UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

I.T. Telecomunicación (Sist. de Telecomunicación)

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

"Desarrollo de una aplicación de realidad aumentada sobre Android para el apuntamiento de los nodos en el telescopio de neutrinos Antares"

TRABAJO FINAL DE CARRERA

Autor/es:
Óscar Camino Costa

Director/es:
D. Jesús Tomás Gironés

GANDIA, 2012

Índice

Capítulo 1: introducción

En las siguientes líneas se hace una breve introducción al presente proyecto, exponiendo cuál es su motivación y qué objetivos son los que persigue, así como la explicación de para que se utiliza el telescopio de neutrinos.

1.1 Motivación del proyecto

El departamento de física aplicada necesitaba una herramienta que facilitara el apuntamiento a los nodos del telescopio de neutrinos Antares situado en el mar Mediterráneo. Este apuntamiento es necesario para las pruebas de campo realizadas desde una embarcación, durante la fase de calibrado del sistema de posicionamiento ultrasónico de los nodos.

Desde un principio se pensó en utilizar una herramienta que hoy en día fuera fácil de utilizar y cotidiana. Debido a ello se pretende desarrollar una aplicación para uno de los dispositivos más en auge del momento, Android.

Se ha escogido este tipo de dispositivos ya que se está imponiendo en el mercado y muchas personas hoy en día ya disponen de uno de estos dispositivos por lo que facilitará su implantación y utilización. Además estos dispositivos disponen de un software libre y abierto.

Otro factor a tener en cuenta es que mediante un único dispositivo se puede gozar de un apuntamiento preciso gracias a la facilidad de poder utilizar varias herramientas como pueden ser el GPS, sensores de orientación y la cámara para poder ver el punto a tiempo real mediante realidad aumentada y así facilitar la tarea de localizar los nodos con un solo dispositivo.

1.2 *Objetivos*

Como objetivo principal se pretende desarrollar un visor de realidad aumentada para el sistema operativo Android. Esta aplicación recogerá puntos de interés de una base de datos y si se encuentran dentro del margen visible de la cámara mostrará un círculo que mostrará donde se encuentra el nodo del telescopio de neutrinos.

La realidad aumentada consiste en combinar una visión de un entorno físico del mundo real con elementos creados virtualmente. En concreto el proyecto se basa en el desarrollo de una aplicación para un terminal móvil del que conocemos su posición y su orientación poder dibujar sobre la imagen de la cámara obtenida en tiempo real una serie de puntos de interés geo-posicionados que son visualizados virtualmente.

El visor de realidad aumentada obtenido como objetivo principal del presente proyecto va a ser aplicado en una tarea concreta. Nuestra intención es utilizar el software anterior para el apuntamiento de los nodos del telescopio de neutrino Antares. Estos nodos se encuentran sumergidos a centenares de metros en el mar, por lo que no son visibles desde la superficie. En los trabajos de campo resulta necesario utilizar una embarcación para realizar labores de mantenimiento. La falta de visibilidad unido al continuo movimiento de la embarcación hacen muy útil la herramienta que queremos desarrollar.

Es decir el software desarrollado permitirá conocer la posición de los nodos del telescopio Antares desde una embarcación, de forma que apuntaremos con la cámara del móvil en una dirección y en la pantalla aparecerá la imagen con los nodos superpuestos a la imagen real.

Capítulo 2: Descripción de las tecnologías utilizadas

En este apartado explicaremos las herramientas utilizadas para realizar este proyecto. Empezaremos explicando en que consiste la realidad aumentada así como para que se utiliza el telescopio de neutrinos y por último algún detalle del sistema operativo utilizado, Android.

2.1 Realidad aumentada

Realidad aumentada (RA) es la superposición de elementos virtuales a la visión directa o indirecta de un entorno físico del mundo real o lo que es lo mismo crear una realidad mixta a tiempo real.

A diferencia de la realidad virtual, la realidad aumentada no sustituye la realidad física sino que simplemente añade información a la realidad física ya existente. La mayoría de aplicaciones de realidad aumentada recogen la información de la cámara y dependiendo de la aplicación realiza una serie de funciones determinadas. La mayoría de las aplicaciones sobrepone a la información de la cámara una serie de datos, ya sean puntos de intereses o imágenes.

La realidad aumentada:

- Combina elementos reales y virtuales.
- Es interactiva en tiempo real.
- Está registrada en 3D.

En la siguiente imagen se puede observar como la realidad aumentada es un término intermedio entre la realidad virtual y la realidad física. Esto es por que no se puede considerar ni realidad física ni realidad virtual ya que a la realidad física le añade información pero sigue utilizando esta información del mundo real.

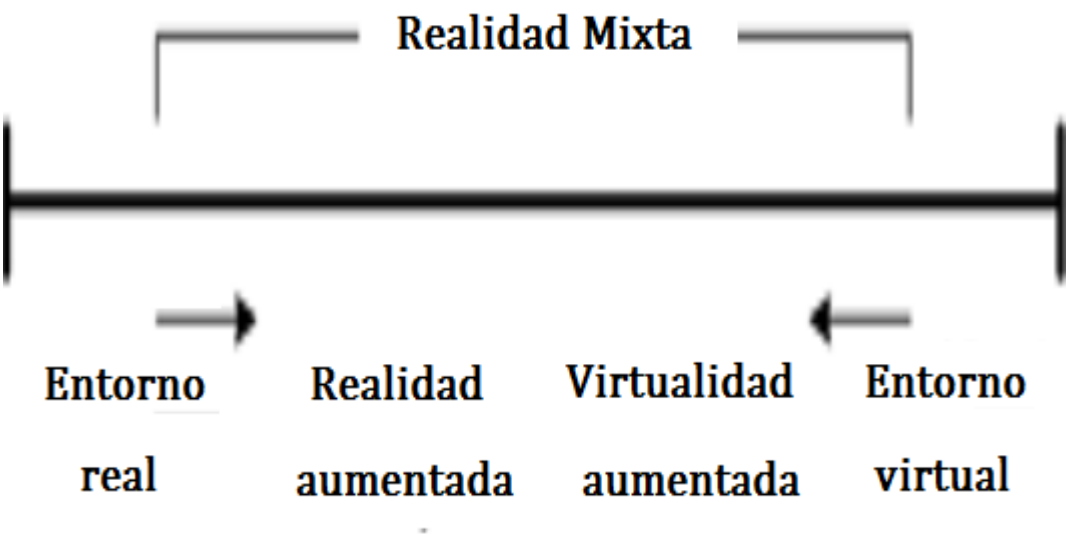

Imagen 1: Diagrama de los niveles de realidad

Por tanto, la realidad aumentada es la forma en la que definimos una visión de la realidad en la que se agregan elementos virtuales. Por ejemplo hay una aplicación que al mirar un edificio importante de algunas ciudades a través de nuestro Smartphone, la aplicación agrega información sobre la historia, características, etc. del edificio en nuestra pantalla además de mostrarnos la imagen que estamos viendo por la pantalla. Es decir, se agrega información a lo que estamos viendo. **[5]**

Imagen 2: Ejemplo de realidad aumentada

Aunque la RA no presenta reglas fijas. Un ejemplo de cuál es su utilización más frecuente, se describe a continuación:

1. Un dispositivo toma una imagen del mundo real.
2. El dispositivo ha de conocer su posición y en qué dirección se ha tomado la imagen.
3. Utilizando una base de datos con elementos geo-posicionados, se seleccionan aquellos que se supone está en el campo de visión.
4. Se superpone en la imagen real información asociada a los elementos encontrados.

