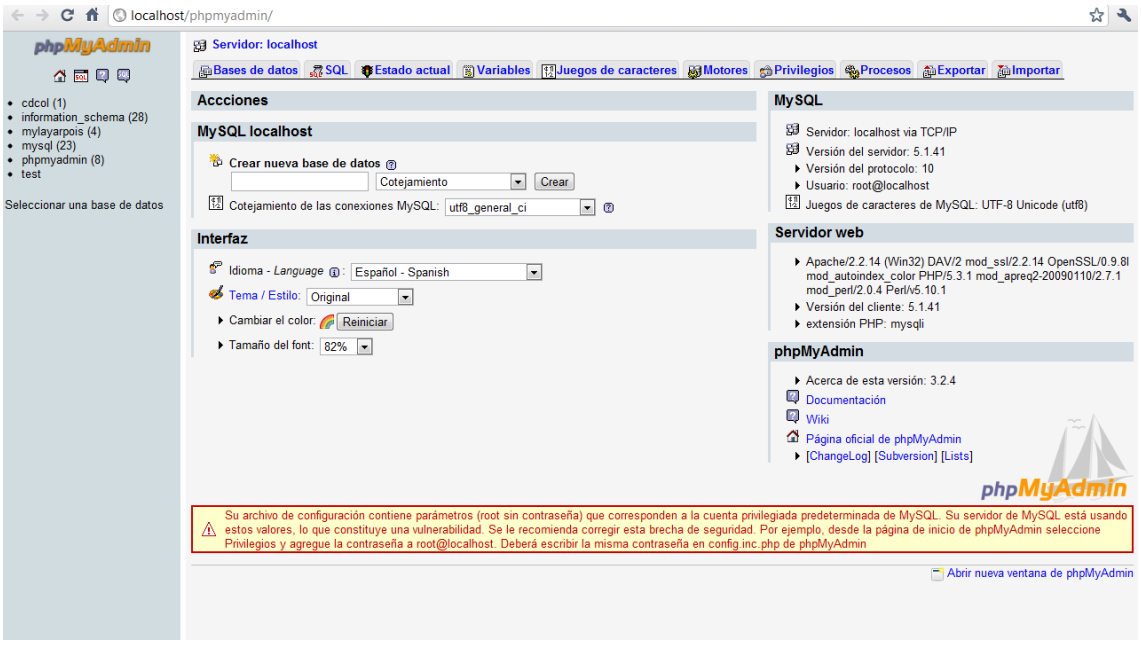

Ilustración 24 Pagina principal de MySQL

Lo primero será crear una BD. Para ello introduciremos en el recuadro donde pone “Crear nueva base de datos” un nombre, (en nuestro tutorial lo llamaremos “mylayarpois”) y le daremos a Crear. Una vez creada la BD se va a la pestaña SQL y se ejecuta la consulta que Layar nos facilita. Ya tendremos la BD creada.

