ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Aplicación Android para la identificación y llegada a un objetivo

AUTORA: María del Carmen Belizón Fernández

DIRECTOR: Juan Carlos Sánchez Aarnoutse

Dr. Simon Pomeroy

Septiembre / 2013

Autor	María del Carmen Belizón Fernández
E-mail del autor	mcarbelfer@gmail.com
Director	Juan Carlos Sánchez Aarnoutse, Dr. Simon Pomeroy
E-mail del director	juanc.sanchez@upct.es ; s.c.pomeroy@lboro.ac.uk
Codirector(es)	
Titulo del PFC	Aplicación Android para la identificación y llegada a un objetivo.
Descriptores	Tablet, Android, Sensores, Realidad Aumentada, OpenGL ES.

Resumen

Este proyecto tiene como objetivo principal el mostrar al usuario por pantalla la distancia a la que se encuentra un punto de interés escogido, perteneciente a una zona geográfica, con respecto a él y las coordenadas geográficas de dicho punto.

Para conseguir tal fin, esta aplicación calcula a través de un algoritmo la información deseada, obteniendo las variables necesarias para resolver las ecuaciones trigonométricas, de los sensores y la altura introducida manualmente por el usuario.

En el proyecto se hace uso de los sensores del dispositivo (acelerómetro, brújula y GPS), de la Realidad Aumentada (AR) para mostrar la imagen que proporciona la cámara y de la API OpenGL ES para diseñar los gráficos.

Titulación	Ingeniería Técnica de Telecomunicación, esp. Telemática
Intensificación	
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de presentación	Septiembre-2013

Agradecimientos

Primero, dar las gracias a Simon Pomeroy por darme la oportunidad de llevar a cabo este proyecto durante mi estancia en la Universidad de Loughborough y a Juan Carlos Sánchez Aarnoutse por dirigir mi proyecto en la Universidad Politécnica de Cartagena ayudándome en todo lo que he necesitado, siempre que ha sido necesario.

Agradecer a mis padres su gran paciencia conmigo cuando las cosas no salen como yo espero, su apoyo incondicional en todo momento y sobretodo destacar su gran esfuerzo y trabajo porque nunca nos falte de nada, por darnos a mi hermana y a mí siempre lo mejor, por hacernos felices y darnos la oportunidad de crearnos un gran futuro. Gracias por esas palabras de ánimo que me hacen creer en mí misma, en mis posibilidades y dejar de creer que hay cosas imposibles. Por haber dejado a un lado sus cosas y haberme ayudado en lo que haya hecho falta. Por haberme convertido en la persona que hoy día soy y porque gracias a ellos he llegado hasta aquí.

Gracias también por darme la maravillosa oportunidad de estudiar en el extranjero, por descubrir mundo y crecer como persona. Mis sueños se cumplen gracias a que vosotros formáis parte de mi vida y habéis hecho y hacéis todo lo posible siempre por mi hermana y por mí. Nunca podré agradeceros como os merecéis todo lo que me dais y todo lo que me habéis enseñado.

También dar las gracias a mi hermana por esas largas conversaciones, por estar a mi lado tanto en los buenos como en los malos momentos y por tener las palabras adecuadas en todo momento. Gracias por enseñarme a disfrutar, a ser feliz en la vida y no dar importancia a aquello que no lo merece. Simplemente gracias por ser mi hermana porque eres la mejor persona.

A mi familia, y en especial a mis abuelos por preocuparse, por confiar y por hacerme ver que soy capaz de conseguir todo aquello que me proponga.

Agradecérselo también a mis compañeros de carrera por haber compartido risas y nervios, por esos momentos de clase y estudio donde todo se hacía mucho más fácil. Gracias por compartir conmigo esta etapa y hacerla más bonita.

Índice

Índice de figuras

Capítulo 1. Introducción

1.1 Introducción

El proyecto se crea con el fin de familiarizarse con el sistema operativo Android [1], incorporándole Realidad Aumentada (RA) [2] y la API OpenGL ES [3].

Ha sido elegida la plataforma Android para el desarrollo de la aplicación por la multitud de ventajas que ofrece: se trata de un software libre y de código abierto, es decir, cualquiera puede estudiar, modificar y mejorar su diseño sin restricciones mediante el uso del código fuente. Asimismo, es posible desarrollar aplicaciones en ordenadores con sistema operativo Windows, existen plugins que permiten el desarrollo de aplicaciones en entorno de programación Eclipse y el lenguaje de programación utilizado es Java.

Además, Android es un sistema operativo que se ha expandido muy rápidamente llegando a convertirse en líder de ventas en muchos países y un modelo a seguir para los desarrolladores de tendencias y negocios de alto impacto, ya que existe un amplio abanico de dispositivos móviles que soportan tal sistema operativo como son las tablets, smartphones, portátiles o Google TV.

La Realidad Aumentada es implementada para superponer los datos que proporcionan los sensores: los datos del acelerómetro, brújula y GPS a la imagen del mundo real proporcionada por la cámara. Esta ofrece un gran número de nuevas posibilidades de interacción, haciendo que esté presente en muchas áreas, tales como la arquitectura, el entretenimiento, la educación, el arte, la medicina o las comunidades virtuales.

La API OpenGL ES, es incorporada al diseño del proyecto para realizar los gráficos en 3D, en particular, será empleada para el desarrollo y obtención de los ejes del sensor orientación.

A lo largo del documento, estos conceptos serán explicados con más detalle.

Con respecto a la aplicación, muestra a través de la cámara trasera del dispositivo una vista de un área geográfica (incluida en su GPS). A partir de esta vista, el usuario puede seleccionar en la pantalla de su dispositivo un lugar de esa zona. En ese momento, haciendo uso de un algoritmo, la aplicación devolverá la distancia que existe entre el punto seleccionado y el usuario y además, puede saber cuál es la latitud y longitud de dicho punto.

El proyecto hace uso del acelerómetro, brújula y GPS de una tablet con el fin de generar datos de la cámara trasera del dispositivo.

Por último, mencionar que la aplicación se puede utilizar para diversos fines en la vida diaria, tales como la localización de dónde está aparcado el coche, avisar a los servicios de emergencia dónde se ha producido un accidente o indicar a otra persona en qué lugar te encuentras.

1.2 Objetivos

Este trabajo ha sido creado con diversos objetivos.

Por un lado, atendiendo a un punto más teórico, el proyecto trata de explicar qué es el sistema operativo Android, la Realidad Aumentada y la API OpenGL ES y cómo estos dos últimos conceptos pueden trabajar e influir sobre el sistema operativo. Es decir, cómo es posible crear e implementar una realidad mixta (mundo real y realidad virtual) o cómo se pueden desarrollar e implementar gráficos 2D y 3D en dispositivos de la vida diaria como pueden ser los smartphones, ordenadores, tablets, etc. También se pretende mostrar qué función realiza cada uno de los sensores presentes en los dispositivos móviles actuales (acelerómetro, brújula, GPS).

Por otro lado ya más práctico, esta aplicación ofrece al usuario la oportunidad de conocer cierta información de interés (distancia, latitud y longitud de un punto seleccionado) de una manera sencilla y en muy poco tiempo, además de observar constantemente en pantalla los valores que representan cada uno de los sensores.

Se pretende además que mediante este proyecto, por un motivo u otro, el usuario pueda comunicarse con otras personas puesto que puede proporcionar cuál es su ubicación o la de un punto de interés o conocer una cierta distancia.

1.3 Estructura del proyecto

Este proyecto ha sido dividido en varios capítulos con el fin de ir desarrollándolo y explicándolo paso a paso desde su comienzo hasta alcanzar el desarrollo completo y obtener la aplicación final deseada.

La estructura que se ha seguido es la siguiente: en el ***capítulo 2,*** se explican con detalle cuáles han sido las tecnologías empleadas para el desarrollo de la aplicación, sus principales características y para qué han sido utilizadas en este caso particularmente. En el ***capítulo 3,*** se explican los algoritmos empleados para obtener, por un lado la distancia que existe entre el usuario y el objetivo seleccionado y por otro lado, las coordenadas geográficas (latitud y longitud) de dicho objetivo. En el ***capítulo 4,*** se realiza detalladamente toda la implementación paso por paso para desarrollar los sensores (acelerómetro, brújula y GPS), obtener la imagen de la cámara, sobreponer datos de los sensores a esta imagen, introducir el gráfico en el centro de la imagen ofrecida por la cámara e introducir los algoritmos explicados en el capitulo anterior para hacer los cálculos trigonométricos. En el ***capítulo 5,*** se enumeran los diversos modos de instalar la aplicación y se proporciona un manual de uso para el usuario. En el ***capítulo 6,*** se presentan las conclusiones generales de este trabajo y se proponen una serie de futuras mejoras para este proyecto.