Hay muchas aplicaciones de realidad aumentada. En ellas se puede añadir la información que se quiera desde fotos, simplemente un apuntador como en nuestro caso, hasta videos promocionales de empresas. Al fin y al cabo la realidad aumentada está en constante evolución para innovar y que se realizan nuevos diseños que sean útiles para los usuarios. **[6]**

Componentes de la realidad aumenta

Un sistema de RA puede tener los siguientes componentes:

1. Pantalla del móvil: en ella se verá la suma de lo real y lo virtual que conforman la realidad aumentada.
2. Cámara: dispositivo que extrae la información del mundo real y la transmite para que sea utilizado en la realidad aumentada.
3. Software: programa que toma los datos reales y los transforma en realidad aumentada.
4. Sistema de geo-localización: en muchos casos de RA resulta imprescindible conocer posición real del dispositivo. El sistema más utilizado es GPS aunque también pueden utilizarse otros como los basados en redes WiFi.
5. Sensor de dirección: Para saber hacia donde está orientado el dispositivo y así el software saber si necesita utilizar la información necesaria es imprescindible la utilización de este tipo de dispositivos. En nuestro caso nosotros utilizamos los sensores de orientación.
6. Marcadores: los marcadores básicamente es la información que se añade en las aplicaciones de realidad aumentada. Dependiendo de la aplicación se usa marcadores, imágenes o videos inclusive al reconocer los puntos de interés. **[6]**

2.2 Telescopio de neutrinos Antares

Los neutrinos son partículas elementales que apenas interaccionan con la materia. Ellas recorren largas distancias del Universo sin ser absorbidas por los medios intergalácticos, propagándose en línea recta desde el corazón de los aceleradores cósmicos, sin ser desviados, por cuya razón permiten sondear el universo remoto y estudiar el hontanar de la radiación cósmica de altísima energía.

El telescopio Antares es un detector de neutrinos situado en el mar Mediterráneo que se encarga de buscar materia oscura y estudiar la astronomía de alta energía.

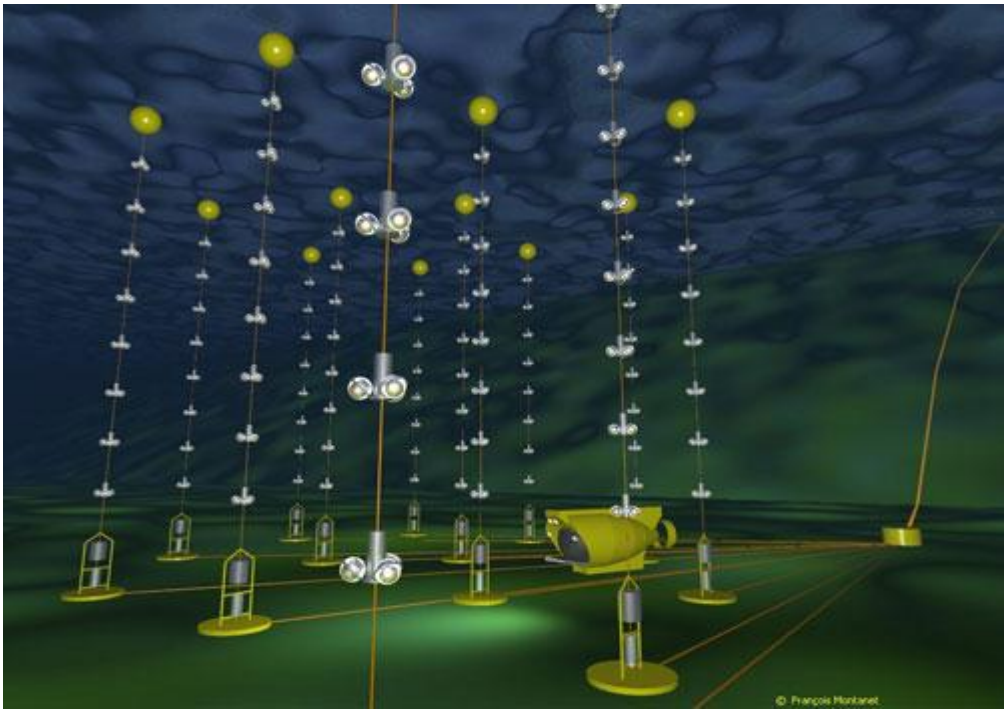

Imagen 3: Representación en imágenes virtuales del detector submarino Antares

La detección de los neutrinos es posible con inmensos detectores protegidos de la radiación cósmica que tiene la tierra que infunde ruido. Estos detectores se encuentran a lo ancho de Toulon (sureste de Francia, cerca de Marsella).

Antares se encuentra a 2.500 metros bajo el mar para proteger de la radiación solar. Antares observa el cielo del hemisferio sur a través del globo terrestre, incluyendo el centro galáctico, lugar de fenómenos energéticos intensos.

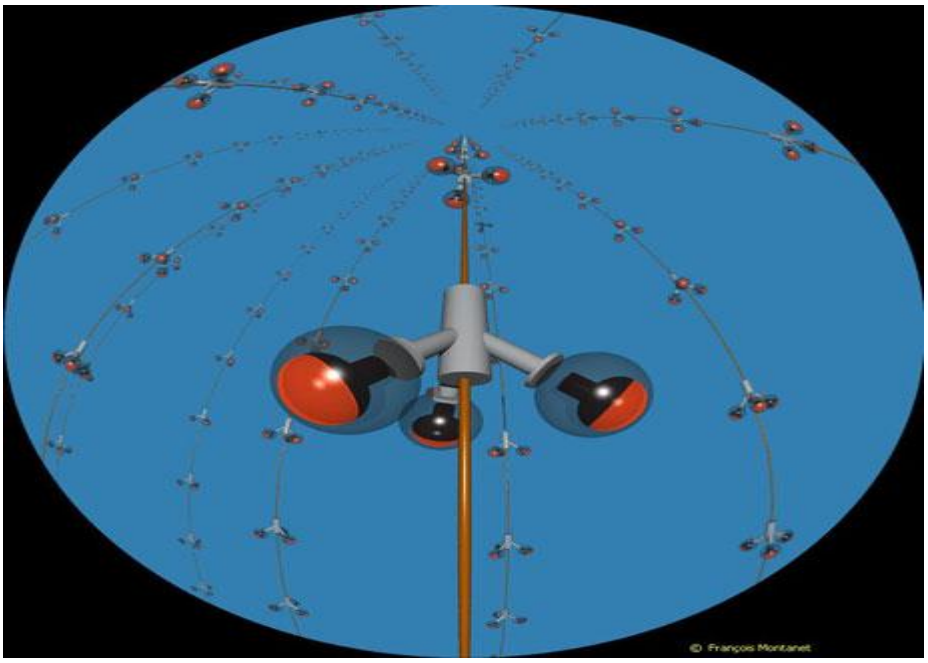

Imagen 4: Representación en imágenes virtuales del detector submarino Antares

Antares constituye una infraestructura científica submarina que va registrando diferentes datos: tanto oceanográficos como geofísicos. **[8]**