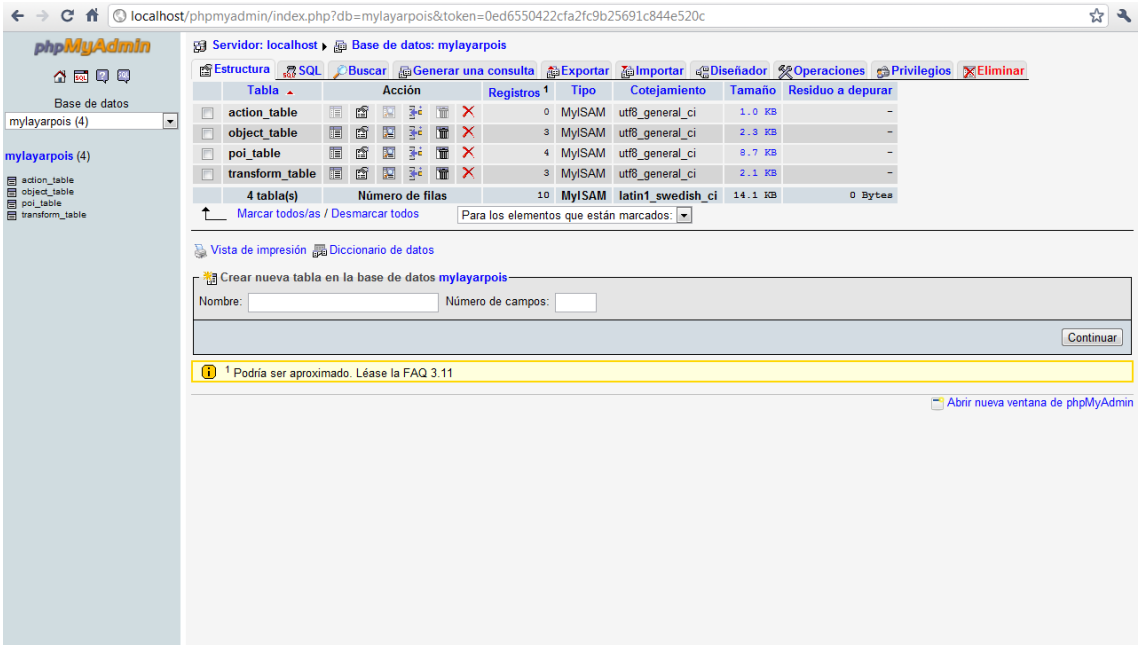

Ilustración 25 Muestra de la BD del hosting de capas

Antes de completar los campos de los POI es importante saber el significado de cada uno de esos puntos. En el Anexo 6.1 podemos ver tanto el diagrama de las bases de datos como el significado de cada campo.

Para introducir los POI, tendremos que ir completando las tablas. Cada POI tiene que estar definido en la *poi_table.sql*, si tiene acciones también tendrá que estar definido en la de *action_table.sql* y de tener un objeto para mostrar en 2d-3d tendrá que estar disponible en la *object_table.sql* y *transform_table.sql.*

Una vez seleccionada la tabla que queremos rellenar, iremos a la pestaña “Insertar” que nos aparece en phpMyAdmin y la seleccionaremos. Como podemos observar de esta manera simplemente tenemos que rellenar los valores de la tabla con los valores que deseemos o mediante consultas.

A continuación explicaremos cómo rellenar un POI para que muestre una imagen 3d, que tendrá que estar en formato l3d. La imagen se mostrará al tener el dispositivo a 100m, un icono en png para mostrar cuando no se muestre el l3d y un audio en formato mp3 que se escuchará cuando el usuario lo seleccione.

Tabla poi_table.sql

Campo	**Ejemplo**
id	1
attribution	Alejandro Andreu
title	Entrada Cafetería-EPSC
lat	41.2754540646
lon	1.9853067398
imageURL	http://147.83.113.237/imagenes/cafetería.png
Line4	Informacion línea 4 -- distance:%distance%
Line3	Información línea 3
Line2	Información línea 2
type	0
dimension	3
alt	0
relativeAlt	0
distance	0
inFocus	0
doNotIndex	0
showSamllBiw	1
showBiwOnClick	1

Tabla action_table.sql

Campo	**Ejemplo**
poID	1
label	audio
uri	http://147.83.113.237/imagenes/lacafetería.mp3

autoTriggerRange	0
autoTriggerOnly	0
ID	1
contentType	Audio/mpeg
method	GET
activityType	2
params	versión
closeBiw	0
showActivity	1
activityMessage	NULL

Tabla object_table.sql

Campo	Ejemplo
id	1
poiID	1
baseURL	http://147.83.113.237/imágenes/
full	taza.l3d
reduced	taza.l3d
icon	cafetería.png
size	40

Tabla transform_table.sql

Campo	Ejemplo
ID	1
poiID	1
rel	1
angle	0
scale	2

El formato l3d es característico de layar, en su página web nos facilitan una aplicación que cambia de formato obj/mtl a l3d y se puede descargar en la siguiente URL:

http://site.layar.com/downloads/Layar3DModelConverter.jnlp

4.2.2.2. Configuración Pagina-Proxy

La página web que se construye a partir del tutorial de layar hemos modificado las líneas:

```
$dbhost = "localhost";
$dbdata = "database_name";
$dbuser = "database_username";
```

$dbpass = "database_password";

Tendremos que sustituir "database_name" por el nombre de nuestra BD, en nuestro caso “mylayarpois”, y "database_username" y "database_password", con la configuración predeterminada de XAMPP son “root” y “” respectivamente.