Capítulo 2. Tecnologías empleadas

2.1 Android

Android es un sistema operativo móvil que se basa en una versión modificada de Linux. Esta plataforma está compuesta por el sistema operativo, Middleware y las aplicaciones claves del sistema. Por otra parte, se encuentra bajo la licencia Apache, es decir, se trata de software libre y de código abierto lo que significa que cualquier persona que quiera utilizar Android puede hacerlo descargando el código fuente completo de Android y también permite a los desarrolladores crear aplicaciones y aprovechar todas las ventajas que ofrecen los dispositivos Android [4].

Creado inicialmente por Android Inc. y más tarde en 2005 fue comprado por Google. En 2007 fue presentado por Open Handset Alliance (más de 50 empresas de diferentes campos) y Google. Esta última es la compañía más importante que lidera este sistema operativo.

Fue diseñado principalmente para dispositivos móviles con pantalla táctil tales como smartphones o tablets aunque hoy en día debido a su liderazgo, su uso se ha extendido llegando a encontrarse en ordenadores portátiles, Google TV, relojes de pulsera, electrodomésticos, etc.

Para introducir en el mercado y dar a conocer todas las aplicaciones desarrolladas para los dispositivos que implementan dicho sistema operativo, se creó el Android Market, actualmente conocido como Play Store, que es un programa informático de distribución de contenidos, el cual permite a sus usuarios navegar, comprar, descargar, instalar y puntuar aplicaciones realizadas por desarrolladores de todas partes del mundo.

Por último, la estructura básica de Android en general, es:

Linux kernel: Android ha sido desarrollado sobre la base de Linux kernel 2.6. El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura. El kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación *(*networking*)*, etc.

Librerías: es una capa que se sitúa justo encima del kernel. Proporciona diversas bibliotecas C / C + + que pueden ser usadas en Android por diversos componentes del sistema. Normalmente, es el fabricante el encargado de hacerlas e instalarlas en los dispositivos antes de la venta de estos. El principal objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia.

Android Runtime: se encuentra al mismo nivel que las librerías y está compuesto por la *máquina virtual Dalvik*, que es la máquina virtual para Android. Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute y *Core Libraries* que son librerías con las clases estándar de Java.

Framework de la aplicación: conjunto de herramientas de desarrollo de cualquier aplicación. Está formado por las clases y servicios que utilizan directamente las aplicaciones para realizar sus funciones.

Aplicaciones: es la última capa y en ella se incluyen todas las aplicaciones Java ya sean por defecto de Android o aquellas desarrolladas por el usuario.

A continuación, se muestra una figura que representa la estructura de Android comentada anteriormente.

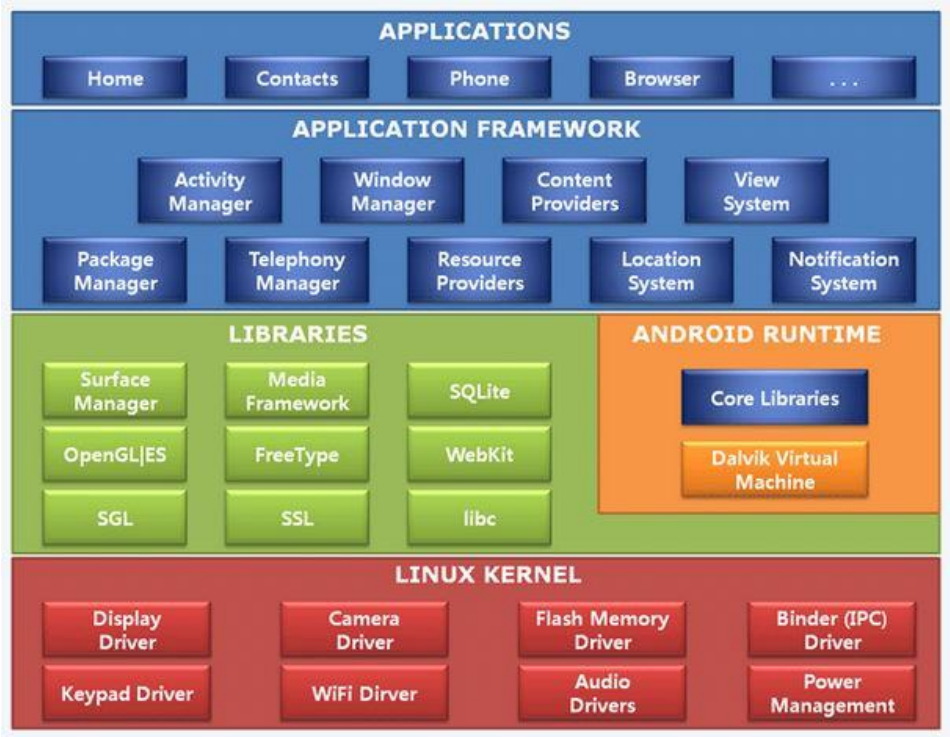


Figura 1. Estructura Android

2.2 Realidad Aumentada

La Realidad Aumentada (RA) [5] es un concepto empleado para definir una vista directa o indirecta de un medio físico del mundo real, cuyos elementos son combinados con elementos virtuales para crear una realidad mixta en tiempo real. Consiste en un conjunto de dispositivos que añaden información virtual a la información física existente. Esta es la principal diferencia con la realidad virtual, ya que esta no reemplaza la realidad física, sino que superpone datos virtuales al mundo real.

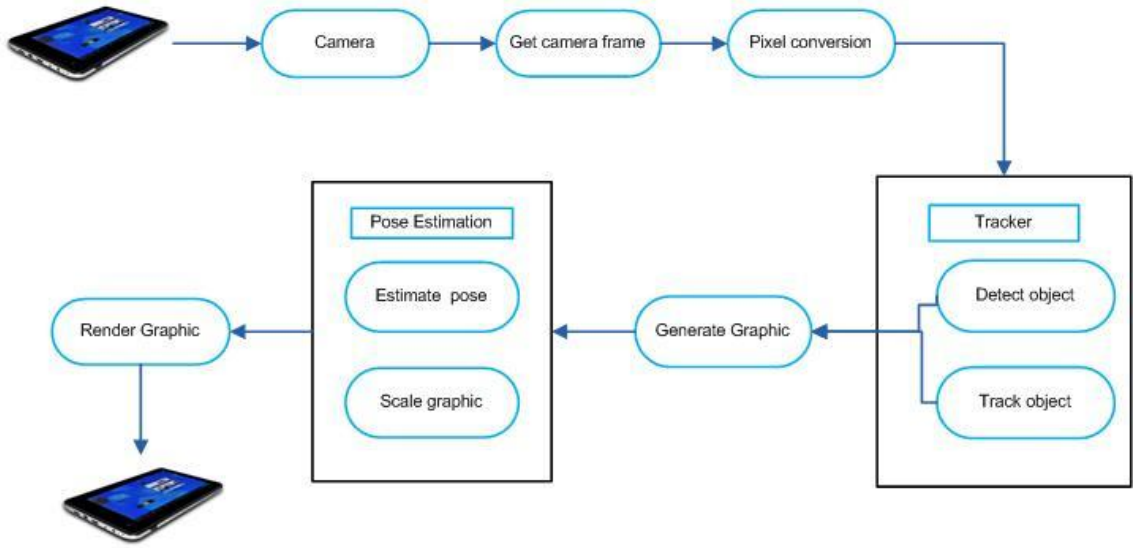


Figura2. Estructura Realidad Aumentada

Los cuatro componentes principales de la Realidad Aumentada [5] son:

Cámara: forma el 99% de la realidad en la Realidad Aumentada. Se encarga de capturar la imagen en tiempo real, del mundo real y mostrarla en la pantalla del dispositivo. Es posible sobreponer datos extraídos de diferentes fuentes de información a esa imagen.

Acelerómetro: se encarga de medir las fuerzas aplicadas al dispositivo a lo largo de los tres ejes direccionales: izquierda-derecha (lateral (X)), delante-atrás (longitudinal (Y)) y arriba-abajo (vertical (Z)). La fuerza de la gravedad influye siempre a la hora de medir las fuerzas. Se mide en m/s^2.

Eje X: mide la fuerza lateral. Los valores negativos indican movimiento hacia la izquierda y los valores positivos representan movimiento hacia la derecha.

Eje Y: mide la fuerza longitudinal. La lectura es positiva cuando el dispositivo es movido hacia su parte superior y una lectura negativa cuando el dispositivo es movido en la dirección opuesta.

Eje Z: este último eje, mide la fuerza hacia arriba y hacia abajo. Las lecturas son positivas cuando se realizan movimientos ascendentes y negativas cuando se realizan movimientos descendentes. Cuando el dispositivo está en reposo, muestra una lectura de $9.8m/s^2$ aproximadamente debido a la gravedad.