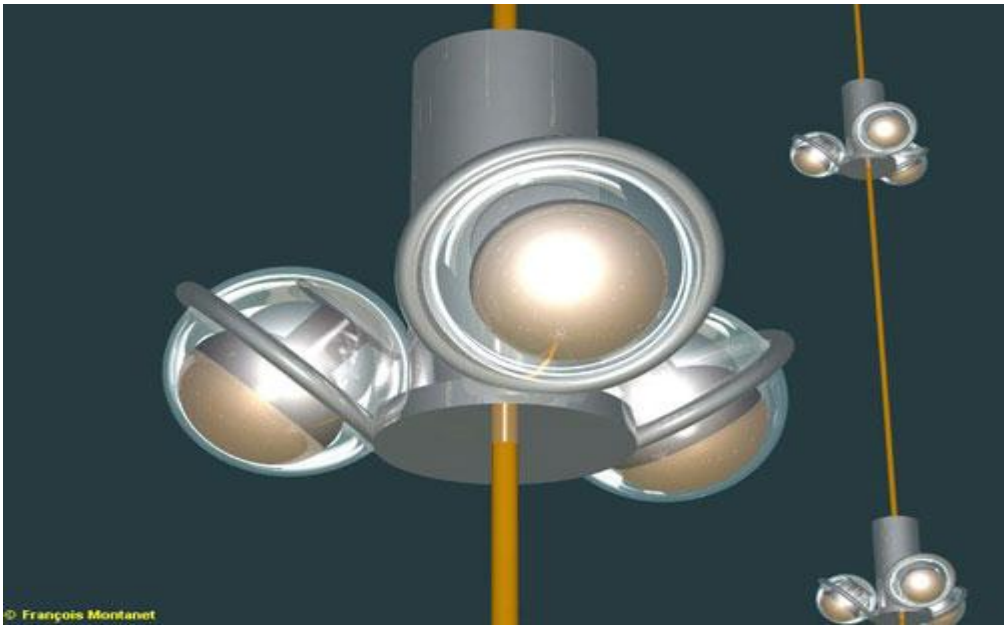

Imagen 5: Representación en imágenes virtuales del detalle de un módulo del detector submarino "Antares".

2.3 Android

La telefonía móvil ha evolucionado considerablemente. Mientras antes solo era útil para realizar llamadas y enviar mensajes hoy en día los terminales son prácticamente un ordenador. Se puede utilizar tanto para mirar páginas web, el correo o jugar a juegos incluso online. También es importante destacar como ventaja que se puede llevar a cualquier sitio sin problemas ya que su peso es reducido y podemos disponer de Internet prácticamente en todos los lugares.

El lanzamiento de Android como nueva plataforma para el desarrollo de aplicaciones móviles ha causado gran expectación y está teniendo una importante aceptación por los usuarios. En la actualidad se está convirtiendo en alternativa a otras plataformas como iPhone, Simbian, Windows Phone o BlackBerry.

Android es una plataforma de código abierto. A diferencia de otros sistemas para móviles que solo pueden ejecutarse sobre procesadores ARM, Android es independiente de la plataforma. En la actualidad existen implementaciones para ARM, MIPS, Power y la familia del x86. Para permitir que las aplicaciones sean multiplataforma se ha apostado por una ejecución en máquina virtual de código Java. No obstante también se permite realizar aplicaciones en código nativo. **[9]**

2.3.1 Características

En este para entrar más en detalle sobre el sistema operativo Android vamos a explicar las principales características de Android y compararlas con otras plataformas móviles como iOS y Windows Phone.

El sistema operativo Android tuvo su lanzamiento en 2008. El sistema Android incorporar una tienda de aplicaciones llamada Play Store donde se pueden encontrar más de 300.000 aplicaciones un número lejano a las 50.000 de Windows Phone pero ya cerca de su máximo competidor iOS que tiene unas 400.000. Android tiene la licencia de software libre y en abierto, mientras que iOS como Windows Phone son de su propiedad. Android al igual que iOS incorpora el motor de navegador web WebKit sin embargo Windows Phone tiene Pocket Internet Explorer. Una gran ventaja de Android respecto las otras dos es que es la única que soporta Flash. También es importante observar que el coste de publicar una aplicación Android es de tan solo un pago de 25 dólares mientras que para iOS como y Windows Phone tienes que pagar 99 dólares al año. Añadir que mientras iOS y Windows Phone solo tienen una plataforma de desarrollo, Mac y Windows respectivamente, Android tiene a Windows, Mac y Linux. Los terminales de la plataforma Android son los únicos que pueden incorporar memorias externas a excepción de Windows Phone e IOS. También es importante comentar que no solo hay una empresa que fabrique terminales con este sistema operativo sino que, al igual que Windows Phone hay varias compañías que fabrican móviles con este sistema operativo lo que permite que Android tengo una variedad de dispositivos muy alta no como iOS que tan solo dispone de un modelo. Por último comentar que el tipo de pantalla de este tipo de dispositivos puede ser tanto capacitiva como resistiva y como había comentado anteriormente Android como iOS puede albergar aplicaciones nativas. **[1]**

2.3.2. Arquitectura de Android

El siguiente gráfico muestra la arquitectura de Android. Como se puede ver, está formada por cuatro capas. Una de las características más importantes es que todas las capas están basadas en software libre.