Una vez modificado guardaremos la página en la dirección **“c:/xampp/htdocs”**.

4.2.3. Testeo de la capa

Para poder comprobar el buen funcionamiento de nuestra capa tendremos que asegurarnos de que tenemos Apache y MySQL funcionando. Si accedemos a la página de Layar, en “my layers”, podremos comprobar en “Edit” si la configuración es la deseada (sobretodo que la dirección de la página web sea la correcta).

Existen 2 maneras de testear la capa, desde un móvil con layar instalado o mediante una herramienta que nos ofrece Layar en su página web.

4.2.3.1. Testeo desde la página de Layar

En la página layar, si accedemos a “my layers”, donde está listada nuestra capa, podremos entrar en “Test” y pasaremos a una página donde aparece una pantallita con google maps, donde podremos arrastrar el icono de la persona hasta una lugar dentro del rango de nuestra capa y emulará la posición GPS y algunos parámetros necesarios como la versión o el rango. Al darle a “Load POIs”, si estás correctamente dentro del rango y la página es accesible, te saldrá un listado de los POI que están en el rango. Si es así ya sabes que tu capa funciona correctamente.

4.2.3.1. Testeo desde la aplicación Layar

Podemos hacer un test de nuestra capa con la aplicación Layar instalada en un móvil con Android sin necesidad de que la capa esté publicada, para ello nos identificaremos con nuestra cuenta entrando en la pestaña de “más” y seleccionando la opción de usuario.

Al introducir nuestra clave correctamente si después volvemos al menú principal y seleccionamos la opción de capas podremos ver una opción que es desarrollador.

Al seleccionarla aparecerán todas las capas que tengamos en fase de testeo y si están accesible las podremos probar.

CAPITULO 5. CONCLUSIONES

5.1. Conclusiones Generales

Como objetivo inicial nos marcamos desarrollar una aplicación que utilizase realidad aumentada y donde se mostrasen objetos 3d en puntos clave del Parc Mediterrani de la Tecnologia, y a pesar de la poca precisión de los GPS actuales, hemos conseguido una aplicación que cumple nuestras expectativas.

Incluso hemos creado una segunda aplicación donde se inyectan la posición GPS desde un GPX de una ruta pre marcada parcheando así el problema de la precisión GPS.

En este proyecto he adquirido una serie de conocimientos que no se habían dado a lo largo de la carrera, entre ellos el funcionamiento de la Realidad Aumentada o la programación para Android, a pesar de que está basado en Java, un lenguaje que ya conocía.

Siendo tecnologías tan recientes, no existe mucha información sobre algunos pasos del proyecto, como por ejemplo con Layar, del que no han aparecido los primeros tutoriales hasta verano del 2010.

Finalmente para concluir podría decir que estoy más que satisfecho con los resultados y que personalmente me gustaría poder desarrollar más aplicaciones tanto de Realidad Aumentada como para Android, los cuales, al parecer, van a jugar un papel importante en el futuro debido al abanico de posibilidades que nos ofrecen.

5.2. Ambientalización

Todo proyecto y actividad tiene un efecto en el medio ambiente, y si todos hacemos un esfuerzo y reconsideramos lo que estamos haciendo, es muy posible que podamos hacer algo para reducir nuestro impacto medioambiental.

En nuestro caso no construimos un objeto, por lo tanto no utilizamos materias primas, pero sí que podemos evitar el gasto innecesario de energía.