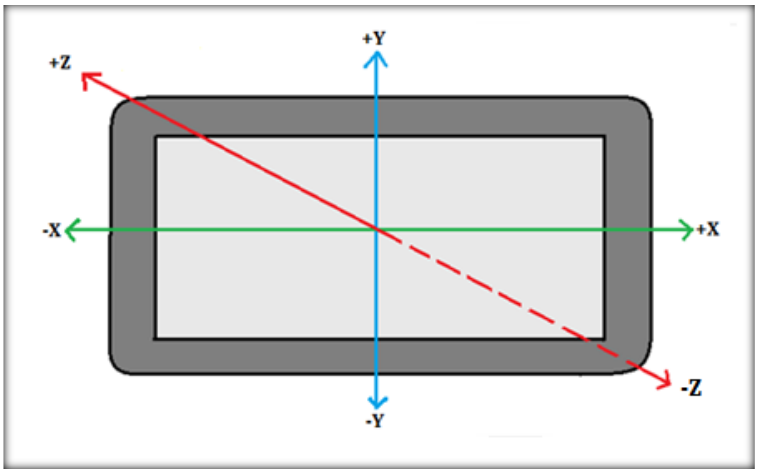


Figura3. Ejes acelerómetro

Brújula: el sensor de orientación permite monitorizar la posición de un dispositivo en relación con el marco de la Tierra de referencia (especialmente Norte magnético).

Consta de tres ejes:

Pitch (grados de rotación alrededor del eje X): este devuelve el ángulo de inclinación del dispositivo. El rango de valores está comprendido entre 180 grados y -180 grados.

Roll (grados de rotación alrededor del eje Y): este eje devuelve el ángulo de rotación del dispositivo. El valor será de 90 grados si la pantalla del dispositivo gira hacia la izquierda y -90 grados si la pantalla gira hacia la derecha.

Heading (grados de rotación alrededor del eje Z): devuelve el valor del ángulo azimuth del dispositivo, donde 0 grados o 360 es el Norte magnético, 90 grados es el Este, 180 grados el Sur y 270 el Oeste.

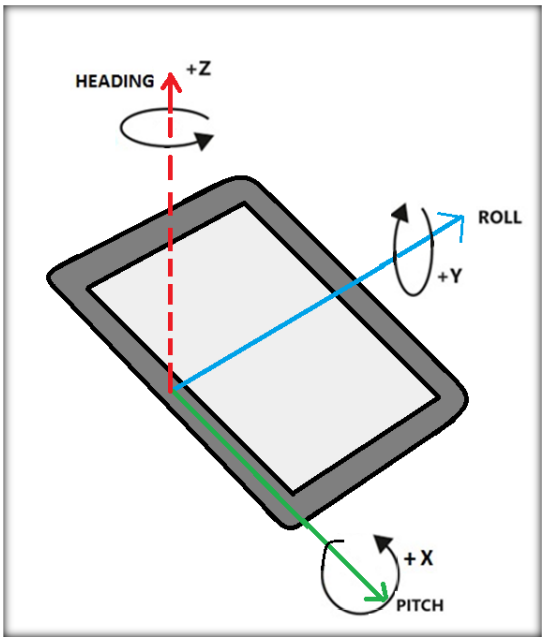


Figura4. Ejes brújula

Sistema de posicionamiento global (GPS): es un sistema de localización que puede dar una localización extremadamente precisa vía satélites. Es posible identificar un punto geográfico exacto de donde se encuentra un objeto o una persona.

El GPS trabaja a través de una red de 24 satélites que orbitan sobren la Tierra y es de libre acceso para cualquier persona que disponga de un receptor GPS.

Consta de dos componentes:

Latitud: es la distancia que existe entre un punto cualquiera y el Ecuador. Se mide de 0º a 90º desde el Ecuador hasta los Polos en dirección Norte o Sur. El Ecuador se toma como línea de referencia y le corresponde la latitud 0º. Todos los puntos situados en el mismo paralelo, tienen la misma latitud.

Longitud: es la distancia que existe entre un punto cualquiera y el Meridiano de Greenwich, situado en Inglaterra. Se mide de 0º a 180º en dirección Este u Oeste según la posición con respecto al Meridiano de Greenwich ya que este se toma como línea de referencia y le corresponde la longitud 0º.

2.3 OpenGL ES

OpenGL ES (OpenGL Embedded Systems) está basado en la API OpenGL pero diseñado para dispositivos integrados. OpenGL (Open Graphics Library) es una API multilenguaje y multiplataforma que permite escribir aplicaciones que produzcan gráficos en 2D y 3D.

Con respecto a la aplicación, hace uso de esta API para representar los tres ejes del sensor orientación, con el fin de focalizar el punto de interés en el origen de estos ejes y además, observar cómo el gráfico se mueve de acuerdo a este sensor.

Hay dos clases esenciales que permiten crear y manipular gráficos con OpenGL ES [1,6]: la clase *GLSurfaceView* es un View donde es posible dibujar y manipular objetos usando las llamadas de esta API. *GLSurfaceView.Renderer* es una interfaz, la cual define los métodos necesarios para dibujar gráficos en una *GLSurfaceView*. Además, es imprescindible que los siguientes métodos estén implementados:

onSurfaceCreated (GL10 gl, EGLConfig config): llamado cuando la superficie es creada o recreada. Usar este método para llevar a cabo acciones que solo ocurren una vez a lo largo del proceso.

onDrawFrame(GL10 gl): usado para dibujar el marco actual. Usar este método como punto primario de ejecución para dibujar y re-dibujar objetos gráficos de *GLSurfaceView*.

onSurfaceChanged(GL10 gl, int width, int height): usado cuando la superficie cambia el tamaño. El sistema llama a este método cuando la geometría de la clase *GLSurfaceView* cambia (incluyendo cambios en el tamaño o la orientación de la pantalla).

Capítulo 3. Algoritmos

Los algoritmos introducidos en el código de la aplicación son una de las partes más importantes del proyecto, pues mediante su uso se proporciona al usuario la información que desea, objetivo principal del proyecto.

La aplicación es capaz de calcular los datos de interés (distancia, latitud/longitud del objetivo seleccionado) usando los datos proporcionados por los diferentes sensores del dispositivo y aplicando una serie de ecuaciones trigonométricas que serán explicadas más adelante. Sólo se debe tener en cuenta que el punto que se desea marcar debe estar centrado en el origen de coordenadas de los ejes del gráfico que se ofrece centrado en la imagen de la cámara y representado mediante el uso de la API OpenGL ES.

3.1 Cálculo de la distancia

Es necesario saber la distancia a la que se encuentra el punto de interés seleccionado con respecto a la posición del usuario.

En primer lugar, para poder llevar a cabo este cálculo, se debe introducir la altura a la que se encuentra el dispositivo con respecto al suelo y en metros. Esta altura, es introducida manualmente por el usuario ya que la aplicación proporciona en su interfaz un cuadro de texto con este fin. Cuando se presiona en este cuadro, automáticamente aparece un teclado numérico, en el cual, el usuario debe introducir la altura y pulsar la tecla Enter al terminar.

Una vez que la altura ha sido introducida, se debe centrar el punto deseado, como se ha mencionado anteriormente, en el origen de coordenadas (centro de la pantalla), con el fin de averiguar el valor del ángulo de inclinación del dispositivo, empleado en la ecuación.

Pulsando el botón que se encuentra en la parte inferior derecha de la pantalla llamado "*Get the data!*", la aplicación muestra la distancia [7], tras haber efectuado la Ecuación 1 que se muestra a continuación.

$$d = h * \tan(\alpha) \qquad (1)$$

Donde: d es la distancia
h es la altura
α es el ángulo de inclinación del dispositivo (ángulo pitch)

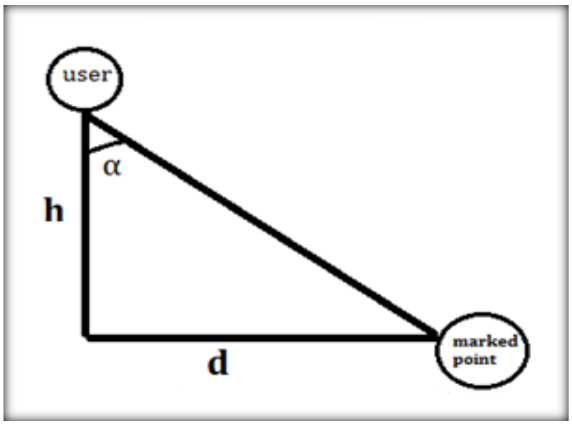


Figura 5. Cálculo distancia

3.2 Cálculo de la latitud y la longitud

Además, también es posible que el usuario conozca las coordenadas geográficas, latitud y longitud de dicho punto elegido [8] usando la Ecuación 2 y la Ecuación 3 que se muestran a continuación y los datos proporcionados por el sensor de localización.