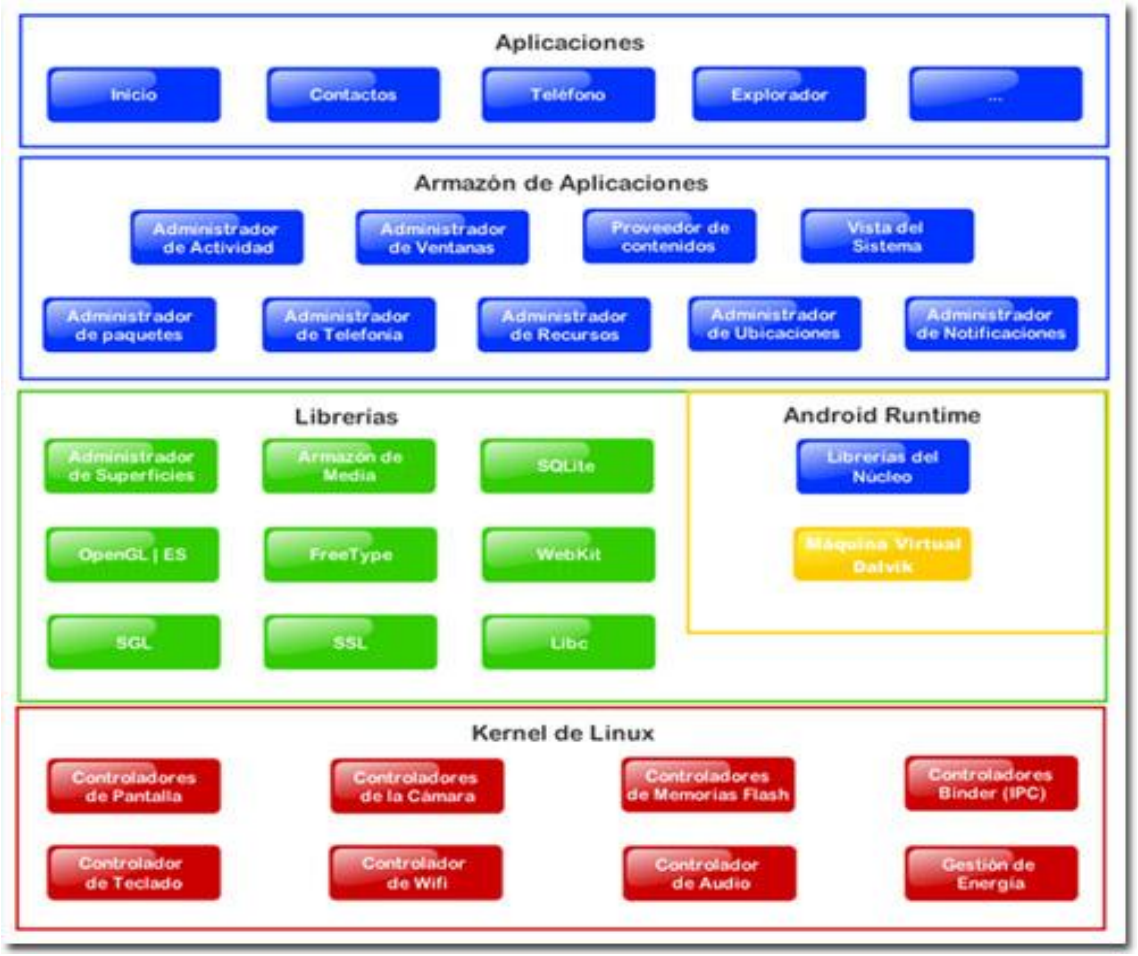

Imagen 6: Arquitectura Android **[1]**

2.3.3 Dispositivos Android

De momento, el Sistema Operativo Android sigue en proceso de desarrollo, aunque ya se han lanzado una infinidad de terminales que funcionan bajo esta plataforma.

El primer modelo lanzado al mercado bajo esta plataforma fue el HTC Dream, comercializado en EEUU y Reino Unido bajo la marca T-Mobile G1.

Android se ha convertido en competencia directa a los sistemas operativos móviles como Windows Mobile, Symbian, iPhone OS 3.0. Esto es debido a

que muchas compañías y no sólo la creadora de dicho sistema Google ha optado por está opción. Esto ha hecho que se creen muchos modelos como pueden ser el HTC Desire, HTC Hero o Samsung Galaxy SIII. **[9]**

Capítulo 3: Solución adoptada

En este apartado se justificarán las decisiones más importantes que se han tomado para implementar el sistema y se describirá su arquitectura. En concreto se comentarán las razones por las que se ha escogido el sistema operativo Android, las ecuaciones necesarias para resolver el problema de la realidad aumentada, así como se describirá el funcionamiento del software implementado.

3.1 ¿Por qué elegir Android?

Para realizar una aplicación de realidad aumentada son necesarios varios elementos que van desde un sistema de posicionamiento (como GPS), pasando por una cámara, un sensor de dirección y algo que procese todo está información para después poder visualizar la información aumentada superpuesta a la real.

Para realizar este tipo de aplicaciones antes eran necesarias varias herramientas (un ordenador, sensor de dirección, un receptor GPS) o realizar un diseño propio que incorporará todo con lo que elevaría los precios. Hoy en día un terminal móvil incorpora los elementos necesarios para cubrir todas estas necesidades por lo que será muy práctico ya que con un solo elemento de tamaño, peso y precio reducido podremos realizar este tipo de aplicaciones.

Por ello hemos elegido un terminal móvil pero, dentro del mercado de móviles existen muchas plataformas (iPhone, Symbian, Windows Phone, Blackberry, Palm, Java Mobile Edition, Linux Mobile (LiMo), etc.); sin embargo las únicas que tienen una importante aceptación en el mercado y tienen capacidad para realizar este tipo de aplicación hoy en día son Android e iPhone.

Dentro de estas dos plataformas de terminales móviles hay que decir que Android en contra de iPhone es una plataforma de desarrollo libre. Esto quiere decir que cualquiera podrá realizar aplicaciones sin necesidad de la aprobación de la plataforma y subirla para compartirla con otros usuarios.

También es importante destacar que está escrita en un lenguaje muy utilizado como es el Java y que ya conocía lo que hace que me resulte más sencillo aprender a programar en Android ya que se basa en este lenguaje y hará que no sea necesario aprender un lenguaje nuevo para desarrollar aplicaciones como tendría que hacer para realizar una aplicación para terminales iPhone.

También es importante destacar que en Android se fabrican muchos terminales de varias compañías como pueden ser HTC o Samsung y no solo dispositivos móviles sino que también se están fabricando tabletas o netbook que podrán descargar y utilizar está aplicación sin ninguna dificultad desde el Market cuando la aplicación sea subida lo que facilitará la distribución.

Como hemos visto, Android combina características muy interesantes. Android nos ofrece una forma sencilla y novedosa de implementar potentes aplicaciones para móviles.

Otro aspecto por el que elegimos este y no otro dispositivo es su cuota de mercado. En la siguiente gráfica podemos ver un estudio realizado por la empresa Grantner Group, donde se muestra la evolución del mercado de los sistemas operativos para móviles según el número de terminales vendidos.