En este proyecto se trabaja en una aplicación para un teléfono móvil, en el que cada recurso que se utiliza aumenta el consumo de energía, como por ejemplo el GPS o los Sensores, por eso es importante controlar bien el ciclo de vida de las actividades cómo se explica en el apartado 3.6.

5.3. Posibles Futuras Mejoras

A pesar de que se han cumplido los objetivos iníciales del proyecto existe un gran número de posibles mejoras para nuestra aplicación.

- La primera de todas sería la mejora de la precisión del GPS que actualmente existe. En breve aparecerá el sistema de posicionamiento Galileo, que mejorará notoriamente la precisión que tiene actualmente el GPS.

- Otra mejora podría ser la interacción con los objetos 3D, y dotarlos de movimiento o incluso inteligencia, con lo que se podrían conseguir videojuegos muy reales.

- Una posible aplicación de lo que hemos visto sería el que cada usuario pudiese anunciar a los conocidos de una red social (como podría ser twitter o facebook) la posición actual del usuario y marcarla con un avatar 3D.

- Actualmente existen tecnologías en pleno desarrollo que reconocen objetos capturados con la cámara y cotejados con su base de datos. Esta tecnología nos podría evitar el hacer uso de la inserción de puntos y podría localizar nuestra situación a partir del entorno.

CAPITULO 6. REFERENCIAS DE INTERÉS

Paginas de interés de la Realidad Aumentada:

http://es.encydia.com/pt/Realidad_aumentada

http://asturel.blogcindario.com/2010/03/00811-realidad-aumentada.html

Tutoriales para la configuración del plugin de Android para Eclipse podemos visitar:

http://sites.google.com/site/swcuc3m/home/android/4-desarrollo-de-aplicaciones

Tutoriales para la instalación de aplicaciones Android en el emulador:

http://www.botskool.com/geeks/how-install-apk-files-android-windows-sdk-emulator

http://onsoftware.softonic.com/prueba-android-google

Tutoriales para el desarrollo de aplicaciones Android:

http://code.google.com/p/cursopudeandroid/downloads/list

http://developer.android.com/

Tutoriales sobre algunas aplicaciones que utilizan el GPS:

http://android.javielinux.com/locationgps.php

http://nelopauselli.blogspot.com/2010/08/android-integrando-google-maps-con-el.html

http://www.android-spa.com/viewtopic.php?t=1156

Tutoriales donde se explica cómo simular los sensores para el emulador:

http://code.google.com/p/openintents/wiki/SensorSimulator

http://www.botskool.com/geeks/how-use-sensor-simulator-android-sdk-emulator

http://javiercancela.com/2009/07/20/desarrollo-en-android-acelermetro-magnetmetro-y-sensores-de-orientacin-y-temperatura-en-el-htc-magic/

http://awesomebytes.com/category/android/

Tutoriales sobre Layar:

http://layar.pbworks.com/w/page/30832324/First%20Layar%20Tutorial%20-%20Create%20a%20simple%20layer

http://hectorlorenzo.blogspot.com/2010/06/com-crear-una-capa-per-layar-primera.html

https://groups.google.com/group/layar-developers?hl=es&pli=1

Paginas de interés sobre UTM:

http://www.ibm.com/developerworks/java/library/j-coordconvert/index.html

http://www.slideshare.net/profenatu/coordenadas-utm

Paginas de interés sobré XML:

http://www.latascadexela.es/2008/07/java-y-xml-sax.html

ANEXOS

6.1. Explicación campos de las BD

6.1.1. poi_table.sql

Campo	Tipo	Uso
id	varchar(255)	El numero con el que indexaremos nuestro POI
attribution	varchar(150)	El autor del punto, aparece en pantalla pero no es necesario
title	varchar(150)	Nombre del POI
lat	decimal(20,10)	Latitud del POI
lon	decimal(20,10)	Longitud del POI
imageURL	varchar(255)	URL de imagen que se podrá ver en el BIW del POI. La imagen de 100x75 pixels, tiene que ser accesible desde internet y no tener un tamaño superior a 100 kb
Line4	varchar(150)	Información que aparecerá en el BIW
Line3	varchar(150)	Información que aparecerá en el BIW
Line2	varchar(150)	Información que aparecerá en el BIW
type	int(11)	Determina el tipo de punto que se muestra en pantalla para un POI, pueden haber 3 posibles para cada capa
dimension	int(1)	Puede ser “1” si es un objeto que