Latitud del punto seleccionado:

$$\varphi 2 = \text{asin}(\sin(\varphi 1)*\cos(d/R) + \cos(\varphi 1)*\sin(d/R)*\cos(\theta)) \qquad (2)$$

Longitud del punto seleccionado:

$$\lambda 2 = \lambda 1 + \text{atan2}(\sin(\theta)*\sin(d/R)*\cos(\varphi 1), \cos(d/R) - \sin(\varphi 1)*\sin(\varphi 2)) \qquad (3)$$

Donde: $\varphi 1$ es la latitud del usuario, $\lambda 1$ es la longitud del usuario, θ es el rumbo (bearing), d es la distancia recorrida y calculada anteriormente, R es el radio de la Tierra, el cual equivale a 637100 metros y d/R es la distancia angular.

"Bearing" de un punto es el número de grados en el ángulo medido en sentido horario desde la línea norte a la línea que une el centro de la brújula con el punto. Se utiliza para representar la dirección de un punto con respecto a otro.

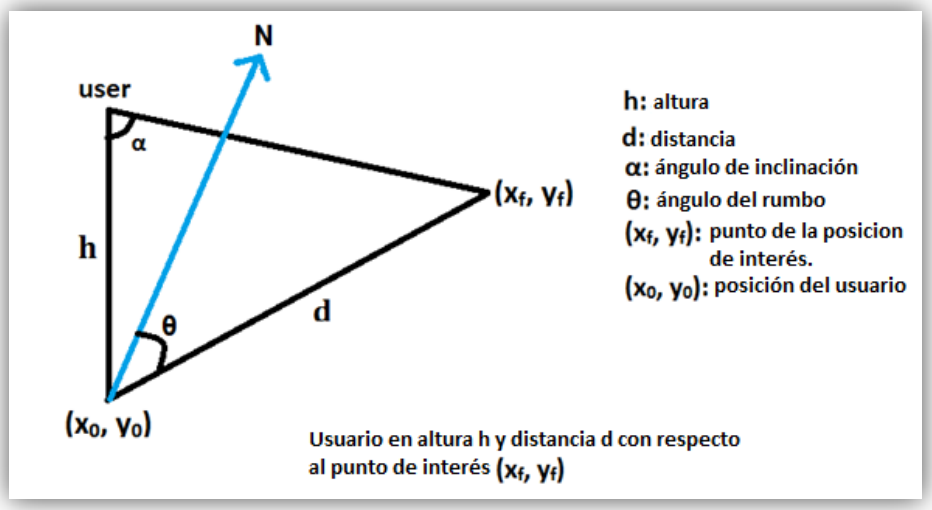


Figura 6. Cálculo coordenadas geográficas

Un ejemplo de cómo estos algoritmos son usados en el proyecto está explicado en el [Apéndice 1].

Capítulo 4. Desarrollo e implementación

En este apartado se explican detalladamente los pasos más importantes realizados para desarrollar e implementar cada uno de los componentes que forman la aplicación hasta obtener el proyecto final y lo que este representa.

4.1 Partes fundamentales del proyecto

Para poder comenzar a trabajar es necesario en primer lugar, si no se tiene ya, instalar el programa Eclipse http://www.eclipse.org/downloads/ y todas las herramientas necesarias para poder trabajar con Android pero usando el entorno Eclipse.

Una vez realizado todo el proceso anterior, el siguiente paso es crear un nuevo proyecto en el entorno de trabajo Eclipse. Para ello, se debe seleccionar del menú de herramientas *File- New- Android Application Project* e ir completando los campos que aparecen en cada una de las distintas ventanas emergentes. Al terminar, presionar el botón de *Finish* y aparecerá a la izquierda de la pantalla el nuevo proyecto con el nombre que se le haya dado y todas sus carpetas y archivos correspondientes. Llegado este punto, ya se puede empezar a trabajar.

Además, se tiene que tener presente que a parte de la clase principal de la cual consta el proyecto, hay tres archivos importantes dentro de este que son: *AndroidManifest.xml*, *main.xml* cuya ubicación es res/layout/main.xml y *strings.xml* que se encuentra en la carpeta res/values/strings.xml.

AndroidManifest.xml: este archivo se encuentra en la raíz de directorios de cualquier proyecto Android y contiene todos los componentes que aparecen dentro de la aplicación (actividades, servicios, permisos, etc). Sus partes principales son:

<manifest>: dentro de este tag se puede encontrar el atributo que especifica el número de versión de desarrollo del programa, el número de versión de nuestro programa y por último el paquete de nuestro programa y el cual se utiliza para referenciar nuestra aplicación en el Play Store y en los dispositivos móviles.

<application>: contiene todas las Activities, Services, Providers, Receivers y bibliotecas usados en la aplicación.

<activity>: contiene los atributos y los distintos Intents que utiliza una pantalla para comunicarse. Se tendrá un Activity por cada pantalla que tenga la aplicación.

<supports-screens>: describe las pantallas soportadas por la aplicación.

<uses-permissions>: este tag contiene los permisos de los que precisa la aplicación para que se pueda ejecutar y deberán ser especificados por el usuario antes de instalarla.

<uses-sdk>: contiene las distintas versiones de Android que va a utilizar la aplicación. Determina tanto las versiones que va a correr como la versión que fue utilizada para realizar las pruebas.

main.xml: define el diseño de la interfaz gráfica, es decir, la vista que se utiliza en la aplicación y la pantalla que ve el usuario cuando arranca la aplicación.

En este archivo se definen los elementos a mostrar como botones, texto, imágenes, etc y es el programador el encargado de diseñarlo.

strings.xml: en este archivo se crean todas las constantes de cadenas de caracteres a incluir en el programa (TextView, Button, etc). La idea principal de este archivo es tener todos los mensajes que se muestran a lo largo de la aplicación en un mismo archivo agrupados.

4.2 Crear el icono de la aplicación

Eclipse y la plataforma Android permiten elegir el icono que tendrá la aplicación y por tanto que aparecerá en cualquier dispositivo cuando esta sea instalada.

Este proceso se puede realizar de dos formas:

Por un lado, se puede crear un icono propio o seleccionar una imagen de Internet, guardando en cualquiera de los dos casos la imagen con nombre y formato ic_launcher.PNG (usado por defecto) e incorporándola al proyecto cuando se esté creando y aparezca la ventana con el nombre *Configure Launcher Icon*.

Por otro lado, se pueden elegir iconos por defecto proporcionados ya por Eclipse y pueden ser tanto texto como imagen.

En el caso de este proyecto, se ha creado una imagen propia para la aplicación y la cual se muestra a continuación.

Figura 7. Icono aplicación

4.3 Desarrollar los sensores

Como ya se ha explicado anteriormente, los sensores utilizados en este proyecto han sido el acelerómetro, la brújula y el GPS [4].

En primer lugar, para poder trabajar con los sensores de una manera eficaz, se han tenido que importar las librerías de Android necesarias para poder acceder a ellos y hacer uso de todos los métodos, llamadas y variables que los sensores necesitan para obtener y representar la información que van a proporcionar. Algunos ejemplos de estas librerías son:

```
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.widget.TextView;
```

También se debe tener en cuenta si el sensor requiere o no de permisos para su uso y obtención de datos. Si fuese necesario, se deberán conceder en el archivo AndroidManifest.xml para que la aplicación pueda ser ejecutada.

La clase *SensorEventListener* debe ser implementada y contiene los dos métodos fundamentales en los sensores, de los cuales solo uno de ellos es usado.

onSensorChanged (SensorEvent event) es usado para recibir notificaciones del *SensorManager* (clase que permite acceder a los sensores de un dispositivo Android) cuando los valores del sensor han cambiado y extraer la información de esos cambios que se han producido.

```
SensorManager sensorManager;
sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
```

Finalmente, se crean varias etiquetas TextView, declaradas previamente en el archivo explicado *main.xml*, para poder mostrar en pantalla el nombre de las variables y los valores que proporcionan cada uno de los sensores.

4.3.1 Acelerómetro

El acelerómetro es el sensor que se encarga de medir y representar las fuerzas a lo largo de los tres ejes direccionales. Con el fin de obtener los valores de dichos ejes del acelerómetro [5], es necesario crear primeramente tres variables, las cuales almacenen los datos representativos de cada eje y una constante que describa un sensor tipo acelerómetro para poder empezar a trabajar con él, como se muestra en el siguiente fragmento de código:

```
float xAxis;
float yAxis;
float zAxis;

int accelerometerSensor;
accelerometerSensor = Sensor.TYPE_ACCELEROMETER;
```

Una vez creada la constante que hace referencia al sensor, se debe registrar para poder recibir notificaciones de este más tarde.

```
sensorManager.registerListener(sensorEventListener, sensorManager.getDefaultSensor
(accelerometerSensor), SensorManager.SENSOR_DELAY_NORMAL);
```

El método siguiente es el encargado de obtener los valores del acelerómetro cada vez que se produce un cambio en alguno de ellos. En primer lugar, accede a los ejes del acelerómetro para obtener la información de cada uno de ellos y después se almacenan esos valores.