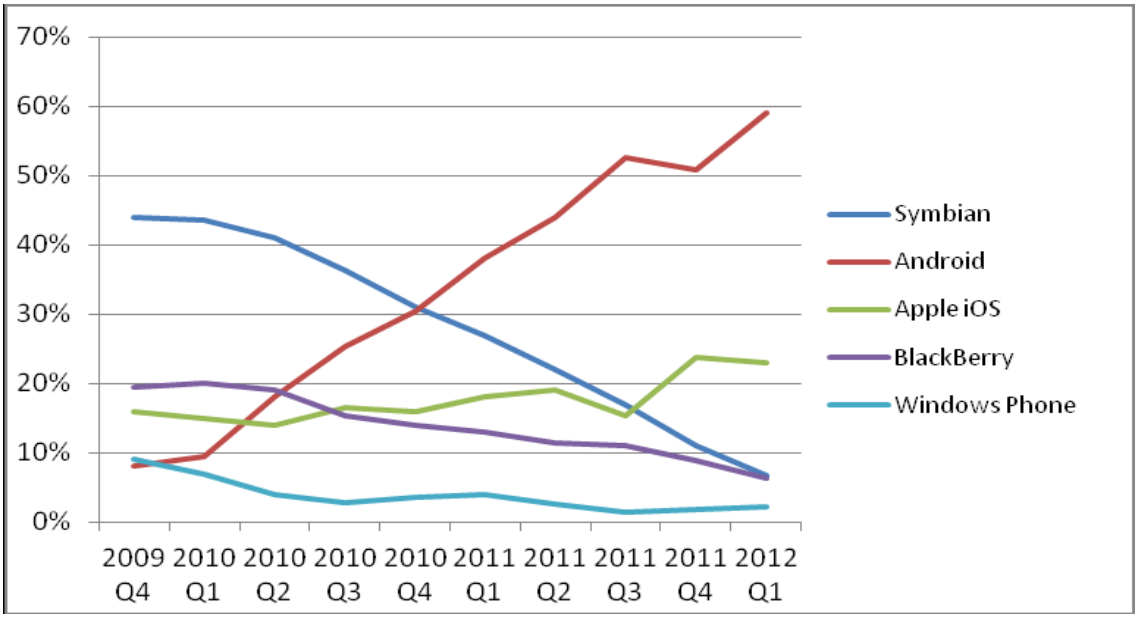

Imagen 7: gráfico comparativo de ventas de los sistemas operativos

En esta gráfica se observa que la cuota de mercado de Android es superior al resto por lo que será más probable que está aplicación pueda ser utilizada por varias personas con otros fines ya fuera del departamento de física. **[1]**

3.2 Esquema de funcionamiento

En este apartado voy a explicar con detenimiento el esquema de funcionamiento de la aplicación realizada. Como ya había comentado es una aplicación de realidad aumentada que se encargará de apuntar a los nodos del telescopio de neutrinos. El apuntamiento simplemente será incorporar a la información de la cámara un círculo en la posición exacta de la pantalla donde se encuentran dichos nodos si se encuentran dentro del campo de visión de la cámara. Para ello debemos realizar una serie de pasos:

1. Encender la cámara, para que en tiempo real se muestre la imagen sobre la pantalla.
2. Obtener la posición (longitud, latitud y altura) del terminal utilizando el GPS.

3. Obtener la información de los sensores de orientación tanto del ángulo respecto del Norte (azimut) como el ángulo de inclinación del terminal respecto del suelo (elevación).
4. Obtener posición (longitud, latitud y altura) del nodo a visualizar.
5. A partir de la posición del terminal, su orientación y la posición del nodos, calcular los ángulos tanto en elevación como en azimut desde el terminal al nodo.
6. Con los valores obtenidos en el punto anterior y conociendo el ángulo de apertura de la cámara y la resolución de la pantalla, calcular la posición en pixeles (x e y) sobre la que se proyecta el nodo en la pantalla.
7. Dibujar uno círculo en la posición que acabamos de calcular.

Estos siete pasos se debería realizar para cada nodo del telescopio de neutrinos y teniendo en cuenta que siempre que cambiara la posición del GPS o los ángulos del sensor de orientación (azimut o elevación) se tendrán que repetir los pasos y se volverán a calcular la proyección de cada nodo sobre la pantalla ya que variará en cada uno de los nodos.

Para ello hemos realizado un proyecto dividido en 4 clases. A continuación se muestra el esquema de funcionamiento:

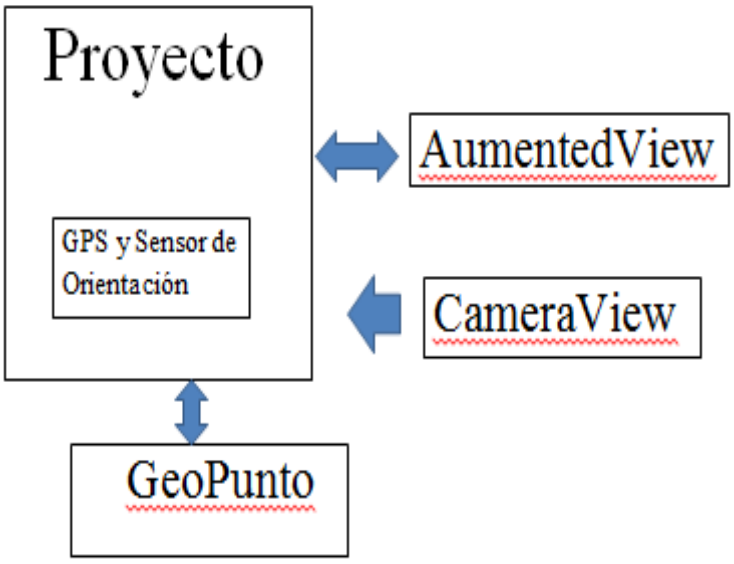

Imagen 8: Esquema del funcionamiento de la aplicación

La clase Proyecto es la clase principal y como se puede ver a en el esquema es la encargada de recibir la información tanto del sensor de orientación como del GPS. En dicha clase se utilizarán las funciones de GeoPunto que serán las encargadas de calcular los ángulos que deberíamos de tener para estar apuntando a los nodos del telescopio de neutrinos en la realidad. También esta clase incorporará un contenedor que contendrá la imagen de la cámara y los posibles círculos que apunten en el píxel adecuado a los nodos. Para ello hemos utilizado las clases CameraView y AumentedView. La primera de ellas simplemente recoge la información de la cámara. La segunda, AumentedView es la encargada de calcular si cada nodo se encuentra dentro del margen visible de la cámara y en que posición se encuentra. Para ello Proyecto le pasará las diferencias entre los puntos 3 y 4 (sensor de orientación y ecuaciones de GeoPunto) antes comentados y calculara el píxel donde se encuentra teniendo en cuenta el ancho del móvil y las dos aperturas angulares de la cámara calculando así la posición X e Y de la pantalla donde se encontraría. Como hemos comentado anteriormente para que no se queden estáticos dichos círculos, si la posición GPS o los datos de los sensores de orientación cambiaran, cambiarían los ángulos donde se encontrarían los nodos y por tanto en AumentedView si esto pasara desactivaría el anterior círculo y añadiría el nuevo con la nueva posición respecto de la pantalla calculada.

3.3 Ecuaciones

En este apartado explicaremos las fórmulas necesarias para calcular los ángulos para apuntar a los nodos del telescopio de neutrinos y también los cálculos necesarios para poder realizar la proyección sobre la imagen de la cámara.

En un principio se observó que lo necesario era calcular dos ángulos. El primer ángulo sería la desviación en azimut o lo que es lo mismo el ángulo hacia el cual tendrías que mirar desde tu posición para mirar a uno de los nodos del telescopio de neutrinos. Este ángulo en 0 correspondería con mirar al norte, 90 al este 180 al sur y

270 al oeste. El segundo ángulo que habría que calcular es la desviación en altura o elevación. Este ángulo correspondería con el ángulo que tendrías que mirar levantando o agachando la cabeza para apuntar a uno de los nodos del telescopio de neutrinos. En este caso el 0 correspondería con mirar al horizonte o mirar recto. La siguiente imagen se verá mejor explicado los ángulos:

Imagen 9: ángulos usados en la aplicación

Una vez hallados estos valores deberemos hacer la proyección sobre el móvil si estuviera dentro del margen visible y colocarlo como posición en píxeles X e Y.