		no queremos que muestre alguna imagen en la pantalla, “2” si queremos que el icono se muestre como un cartel en 3D o “3” si es un objeto en 3D
alt	int(10)	Altura a la que se mostrará el objeto, si es null se asume la misma altura que el usuario.
relativeAlt	int(10)	Altura a la que se mostrará el objeto respecto al usuario.
distance	decimal(20,10)	distancia del usuario al objetivo, este campo se autocompleta con la aplicación.
inFocus	tinyint(1)	Solo puede estar activo en un POI, si lo está el POI se quedara como objetivo fijo si se le enfoca.
doNotIndex	tinyint(1)	Se activa si no quieres indexar el objeto en el campo id
showSamllBiw	tinyint(1)	Si esta activo no se mostrara el BIW
showBiwOnClick	tinyint(1)	Si esta activo muestra una BIW más detallada

6.1.2. action_table.sql

Campo	Tipo	Uso
poID	varchar(255)	Id del POI en la object_table
label	varchar(30)	Tipo de recurso
uri	varchar(255)	Uri del recurso
autoTriggerRa	int(10)	Si lo rellenamos cuando estemos a esa

nge		distancia del POI se mostrará el recurso
autoTriggerOnly	tinyint(1)	Si esta activado solo se podrá ver el objeto cuando se cumpla el autoTriggerOnly
ID	int(10)	Id de la acción, se incrementa de 1 en 1
contentType	varchar(255)	Tipo de contendio entre los disponibles: "Text / html", "text / plain", "audio / mpeg", "audio/mp4", "Video/3gpp", "Video/mp4", "application / vnd.layar.internal", "application / vnd.layar.async"
method	enum('GET','POST')	GET por defecto
activityType	int(2)	Muestra el icono con el que se mostrará la actividad de los reflejados en esta lista: http://layar.pbworks.com/w/page/30763878/Activity-types-for-POI-actions
params	varchar(255)	Por defecto versión
closeBiw	tinyint(1)	Si está activado al acceder a la acción se cierra el BIW
showActivity	tinyint(1)	Si está activado al seleccionar un POI se muestra que tiene acciones, si no estarán ocultas.
activityMessage	varchar(255)	Mensaje que se muestra cuando está cargando la acción.

6.1.3. transform_table.sql

Campo	Tipo	Uso
ID	*int(10)*	*número de objeto*
poiID	*varchar(255)*	*Id del POI al que pertenece*
rel	booleano	Si el valor es true, el objeto siempre dará la cara al usuario.
angle	decimales	Angulo de rotación del objeto respecto a z.
scale	decimales	Se usa para escalar el objeto a diferentes tamaños.

6.1.4. object_table.sql

Campo	Tipo	Uso
id	int(10)	número de objeto

poiID	varchar(255)	Id del POI al que pertenece
baseURL	varchar(255)	dirección URL raíz de los objetos
full	varchar(255)	Representación del objeto cuando se está cerca (menos de 25 m)
reduced	varchar(255)	Representación del objeto cuando se está a media distancia (25 -100 m) 2D: Resolución menor a 100px 3D: tamaño inferior a unos 50 polígonos.
icon	varchar(255)	Representación del objeto cuando está a más de 100m, si las representaciones full o reduced fallan se muestra ésta.
size	float(15,5)	2D: tamaño de la imagen en metros 3D:Tamaño de la arista más pequeña del objeto en metros

6.1.5. Diagrama de la Base de datos