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

    public void onSensorChanged(SensorEvent sensorEvent) {

                                        .
                                        .
                                        .
                                        .


    else if (sensorEvent.sensor.getType() ==Sensor.TYPE_ACCELEROMETER)  {
        xAxis = sensorEvent.values[0];
        yAxis = sensorEvent.values[1];
        zAxis = sensorEvent.values[2];

        xAxisValue.setText(String.valueOf(xAxis));
        yAxisValue.setText(String.valueOf(yAxis));
        zAxisValue.setText(String.valueOf(zAxis));

    }


}
```

Cuando se ha accedido al sensor acelerómetro y se han obtenido los valores de cada uno de los ejes, se desean mostrar por pantalla los resultados, por lo que se deben crear seis variables del tipo TextView (mostrar texto por pantalla al usuario) que mostrarán el nombre de las variables y sus respectivos valores. En este caso, *eje X, eje Y y eje Z* y el valor que se obtiene de cada uno de ellos. A continuación, haciendo uso de findViewById() se enlaza la interfaz de usuario de la aplicación, con variables del código. Estas variables contienen, las tres primeras, texto y las tres siguientes los valores numéricos de cada eje.

Por último se establece el color con el que se quiere que aparezcan estas etiquetas por pantalla, en este caso, se ha elegido el color amarillo.

```
TextView xAxisLabel;
TextView yAxisLabel;
TextView zAxisLabel;

TextView xAxisValue;
TextView yAxisValue;
TextView zAxisValue;

 xAxisLabel= (TextView) findViewById(R.id.xAxisLabel);
 xAxisLabel.setTextColor(Color.YELLOW);
 yAxisLabel= (TextView) findViewById(R.id.yAxisLabel);
 yAxisLabel.setTextColor(Color.YELLOW);
 zAxisLabel= (TextView) findViewById(R.id.zAxisLabel);
 zAxisLabel.setTextColor(Color.YELLOW);
 xAxisValue = (TextView) findViewById(R.id.xAxisValue);
 xAxisValue.setTextColor(Color.YELLOW);
 yAxisValue = (TextView) findViewById(R.id.yAxisValue);
 yAxisValue.setTextColor(Color.YELLOW);
 zAxisValue = (TextView) findViewById(R.id.zAxisValue);
 zAxisValue.setTextColor(Color.YELLOW);
```

4.3.2 Brújula

La brújula proporciona información acerca de la orientación que experimenta el dispositivo en todo momento.

Los pasos seguidos para el desarrollo y obtención de los datos de cada uno de los ejes de la brújula son similares a los explicados anteriormente para el sensor de tipo acelerómetro. Solamente es necesario hacer unos pequeños cambios en cuanto al tipo de sensor con el que se quiere trabajar y del cual se quiere obtener la información y las variables que se quieren conocer.

Se deben declarar tres variables, las cuales almacenan los valores obtenidos de cada eje del sensor y una constante que describa un sensor de tipo orientación para poder empezar a trabajar con este, como se muestra en el siguiente fragmento de código:

```
float headingAngle;
float pitchAngle;
float rollAngle;

int orientationSensor;
orientationSensor = Sensor.TYPE_ORIENTATION;
```

A continuación, se debe registrar para poder obtener información cada vez que se produzca un cambio de valor en cualquiera de los tres ejes.

```
sensorManager.registerListener(sensorEventListener, sensorManager.getDefaultSensor(orientationSensor),
SensorManager.SENSOR_DELAY_NORMAL);
```

Como ocurría en el acelerómetro, el método encargado de obtener los valores de la brújula cada vez que se recibe una notificación de cambio es el método que se muestra a continuación. Una vez que se reconozca que el sensor es de tipo orientación, se accede a los valores que estos ejes contienen y después, se almacenan para su posterior uso y muestra por pantalla.

```
final SensorEventListener sensorEventListener = new SensorEventListener() {
    public void onSensorChanged(SensorEvent sensorEvent) {
        if (sensorEvent.sensor.getType() == Sensor.TYPE_ORIENTATION){
            headingAngle = sensorEvent.values[0];
            pitchAngle = sensorEvent.values[1];
            rollAngle = sensorEvent.values[2];

            headingValue.setText(String.valueOf(headingAngle));
            pitchValue.setText(String.valueOf(pitchAngle));
            rollValue.setText(String.valueOf(rollAngle));

        }
```

.
.
.
.

Finalmente, cuando se han obtenido los valores de cada uno de los ejes, se desean mostrar por pantalla los resultados, por lo que se deben crear seis variables de tipo TextView que mostrarán el nombre de las variables y sus respectivos valores. En este caso,

Heading, Pitch y Roll y el valor que se obtiene para cada uno de ellos. A continuación, haciendo uso de findViewById() se enlaza la interfaz de usuario de la aplicación, con variables del código. Estas variables contienen, las tres primeras texto y las otras tres los valores numéricos de cada eje.

Por último, se establece el color con el que se quiere que aparezcan estas etiquetas en la pantalla de la aplicación, en este caso, se ha elegido el color amarillo.

```
TextView headingLabel;
TextView pitchLabel;
TextView rollLabel;

TextView headingValue;
TextView pitchValue;
TextView rollValue;


headingLabel= (TextView) findViewById(R.id.headingLabel);
headingLabel.setTextColor(Color.YELLOW);
pitchLabel = (TextView) findViewById(R.id.pitchLabel);
pitchLabel.setTextColor(Color.YELLOW);
rollLabel = (TextView) findViewById(R.id.rollLabel);
rollLabel.setTextColor(Color.YELLOW);
headingValue= (TextView) findViewById(R.id.headingValue);
headingValue.setTextColor(Color.YELLOW);
pitchValue = (TextView) findViewById(R.id.pitchValue);
pitchValue.setTextColor(Color.YELLOW);
rollValue = (TextView) findViewById(R.id.rollValue);
rollValue.setTextColor(Color.YELLOW);
```

4.3.3 Posición GPS

El GPS permite obtener la localización de un punto geográfico exacto.

Llegados a este punto, aparece un nuevo elemento que hasta ahora no había sido utilizado en el desarrollo del proyecto y se trata de los permisos de Android. Este concepto ha sido explicado anteriormente en el archivo *AndroidManifest.xml* y para utilizar el GPS en una aplicación Android [5] es necesario conceder ciertos permisos en dicho archivo, de la siguiente manera:

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

Algunas de las librerías que deben ser importadas en el proyecto para el correcto uso de este sensor, son:

```
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
```

Como se ha hecho para los dos sensores anteriores, lo primero que se debe declarar antes de comenzar a trabajar son las variables de las que se va a obtener la información y necesarias para poder hacer buen uso de los métodos de los cuales dispone el GPS.

```
double latitude;
double longitude;
double altitude;
double bearing;
```

Se implementa la clase *LocationManager* para activar y acceder a los servicios de localización del GPS.

```
LocationManager locationManager;
locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
```

El siguiente método es el encargado de solicitar y registrar las actualizaciones que se producen en la localización.

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 2000,2, locationListener);
```

El fragmento de código expuesto más abajo, muestra el método que es llamado cuando se detecta una nueva posición del usuario. El método en primer lugar se encarga de obtener los valores de latitud, longitud y bearing y a continuación, los almacena para su posterior uso.

La condición if que se presenta, informa del rumbo del usuario siempre y cuando se obtenga una localización del usuario, es decir, si por algún motivo no se pueden obtener las coordenadas geográficas del usuario, no será posible conocer el rumbo.

```
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        locationUser = location;
        latitude = location.getLatitude();
        longitude = location.getLongitude();
        altitude = location.getAltitude();

        latitudeValue.setText(String.valueOf(latitude));
        longitudeValue.setText(String.valueOf(longitude));
        altitudeValue.setText(String.valueOf(altitude));

            if(locationUser != null) {
               bearing = locationUser.bearingTo(setLoc);
               bearingValue.setText(String.valueOf(bearing));
            }

    }

};
```

Por último, cuando se han obtenido los valores, se desean mostrar por pantalla los resultados, por lo que se deben crear ocho variables de tipo TextView que mostrarán al usuario el nombre de las variables y sus respectivos valores. En este caso, *Latitud*, *Longitud*, *Altitud* y *Bearing* y el valor que se obtiene de cada uno de ellos.
A continuación, haciendo uso de findViewById() se enlaza la interfaz de usuario de la aplicación, con variables del código. Estas variables contienen, las cuatro primeras texto y las cuatro siguientes sus respectivos valores numéricos.