Primero empezaremos por calcular la diferencia de altitud tendremos que saber nuestra altitud (que nos la dará la función getAltitude()) y la altitud de cada uno de los nodos del telescopio de neutrino que la estarán en una base de datos.

Una vez sabida la altitud tendremos que saber la distancia que separa ambos puntos. Para ello obtenemos la longitud y latitud con getLongitude() y getLatitude(). Una vez realizado esto y con la posición obtenemos la distancia.

Se utiliza la fórmula de ' Haversine " para calcular la distancia de círculo máximo entre dos puntos - es decir, la distancia más corta sobre la superficie de la Tierra –

dando la distancia entre los puntos. La fórmula necesaria para calcular la distancia es:

$$a = sin^2\left(\frac{\Delta \text{lat}}{2}\right) + \cos(lat1) * \cos(lat2) * sin^2\left(\frac{\Delta \text{long}}{2}\right)$$

$$c = 2 * atan2\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d = R * c$$

Donde R es el radio de la Tierra (radio medio = 6.371 kilómetros), Δlat es la diferencia de latitudes y Δlong es la diferencia de longitudes. c es la distancia angular en radianes, y a es el cuadrado de la mitad de la longitud de la cuerda entre los puntos. Es necesario que los ángulos estén en radianes para pasar a funciones trigonométricas.

La formula de Haversine es especialmente precisa en el cálculo numérico incluso a distancias pequeñas - a diferencia de los cálculos basados en la ley de los cosenos esférica. Fue publicado por RW Sinnott en Sky and Telescope en 1984, aunque se conoce desde hace mucho tiempo por los navegantes. Una mejora del rendimiento se puede obtener mediante la factorización de las condiciones que se obtienen al cuadrado.

Una vez calculada la distancia se calculará la elevación que directamente será ya que es un triángulo rectángulo el Arcotangente de la diferencia de altitud partido la distancia al punto que quedará así:

$$elevación = \ arctang\left(\frac{dalt}{distanciaRecta(punto)}\right)$$

Dalt es la diferencia entre altitudes y la distanciaRecta(Punto) simplemente es calcular la mínima distancia con la fórmula de haversine.

Una vez calculado faltará calcular el azimut o distancia en grados respecto al horizonte. En general, el rumbo actual variará a medida que siguen una trayectoria de círculo máximo, el ángulo definitivo será diferente de la partida inicial por diversos grados según la distancia y la latitud.

Esta fórmula es para el ángulo inicial (a veces se denomina azimut hacia delante) que si se sigue en línea recta a lo largo de un arco de círculo le llevará desde el punto inicial hasta el punto final. La formula es la siguiente:

$$azimut = arctang\left(\frac{sin(\Delta long) * cos(lat2)}{cos(lat1) * sin(lat2) - sin(lat1) * cos(lat2) * cos(\Delta long)}\right)$$

Igual que antes Δlong es la diferencia de longitudes entre el punto destino y el punto de origen. **[10]**

Una vez calculados dichos valores tendremos que calcular la proyección sobre la pantalla si se encuentra dentro del margen visible. Para ello primero calculamos las diferencia respecto azimut y respecto azimut que una vez hallada la información de los sensores quedaría así.

$$\text{difX} = \text{x\_sensor} - \text{azimut};$$
$$\text{difY} = \text{y\_sensor} - \text{elevación};$$

Para calcular la proyección simplemente sería poner el centro de la pantalla en X o Y y sumarle difX o difY por la apertura angular de la cámara respectiva quedaría así:

$$x = centerX + difX * ratioX;$$
$$y = centerY + difY * ratioY;$$

Con estos datos ya tendríamos lo necesario para dibujar el círculo dentro de la pantalla ya que si X o Y es mayor a la posición se dibujaría fuera y por lo tanto no aparecería en la pantalla con lo que tendríamos todo solucionado.

Capítulo 4: Explicación del código

En este capítulo explicaré el código que ha sido necesario para realizar la aplicación. Como había comentado anteriormente está subdividido en 4 clases. En la clase principal como ya había mencionado utilizo los sensores de orientación y GPS. A continuación se muestra el código necesario para el sensor de orientación:

4.1 Obtención de la orientación del terminal

En el siguiente código buscamos los sensores del dispositivo móvil y los metemos en una lista. Una vez hecho esto escogemos el sensor de orientación.

```
SensorManager sensorManager = (SensorManager)
getSystemService(SENSOR_SERVICE);
List<Sensor> listaSensores = sensorManager.getSensorList(Sensor.TYPE_ALL);
             listaSensores =
sensorManager.getSensorList(Sensor.TYPE_ORIENTATION);
             if (!listaSensores.isEmpty()) {
                    Sensor orientationSensor = listaSensores.get(0);
                    sensorManager.registerListener(this, orientationSensor,
                                 SensorManager.SENSOR_DELAY_UI)
```

Para extraer los datos tenemos:

```
public void onSensorChanged(SensorEvent evento) {
             synchronized (this) {
                    switch (evento.sensor.getType()) {
                    case Sensor.TYPE_ORIENTATION:
                           y_sensor = evento.values[1];
                           x_sensor = evento.values[0];
                           z_sensor = evento.values[2];

                           actualizaPosicion();
                           break;
                    }
```

Aquí se puede ver como se extrae la información cada vez que cambia el sensor de orientación y como había comentado anteriormente cada vez que esto pasa como variará

los ángulos de diferencia deberá llamar a *actualizaPosición()* que será el encargado de cambiar la realidad aumentada y se explicará mas adelante.

4.2 Obtención de la posición GPS

A continuación se muestra el código para activar el GPS:

```
mLocationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

            if
(mLocationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {

                  mLocationListener = new MyLocationListener();

                  mLocationManager.requestLocationUpdates(
                              LocationManager.GPS_PROVIDER, 100, 0f,
mLocationListener);

            } else {

                  Toast.makeText(getBaseContext(),

      getResources().getString(R.string.gps_signal_not_found),
                              Toast.LENGTH_LONG).show();
```

Se puede observar que llama el servicio de localización y si está activado el GPS y disponible llama a la función *MyLocationListener()* y recogerá los datos cada 100 ms. Si no mostrará un mensaje por pantalla indicando que el GPS está inactivo. Para poder utilizar el GPS es necesario dar permisos en AndroiManifest que serán declarados así:

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

A continuación se muestra la función *MyLocationListener():*

```
private class MyLocationListener implements LocationListener {
            public void onLocationChanged(Location loc) {

                  if (loc != null) {
                        currentLocation = loc;
                        actualizaPosicion();
                  }
            }
```

Esta función simplemente cuando la señal GPS cambie y siempre y cuando haya una localización la guardará en *currentLocation* para que después pueda ser utilizada. Como si cambia la posición GPS cambiará el ángulo también deberá llamar a la función actualizaPosicion().

4.3 Obtención de las diferencia entre los sensores y los ángulos

A continuación se muestra el método actualizaPosicion() que se encarga de calcular la diferencia entre los sensores y los ángulos calculados, como se muestra a continuación:

```
private void actualizaPosicion() {
            double difX;
            double difY;

      if (currentLocation != null) {
                  longit = currentLocation.getLongitude();
                  latit = currentLocation.getLatitude();
                  altit = currentLocation.getAltitude();
                  GeoPunto a = new GeoPunto(longit, latit, altit);
            GeoPunto b = new GeoPunto(-0.266111, 39.008611, 840); //MONDUVER

                  az = Math.toDegrees(a.azimut(b));
                  if (az < 0) {
                        az = az + 360;
                  }
                  el = Math.toDegrees(a.elevacion(b));
                  difX = x_sensor - az;
                  difY = y_sensor + 90 + el;

                  ca.ActualizaAumentedPoint(difX, difY, z_sensor);
            }

      }
```

Esta función es la encargada de pasar la información relevante a la clase *AumentedView*. En ella se extrae la información de altitud, longitud y latitud antes recogida en *currentLocation y crear un GeoPunto que será explicado más adelante. S*e calcula los ángulos mediante la utilización de las funciones *azimut()* y *elevación().* Como el departamento de física aún no tiene creada la base de datos hemos escogido un punto relevante para poder probar como es el Mondúber. También se puede observar que el ángulo de azimut se le suma 360 si es negativo como habíamos comentado

anteriormente. Después se calculan las diferencias respecto de los dos ejes. *DifX* simplemente será la resta ya que los valores devueltos por azimut serán siempre positivos entre *0 y 360* pero *DifY* devuelve valores tanto positivos como negativos así que será la suma. También tiene la peculiaridad que se suma 90 ya que en posición vertical que es como funciona la aplicación el valor es -90 y no 0.

4.4 Utilización de las vistas

A continuación se muestra como se añade la información tanto de la cámara como la información adicional que hará la función de realidad aumentada:

```
cv = new CameraView(this);
ca = new AumentedView(this);
rl = new FrameLayout(this.getApplicationContext());
setContentView(rl);
rl.addView(cv);
rl.addView(ca);
```

Primero recoge las dos vistas y las añade a un *FrameLayout* que simplemente es el contenedor que mostrará por pantalla.

4.5 Clase GeoPunto

Ahora pasaremos a comentar la clase GeoPunto que será la encargada de crear los puntos de interés y calcular los ángulos.

```
public GeoPunto(double longitud, double latitud, double altura) {
            this.longitud = longitud;
            this.latitud = latitud;
            this.altura = altura;
      }
```

Aquí se observa que recoge la información que les de y las guarda en las variables creadas.

```
public double azimut(GeoPunto punto) {
            double lat2 = Math.toRadians(punto.latitud);
            double lat1= Math.toRadians(latitud);
            double long1=Math.toRadians(longitud);
            double long2=Math.toRadians(punto.longitud);
            double dLon = long2 - long1;
```

```
            double y = Math.sin(dLon) * Math.cos(lat2);
            double x = Math.cos(lat1)*Math.sin(lat2) -
                    Math.sin(lat1)*Math.cos(lat2)*Math.cos(dLon);
            return Math.atan2(y, x);
      }
```

En esta función se calculará el ángulo azimut hacia delante. Simplemente hemos utilizado la expresión antes mencionada y ha sido pasada a código.

```
      public double elevacion(GeoPunto punto) {
            double alt1=altura;
            double alt2=punto.altura;
            double dalt=alt2-alt1;
            return Math.atan2(dalt, distanciaRecta(punto));
      }
```

La función *elevación* también hemos pasado a código lo explicado en esa sección.

```
      public double distanciaRecta(GeoPunto punto) {
            final double RADIO_TIERRA = 6371000; // m

            double lat1=Math.toRadians(latitud);
            double lat2=Math.toRadians(punto.latitud);
            double lon1=Math.toRadians(longitud);
            double lon2=Math.toRadians(punto.longitud);

            double dLat = lat2 - lat1;
            double dLon = lon2-lon1;

            double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
                    Math.sin(dLon/2) * Math.sin(dLon/2) * Math.cos(lat1) *
Math.cos(lat2);
            double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
            return RADIO_TIERRA*c;
      }
```

Y aquí vemos *distanciaRecta* que es la encargada de calcular la mínima distancia entre los dos puntos. El código también corresponde a pasar la función a código sin modificaciones.

4.6 Obtención de la imagen de la cámara

Una vez ya hemos explicado esto faltará explicar las dos vistas. Primero pasaremos a explicar la clase CameraView que es la encargada de recoger la información de la cámara:

```
public CameraView(Context ctx) {
                  super(ctx);
```

```
                previewHolder = this.getHolder();

        previewHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
                previewHolder.addCallback(surfaceHolderListener);
              }


       SurfaceHolder.Callback surfaceHolderListener = new
SurfaceHolder.Callback() {

public void surfaceCreated(SurfaceHolder holder) {
                  camera=Camera.open();

                  try {
                          camera.setPreviewDisplay(previewHolder);
                  }
                  catch (Exception e){ }
                 }


public void surfaceChanged(SurfaceHolder holder, int format, int width,
                int height)
          {
             Parameters params = camera.getParameters();
             params.setPreviewSize(width, height);
             params.setPictureFormat(PixelFormat.JPEG);
                    camera.setParameters(params);
                    camera.startPreview();
          }


 public void surfaceDestroyed(SurfaceHolder arg0)
          {
             camera.stopPreview();
             camera.release();
          }
        };

       }
```

Esta clase implementa la interfaz SurfaceHolder.Callback. La claseSurfaceHolder es el contendor que nos da acceso a la superficie, la cual no se suele usar directamente. A través de los métodos de esta interfaz nos enteraremos de cuando se crea, destruye o cambia de tamaño la superficie.

Podemos ver que se indicar al SurfaceHolder que notifique los cambios de la superficie a nuestra instancia. Luego especifica el tipo de superficie a usar, en este caso una que no posee sus propios buffers.

En los tres últimos métodos se notifican los eventos relacionados con la superficie donde mostramos la imagen de la cámara. El primero de ellos es surfaceCreated. El bucle de nuestra aplicación recibe un mensaje indicándole que la superficie ha sido creada, e invoca este método pasándole el SurfaceHolder de la misma. Activamos la cámara y le decimos que muestre la imagen de previsualización usando la superficie asociada al holder. Si hay algún problema liberamos la cámara.

Cuando la superficie se destruye (por ejemplo, porque se va a iniciar otra actividad) detenemos la previsualización y liberamos la cámara.

El método surfaceChanged se invoca si la superficie cambia de tamaño (si la pantalla gira, por ejemplo). En nuestro ejemplo no se usará, porque vamos a fijar la ventana de la actividad a formato landscape. **[11]**

Para usar la cámara necesitaremos los permisos adecuados, en este caso:

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

Que tendrán que ser declarados en AndroidManifest.