El color con el que se quiere que aparezcan estas variables por pantalla es el color amarillo.

```
TextView latitudeLabel;
TextView longitudeLabel;
TextView altitudeLabel;
TextView bearingLabel;

TextView altitudeValue;
TextView latitudeValue;
TextView longitudeValue;
TextView bearingValue;

latitudeLabel= (TextView) findViewById(R.id.latitudeLabel);
latitudeLabel.setTextColor(Color.YELLOW);
longitudeLabel= (TextView) findViewById(R.id.longitudeLabel);
longitudeLabel.setTextColor(Color.YELLOW);
altitudeLabel= (TextView) findViewById(R.id.altitudeLabel);
altitudeLabel.setTextColor(Color.YELLOW);
bearingLabel= (TextView) findViewById(R.id.bearingLabel);
bearingLabel.setTextColor(Color.YELLOW);
altitudeValue = (TextView) findViewById(R.id.altitudeValue);
altitudeValue.setTextColor(Color.YELLOW);
longitudeValue = (TextView) findViewById(R.id.longitudeValue);
longitudeValue.setTextColor(Color.YELLOW);
latitudeValue = (TextView) findViewById(R.id.latitudeValue);
latitudeValue.setTextColor(Color.YELLOW);
bearingValue = (TextView) findViewById(R.id.bearingValue);
bearingValue.setTextColor(Color.YELLOW);
```

4.4 Obtener la imagen de la cámara

Al igual que ocurre con el GPS, para poder hacer uso de la cámara en una aplicación Android [5] es necesario conceder los permisos que esta requiere en el archivo *AndroidManifest.xml* de la siguiente manera:

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

Además, la cámara también necesita informar a cualquier entidad externa del conjunto de hardware y software del que la aplicación depende y para ello se debe incorporar al archivo *AndroidManifest.xml*:

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

Una vez establecidos los permisos y las características que requiere la cámara, se puede empezar a desarrollar en la clase principal del proyecto el código necesario para tener acceso a la imagen del mundo real.

Se necesita un objeto para poder almacenar una vista de lo que la cámara ofrece y por ello se crea uno de tipo *SurfaceView*. Más adelante, se crea un objeto de la clase *Camera* ya que dicha clase es la encargada de tomar fotos cuando se realiza una aplicación de la cámara y la última línea de código enlaza la interfaz de usuario de la aplicación, con una variable del código.

```
SurfaceView cameraPreview;
Camera camera;
```

```
cameraPreview = (SurfaceView)findViewById(R.id.cameraPreview);
```

La clase *SurfaceHolder* es el contenedor que proporciona acceso a la superficie.

Las siguientes líneas de código sirven para indicarle a SurfaceHolder que notifique los cambios de la superficie a la instancia. La última línea, especifica el tipo de superficie a usar, en este caso, se trata de una superficie que no utiliza el buffer de información por lo que el streaming de video es más eficiente y fluido.

```
previewHolder = cameraPreview.getHolder();
previewHolder.addCallback(surfaceCallback);
previewHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
```

Para abrir la cámara la función empleada es:

```
camera=Camera.open();
```

Una vez abierta la cámara, se establecen unas condiciones de manera que siempre se obtenga la mejor vista posible.

```
private Camera.Size getBestPreviewSize(int width, int height,Camera.Parameters parameters) {
    Camera.Size result=null;

    for (Camera.Size size : parameters.getSupportedPreviewSizes()) {
        if (size.width<=width && size.height<=height) {
            if (result==null) {
                result=size;
            }

            else {
                int resultArea=result.width*result.height;
                int newArea=size.width*size.height;
                    if (newArea>resultArea) {
                        result=size;
                    }
            }
        }
    }

    return(result);
}
```

La interfaz definida a continuación, recibe información acerca de los cambios que se producen en la superficie. Se hace una llamada al método surfaceCreated (SurfaceHolder holder) pues es llamado siempre que se crea por primera vez una superficie y especifica a la cámara dónde dibujar la vista.

```
SurfaceHolder.Callback surfaceCallback=new SurfaceHolder.Callback() {
    public void surfaceCreated(SurfaceHolder holder) {

                                        .
                                        .
                                        .
                                        .

    }
};
```

Se detiene la captura de vistas de la cámara:

```
camera.stopPreview();
```

La siguiente función libera la cámara cuando ya no se hace uso de ella para que pueda ser utilizada por otras aplicaciones si es necesario.

```
camera.release();
```

Debido a que el sistema Android puede ser utilizado en multitud de dispositivos hardware, es imposible, establecer correctamente un tamaño determinado para la pantalla donde se realizan las vistas previas por lo que, cuando se producen cambios en la superficie, se obtiene el tamaño de SurfaceView y se pasa a un objeto tipo camera.Paramenters. Con estos parámetros se actualiza la vista previa de la superficie de la cámara y tras realizar un startPreview es posible comenzar a tomar fotos. Cada vez que se quiera comenzar a capturar y dibujar vistas previas, es necesario llamar a este método.

```
public void surfaceChanged(SurfaceHolder holder, int format, int width,int height) {
    Camera.Parameters parameters=camera.getParameters();
    Camera.Size size=getBestPreviewSize(width, height, parameters);

if (size!=null) {
        parameters.setPreviewSize(size.width,size.height);
        camera.setParameters(parameters);
        camera.startPreview();
        inPreview=true;
    }

}
```

4.5 Unir los sensores a la imagen de la cámara

En este punto, cuando todos los sensores han sido desarrollados y probado su funcionamiento, entra en juego la Realidad Aumentada ya que cuando los componentes están unidos, lo que se debe hacer es superponer a una imagen generada en tiempo real por la cámara, los datos proporcionados por los sensores.

Observando la declaración de todos los métodos y funciones, los procesos y las variables, se agrupan bajo los métodos que comparten.

4.6 Desarrollo del gráfico mediante el empleo de OpenGL ES

Las clases y métodos principales de OpenGL ES han sido explicadas previamente en el *capítulo 4*.

A continuación, se explica el desarrollo seguido, funciones y procesos más importantes, para crear el gráfico de los tres ejes del sensor orientación en OpenGL ES [6] que presenta la pantalla principal de la aplicación.

Para trabajar con OpenGL ES, se debe crear una superficie dedicada a la visualización de los gráficos realizados en OpenGL ES. Entonces, se crea la superficie donde dibujar y se le asignan las propiedades que se quieren que esta tenga, como puede ser: tamaño, transparencia, profundidad, etc.

```
private GLSurfaceView GLSurfaceView;


GLSurfaceView = new GLSurfaceView(this);
GLSurfaceView.setEGLConfigChooser(8, 8, 8, 8, 16, 0);
GLSurfaceView.getHolder().setFormat(PixelFormat.TRANSLUCENT);
GLSurfaceView.setZOrderOnTop(true);
GLSurfaceView.setRenderer(this);
```

Para poder visualizar en la misma pantalla de la aplicación, dos vistas diferentes, es decir, la imagen del gráfico que se está creando y la imagen que proporciona la cámara, es necesario añadir a la vista de la cámara, la nueva vista creada de tal manera que queden combinadas y el gráfico se superponga a la imagen:

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

Si esta línea de código no es implementada, es posible ver únicamente una de las dos vistas en la pantalla de la aplicación, es decir, solo se podría ver la imagen de la cámara o la imagen del gráfico pero nunca los ejes superpuestos a la imagen del mundo real.

Para asegurarse de que la pantalla no contiene ningún elemento antes de comenzar a dibujar, se realiza una limpieza de esta:

```
gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
```

Una vez que se tiene la superficie, se han establecido las propiedades de esta y se ha limpiado la pantalla, es posible empezar a dibujar el elemento que se quiere que aparezca en la interfaz. Las tres líneas de código siguientes describen un array de coordenadas de los vértices usados para representar el gráfico, un array del color de los componentes usados, el tipo de figura geométrica primitiva que va a ser empleada, y el grosor que se le va a dar a las líneas que forman el elemento, respectivamente.

```
gl.glVertexPointer(3, GL_FLOAT, 0, verticesBuffer);
gl.glColorPointer(4, GL_FLOAT, 0, coloresBuffer);
gl.glDrawElements(GL_LINES, 6, GL_UNSIGNED_BYTE, indicesBuffer);
gl.glLineWidth(5.0f);
```

Para definir en qué lugar del plano se encuentran los vértices que forman la figura, se emplea el siguiente array y representa parte superior izquierda, parte inferior izquierda, parte inferior derecha y parte superior derecha, respectivamente.

```
float vertices[] = {
            0,0,0,
            1,0,0,
            0,1,0,
            0,0,1
};
```

Para establecer el color de los ejes se emplea el siguiente array que representa los colores negro, rojo, verde y azul, respectivamente.

```
float colores[] = {
        0,0,0,0,
        1,0,0,1,
        0,1,0,1,
        0,0,1,1
 };
```

Para unir los vértices y obtener la figura que se desea, se establece un orden de unión representado por el siguiente array de índices de orden:

```
byte indices[] = { 0, 1, 0, 2, 0, 3 };
```

Finalmente, para conseguir que el gráfico de los tres ejes representado se mueva de acuerdo a los valores del sensor orientación, es necesario implementar el método:

```
public void onSensorChanged(SensorEvent event) {   }
```

Y para detectar el tipo de sensor:

```
(type == Sensor.TYPE_MAGNETIC_FIELD)
```

En este momento, el gráfico de los tres ejes aparece en pantalla y su movimiento varía de acuerdo a los valores que el sensor orientación proporciona. El gráfico tiene el siguiente aspecto:

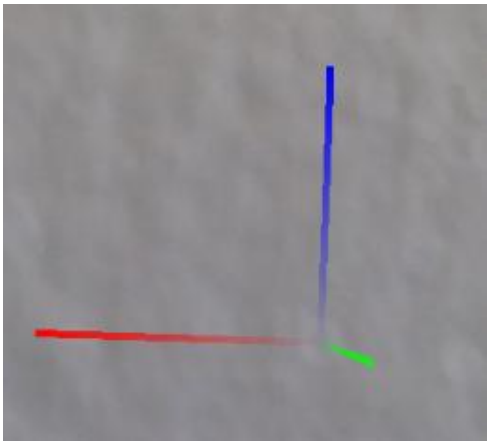


Figura 8. Gráfico OpenGL ES

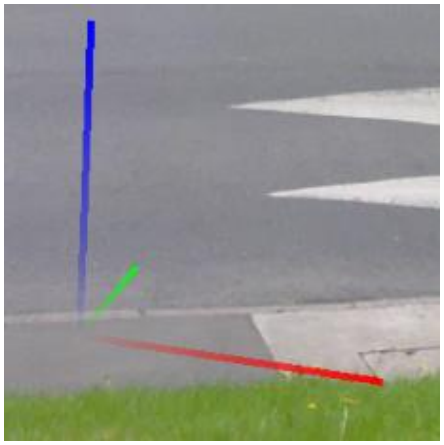


Figura 9. Gráfico OpenGL ES

4.7 Introducir los algoritmos

En este punto, todos los componentes que constituyen la aplicación están ya unidos (acelerómetro, brújula, GPS, cámara y gráfico) para finalizar el proyecto, solamente falta por implementar la parte de código que se encarga de realizar todos los cálculos explicados anteriormente en el *capítulo 3*.

Primeramente, se espera que el usuario introduzca la altura y tras ello, se buscan los datos necesarios para resolver las ecuaciones trigonométricas, como son: la posición geográfica del usuario, el ángulo de inclinación del dispositivo, bearing, etc.

La aplicación realiza internamente las ecuaciones implementadas, con los datos que ha obtenido.

Cuando se tienen los resultados, se transforman la latitud y la longitud en grados y se hace un redondeo para mostrar ocho cifras decimales de los valores obtenidos.

```
double latitud =latitude;
   double lat1= Math.toRadians(latitud);

   double longitud = longitude;
   double lon1= Math.toRadians(longitud);

   double pitch =pitchAngle;
 if(pitch<0)pitch=pitchAngle*(-1);
   double angle=Math.toRadians(pitch);

String height=editText1.getText().toString();
   double h= Double.parseDouble(height);

   double d= h * Math.tan(angle);
    if (d<0) d=(h*Math.tan(angle))*(-1);

     double bearingAngle= bearing;
    double bearing1=Math.toRadians(bearingAngle);

     double R = 6371000;

     double distanceMD=d/R;
     distanceMD=Math.toRadians(distanceMD);

   double latitude2=Math.asin(Math.sin(lat1)*Math.cos(distanceMD)
   +Math.cos(lat1)*Math.sin(distanceMD)*Math.cos(bearing1));

  double longitude2=lon1+Math.atan2(Math.sin(bearing1)*Math.sin(distanceMD)
   *Math.cos(lat1),Math.cos(distanceMD)-Math.sin(lat1)*Math.sin(latitude2));

    latitude2= Math.toDegrees(latitude2);
     longitude2= Math.toDegrees(longitude2);

     latitude2= Math.rint(latitude2*100000000)/100000000;
    longitude2= Math.rint(longitude2*100000000)/100000000;
```

4.8 Mostrar resultados por pantalla

La *clase T*oast, empleada para mostrar los resultados obtenidos sobre la información de interés buscada, permite mostrar al usuario un mensaje en pantalla superpuesto a la ventana durante unos segundos aunque sin congelar la aplicación, para luego volver a desaparecer automáticamente sin necesidad de ningún tipo de actuación por parte del usuario, y sin interferir en las acciones que esté llevando a cabo en ese momento.

Aparece por defecto en la parte inferior de la pantalla, sobre un rectángulo gris y no permite la interacción con el usuario, es decir, no se pueden introducir datos, seleccionar botones ni forzar su cierre.

```
 Toast toast1=Toast.makeText(getApplicationContext(),"Distancia:"+d,Toast.LENGTH_LONG);
toast1.show();
Toast toast2=Toast.makeText(getApplicationContext(),"Latitud:"+latitude2+";Longitud:"+longitude2,Toast.LENGTH_LONG);
toast2.show();
```

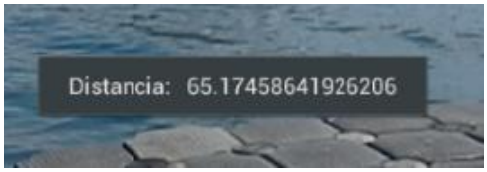


Figura 10. Mensaje con distancia resultante

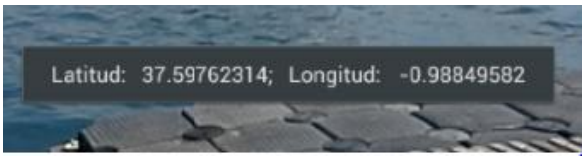


Figura 11. Mensaje con coordenadas geográficas

Capítulo 5. Funcionamiento de la aplicación

5.1 Instalar en un dispositivo móvil

Para poder hacer uso de la aplicación debe instalarse previamente en el dispositivo en el que se va a hacer uso de ella. Para ello tenemos dos opciones:

En primer lugar, si la aplicación ha sido subida al *Play Store* de Android, su descarga será muy sencilla ya que sólo será necesario buscar la aplicación que se quiere, en este caso, la aplicación recibe el nombre de *Camera2*, seleccionar su descarga y esta se instalará de forma automática en el dispositivo móvil.

Por otro lado, se puede hacer uso del *archivo APK* (Application Package File) que consiste en un archivo comprimido ZIP para el sistema operativo Android. Este paquete contiene: AndroidManifest.xml, classes.dex, resources.arsc, la carpeta res, la carpeta META-INF y por último la carpeta lib. Para instalar en un dispositivo un archivo ejecutable como el archivo .apk, se debe seguir el siguiente proceso: en Eclipse, se debe compilar el proyecto y a continuación hacer click en *File – Export* de la barra de herramientas y rellenar todos los campos que van apareciendo en las sucesivas ventanas. Cuando el archivo .apk ha sido generado y almacenado en el lugar que se desee del ordenador, tan solo debemos conectar el dispositivo al ordenador mediante un cable USB, copiar el archivo desde donde ha sido almacenado en el ordenador y pegar en la carpeta o parte del dispositivo que se quiera.

Para poder realizar este tipo de instalación, se debe tener activo en el dispositivo el permiso de poder instalar aplicaciones desde orígenes desconocidos, es decir, aplicaciones que no procedan del Play Store. Si no se tuviera activo, en el momento de instalar la aplicación, aparecería un aviso informando de ello y directamente se accederá a los ajustes para dar permiso de instalación a aplicaciones desde orígenes desconocidos.

5.2 Manual de uso

Una vez que la aplicación ha sido instalada, el usuario puede proceder a hacer uso de ella.

Cuando se abre la aplicación presenta un aspecto similar al de la imagen que se muestra a continuación. En la parte superior izquierda y en grupos se muestran los datos que proporcionan el acelerómetro, brújula y GPS respectivamente.

La imagen que se observa es la que ofrece la cámara trasera del dispositivo. Esta variará según el lugar donde se sitúe el usuario y del punto de interés que seleccione.

En el centro de la pantalla se observan tres ejes de color rojo, azul y verde. Este gráfico está realizado en OpenGL ES como se ha comentado en capítulos anteriores y representa los ejes del sensor orientación, los cuales tendrán una doble funcionalidad en esta aplicación. Por un lado, serán el punto de referencia para centrar el punto de interés que se quiere conocer en el origen de coordenadas (centro de la pantalla) y por otro lado, se

podrá observar cómo se mueve este gráfico con respecto a los valores obtenidos del sensor orientación.

Figura 12. Pantalla principal aplicación

Lo primero que se debe hacer es introducir manualmente la altura a la que se encuentra el dispositivo con respecto al suelo y en metros. Para ello, la aplicación proporciona un teclado como el que se muestra a continuación.