4.7 Añadiendo la realidad aumentada

Por último falta añadir la información adicional a la realidad física para ello está la clase *AumentedView*. Está dividida en tres funciones. La primera de ella es la encargada de calcular la posición de la pantalla donde se encuentra el punto singular. Para ello si es llamada invalidará la vista anterior y calculará la posición X e Y de la pantalla.

```
public void ActualizaAumentedPoint(double difazimut, double difelevacion,
float balanceo){
            x=(int) (centerY + difazimut*ratioX
            y=(int) (centerX - difelevacion*ratioY
            rotacion = balanceo;
            invalidate();
```

```
        }
```

Estos valores se calcularán poniendo el centro y contando la diferencia en azimut o elevación partido los píxeles que hay por grado.

A continuación observamos que se utilizan los valores *centerX*, *centerY*, *ratioX* y *ratioY*. Estos valores dependerán del móvil por ello se ha creado la siguiente función:

```
protected void onSizeChanged(int with, int heigth,
             int with_before, int height_before) {
                    super.onSizeChanged(with, heigth, with_before,
height_before);
                    centerX=with/2;
                    centerY=heigth/2;
                    ratioX=heigth/APERTURA_ANGULAR_X;
                    ratioY=with/APERTURA_ANGULAR_Y;
       }
```

Si cambia el tamaño lo recalcula. Al calcular el tamaño de la pantalla el centro de la pantalla simplemente será el medio y ratio simplemente será el tamaño de la pantalla partido la apertura de la cámara. Esto equivale al número de píxeles que tiene la pantalla en función con los grados.

Y ya para finalizar este apartado estará la función que se encargará de dibujar un círculo en esa posición:

```
protected void onDraw(Canvas canvas){
                    Paint pincel = new Paint();
                    pincel.setColor(Color.BLUE);
                    pincel.setStrokeWidth(8);
                    pincel.setStyle(Style.STROKE);
                    canvas.save();
                    canvas.rotate(rotacion, centerY, centerX);
                    canvas.drawCircle(y, x, 20, pincel);
                    canvas.restore();
             }
```

Simplemente guarda el anterior y lo rota para solucionar los problemas con los ejes y dibuja el círculo en la posición que hemos calculado anteriormente y restaura el canvas haciéndolo otra vez visible.

Capítulo 5: Pruebas de campo y ajustes en el terminal

En este apartado simplemente explicaré que prueba he realizado para comprobar que la aplicación funciona correctamente y los ajustes que han sido necesarios para que el punto se moviera a la velocidad que movíamos el móvil.

En primer lugar como he comentado anteriormente hicimos los cálculos para un punto singular conocido como es el Mondúber ya que era imposible probarlo para el telescopio de neutrinos Antares por la distancia.

Desde la universidad teníamos visión directa por lo que podíamos comprobar perfectamente si los cálculos realizados eran correctos. Simplemente apuntamos a la cima y observamos que nos daba el mismo resultado con los cálculos que con el sensor de orientación.

Una vez comprobado que estaban correctas las funciones simplemente teníamos que ajustar el ángulo de aperturas de la visión de las cámaras. Para ello fuimos probando valores. En primer lugar para que apuntara al norte y luego para que fuera modificando la orientación del móvil y el círculo siguiera apuntando a la cámara.

Una vez realizado esta prueba desde la Escuela Politécnica de Gandía había que probarlo desde otro punto para comprobar que no solo funcionaba desde ahí. Me

desplacé hasta Xeresa donde tenía también visión con la cima y se podía comprobar efectivamente que seguía funcionando como era de esperar así que ya estaba en perfecto funcionamiento y teníamos la apertura angular de nuestro dispositivo correctamente.

Capítulo 6: Conclusiones y líneas abiertas

En este apartado explicaré las líneas abiertas que han quedado y las conclusiones que podemos sacar de este proyecto final de carrera.

En primer lugar comentar que la aplicación solo funcionará en posición vertical con una pequeña inclinación del balanceo. Esto es debido a que el sensor de orientación deriva sus datos mediante el procesamiento de los datos de sensores sin procesar desde el acelerómetro y el sensor de campo geomagnético. Debido al procesamiento pesado que esto implica, la exactitud y la precisión del sensor de orientación es disminuida (específicamente, este sensor sólo es fiable cuando el componente balanceo es 0). Por ello para un posible trabajo se deberá de estudiar esta situación y observar como será posible corregirla. **[12]**

También se observa que la aplicación realiza saltos bruscos y otras aplicaciones comerciales no, como podría ser el caso de *Layar*. Para evitar estas situaciones se podrían utilizar filtros. Este problema también es debido a los sensores y para el usuario es bastante molesto por lo que también debería de ser analizado.

Otro mejora que tendrá que realizarse en un futuro para que la aplicación sea totalmente operativa es la obtención en tiempo real de los nodos del telescopio Antares. Será imprescindible establecer un mecanismo de comunicación entre el sistema Anteres y la aplicación para el intercambio de esta información.

Como conclusión comentar que es una aplicación útil para las necesidades del departamento de física con las cuales hemos cubierto el objetivo. Está aplicación podrá ser utilizada en posición vertical sin ninguna dificultad y con total precisión con lo que se podrá apuntar a los nodos con facilidad desde cualquier Smartphone que es lo que se pretendía con lo que se han cumplido los objetivos que nos marcamos al embarcarnos en este proyecto.

Capítulo 7: Bibliografía

1 "El gran libro de Android". Jesús Tomás Gironés (2011), Ed. Marcombo, ISBN 9788426717320.

2 "Desarrollo de un emisor de ultrasonidos para posicionar el telescopios de neutrinos KM3Net" Llorens Alvarez, Carlos David (2011), Tesina de master http://riunet.upv.es/handle/10251/14607, 2011. Directores: Sogorb Devesa, TC. y Ardid Ramírez, M.

3 "Android: Guía para desarrolladores" Frank Ableson (2011), Ed. Anaya Multimedia, ISBN 9788441529588

4 "Realidad aumentada. Fundamentos y aplicaciones" Rocio Vian Gimeno (2011) Recurso electrónico: http://hdl.handle.net/10251/14095

5 "Realidad aumentada" Wikipedia (2011) Recurso electrónico: http://es.wikipedia.org/wiki/Realidad_aumentada

6 "Realidad aumentada" de César Vallejo Martín (2010)

Recurso electrónico: http://recursostic.educacion.es/observatorio/web/cajon-de-sastre/38-cajon-de-sastre/922-realidad-aumentada

7 "¿Qué es la realidad aumentada" de Santiago Bernal Betancourth (2009)

Recurso electrónico: http://www.maestrosdelweb.com/editorial/que-es-realidad-aumentada/

8 “Antares: un telescopio de neutrinos en el fondo del Mediterráneo” del departamento de física de la UPV (2012)

Recurso electrónico: http://www.astroseti.org/noticia/2089/antares__telescopio_neutrinos_fondo_del_mediterraneo

9 “Que es Android: Características y Aplicaciones” de Ángel Vilchez (2009)

Recurso electrónico: http://www.configurarequipos.com/doc1107.html

10 “Calculate distance, bearing and more between Latitude/Longitude points” de Chris Veness (2002)

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

11 “Pasado, presente y futuro de la realidad aumentada” de Javier Candela (2009)

Recurso electrónico: http://javiercancela.com/

12 “SensorManager” de Android Developers (2012)

Recurso electrónico: http://developer.android.com/reference/android/hardware/SensorManager.html