Figura 13. Teclado aplicación

Figura 14. Teclado introducir altura

A continuación, cuando la altura ha sido introducida, el punto de interés del cual se quieren conocer sus coordenadas geográficas y la distancia que existe entre este y la situación del usuario, es centrado en el origen de coordenadas de los ejes de la brújula representados en el gráfico. En este caso, se ha seleccionado como punto de interés un edificio del Puerto de Cartagena.

Figura 15. Punto de interés centrado en pantalla

Por último, la interfaz de la aplicación presenta en la parte inferior derecha de la pantalla un botón con el nombre de "*Get the data!*". El usuario debe presionar este botón si desea conocer la información.

Al pulsarle aparece un mensaje emergente denominado Toast (explicado en el *capítulo 4*) que muestra durante unos segundos en pantalla el resultado de la distancia, latitud y longitud de acuerdo a los datos obtenidos de los distintos sensores y de la altura introducida por el usuario, usados para realizar los algoritmos correspondientes.

Figura 16. Mostrar en pantalla distancia buscada

Figura 17. Mostrar en pantalla coordenadas geográficas buscadas

Capítulo 6. Conclusiones y futuras mejoras

6.1 Conclusiones

Con este proyecto, se han alcanzado y adquirido unos conocimientos acerca del sistema operativo Android y para su estudio se han desarrollado aplicaciones, comenzando desde las más básicas hasta alcanzar un cierto nivel tras haber adquirido los conocimientos necesarios. Esta plataforma, ofrece la ventaja de usar un lenguaje de programación bastante conocido como es Java aunque con algunas modificaciones y nuevas ideas y además el entorno de trabajo Eclipse.

Es también importante mencionar que gracias a este proyecto han sido estudiados conceptos como Realidad Aumentada y la API OpenGL ES que junto con Android tan importantes y tan usados están siendo en la sociedad debido al gran avance que estos están experimentando día tras día, a las oportunidades y a las facilidades que ofrecen.

Además, no solamente se han desarrollado los códigos de los sensores hasta su correcto funcionamiento, sino que se han estudiado un poco teóricamente, teniendo la oportunidad de probar sus funciones en diferentes dispositivos Android tales como la tablet o el smartphone y verificar que los resultados obtenidos coincidían con el estudio teórico.

Otro punto de interés ha sido el uso de la trigonometría para hacer la parte principal de la aplicación. Se ha comprobado cómo es posible unir tecnología y matemáticas para conseguir una aplicación que abarque temas importantes de hoy en día y que tan estudiados están siendo.

Finalmente, ha sido demostrado que es posible desarrollar una aplicación que calcule la distancia hasta un punto y además devuelva las coordenadas geográficas de este, usando Android como plataforma, incorporándole el concepto de realidad aumentada y un gráfico mediante OpenGL ES, objetivos principales de este proyecto.

Sin el conjunto de sensores disponibles que presentan los dispositivos móviles en la actualidad, este proyecto no se hubiese podido realizar.

6.2 Futuras mejoras

Con respecto a las futuras posibilidades de trabajo para mejorar la aplicación presente, las ideas sugeridas son las siguientes:

- Detectar la altura a la que se encuentra el dispositivo de manera automática, es decir, que sea la propia aplicación la que se encargue de obtener a qué altura se encuentra el dispositivo mediante el uso de algún algoritmo o de los sensores disponibles, en lugar de que sea introducida manualmente por el usuario que desea hacer uso de la aplicación.

- Mostrar en la imagen que ofrece la cámara una línea que represente y se mueva de acuerdo a la línea del horizonte de manera que se obtenga más información geográfica y además se siga estudiando y trabajando la gran variedad de opciones

de gráficos que ofrece la API OpenGL ES y se puedan probar las mejoras y nuevas incorporaciones que esta va experimentando.

- Poder almacenar de alguna forma la información de interés obtenida del punto seleccionado, enviarla y que la reciba otra persona en otro dispositivo Android.

- Añadir un marcador de manera que cuando el usuario seleccione un punto, un icono aparezca en dicho punto y sea posible visualizarlo desde diferentes posiciones cuando se realice un movimiento alrededor de este.

- Opción de visualizar los mapas de Google de manera que se pueda saber además de la latitud y la longitud, de manera inmediata, cuál es el nombre del lugar en el que se encuentra el usuario o el punto de interés.

Referencias

[1] Web oficial de Android.

http://developer.android.com

[2] Augmeted Reality.

http://en.wikipedia.org/wiki/Augmented_reality

[3] Mike Smithwick, Mayank Verma, "Pro OpenGL ES for Android", CA, USA, Apress Berkely, 2012.

[4] Reto Meier, "Professional Android 4 Application Development", Indianapolis (Indiana), John Wiley &Sons, Inc, 2012.

[5] Raghav Sood, "Pro Android Augmented Reality", CA, USA, Apress Berkely, 2012.

[6] OpenGL ES for Android Tutorials.

http://www.jayway.com/tag/opengl-es/

[7] Resolución de triángulos.

http://www.aritor.com/trigonometria/triangulo_trigonometria.html

[8] Calculate distance, bearing and more between Latitude/Longitude points.

http://www.movable-type.co.uk/scripts/latlong.html

[9] Bearings.

http://www.mathsteacher.com.au/year7/ch08_angles/07_bear/bearing.htm

Apéndice 1.

Código y ejemplo acerca de cómo usar las ecuaciones.

```
double latitude=52.76203393936004;
double lat1= Math.toRadians(latitude);
double longitude=-1.2412619590758918;
double lon1= Math.toRadians(longitude);

double pitch=86.97848;
if(pitch<0)pitch=pitchAngle*(-1);
double angle=Math.toRadians(pitch);

String altura=editText1.getText().toString();
 double h= Double.parseDouble(altura);
```

En este caso, para poder ser capaz de realizar el ejemplo, la altura se introduce directamente, en lugar de pedir al usuario que la introduzca por teclado o podemos asumir que esa es la altura que el usuario va a introducir o ha introducido.

```
double h=1.67; // Altura introducida del dispositivo con
                  respecto al suelo y en metros.

double distance= h * Math.tan(angle);

 double bearing=178.4354248046875 ;
double bearing1=Math.toRadians(bearing);

double R = 6371000;  //Radio de la Tierra
double distanceMD=distance/R; //distancia angular


 double latitude2=Math.asin(Math.sin(lat1)*Math.cos(distanceMD)
+Math.cos(lat1)*Math.sin(distanceMD)*Math.cos(bearing1));

 double longitude2=lon1+Math.atan2(Math.sin(bearing1)
*Math.sin(distanceMD)*Math.cos(lat1),Math.cos(distanceMD)
+Math.sin(lat1)*Math.sin(latitude2));


          latitude2= Math.toDegrees(latitude2);
          longitude2= Math.toDegrees(longitude2);

     //Mostrar por pantalla las variables y sus valores

            String finalresult= String.valueOf(latitude2);
             finalLatitudeValue.setText(finalresult);

             String finalresult1= String.valueOf(longitude2);
             finalLongitudeValue.setText(finalresult1);

              String finalresult2= String.valueOf(d);
```

```
distanceValue.setText(finalresult2);
```

Observando la consola, se obtienen los siguientes valores:

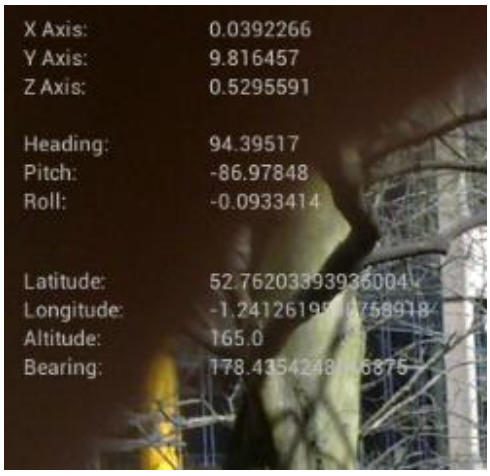
```
Distance is = 31.63812761642139;
Latitude is 52.761749516917796;
Longitude is -1.2412491211040626;
```

La latitud y longitud del punto final, del cual se desea obtener toda la información debe ser:

Latitude: 52.76195883750762
Longitude: -1,2408757209777472

Se observa la variación de algunos decimales pero esto puede deberse al pulso del usuario o no centrar bien el objetivo.

A continuación, se muestran dos imágenes. La primera de ellas muestras los datos del punto donde se encuentra el usuario, es decir, el lugar de origen y del cual se tomarán los datos necesarios para realizar las ecuaciones precisas, mientras que la segunda imagen muestra la información del segundo punto, información que debe ser obtenida mediante el empleo de las ecuaciones trigonométricas.



Punto de origen, posición usuario

Punto final