Trabajo Fin de Grado

Aplicación de realidad aumentada por geolocalización para visualizar trayectorias y colocar obstáculos

Autor

Raúl Castán Gabarre

Director

Alfonso Puértolas Marcén

Grado en Ingeniería Informática

Escuela de Ingeniería y Arquitectura

2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Raúl Castán Gabarre,

con nº de DNI 18056262C en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Aplicación de realidad aumentada por geolocalización para visualizar trayectorias y colocar obstaculos,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 21 de Julio de 2017

Fdo: Raúl Castán Gabarre

Resumen

El objetivo de este proyecto era desarrollar una aplicación en Android que servirá de interfaz de un software de prevención de accidentes.

Esta aplicación utilizara la realidad aumentada para mostrar los resultados de las simulaciones sobre el terreno analizado.

El software simulador utilizado calcula los posibles accidentes en circuitos de motociclismo, mostrando un mapa del mismo con las trayectorias de escape más probables de vehículos y pilotos, basados en datos del terreno y de los recorridos realizados en cada circuito.

La aplicación a desarrollar debe partir de estas trayectorias, en forma de coordenadas, para proyectarlas sobre el circuito. Permitiendo ver en el propio terreno los datos clave del informe de seguridad creado por el simulador, como los lugares más probables de accidente y detalles de las posibles colisiones con las protecciones o paredes.

Así pues, los objetivos más importantes a cubrir por la aplicación son:

- Mediante realidad aumentada, mostrar en el circuito las trayectorias detalladas de los accidentes simulados.
- Mostrar las colisiones con las barreras y protecciones del circuito, junto con la velocidad estimada de impacto.
- Permitir añadir y eliminar protecciones al circuito, actualizando las colisiones si es necesario.

Para desarrollar esta aplicación se requiere examinar las herramientas de desarrollo de aplicaciones con realidad aumentada existentes, buscar una que cumpla con los requisitos de la aplicación (Geolocalización, compatibilidad con cualquier dispositivo Android y suficiente libertad como para cumplir los requisitos).

Índice

Introducción

En todos los deportes de motor la seguridad es uno de los factores mas importantes a la hora de diseñar un circuito. Dadas las altas velocidades que se alcanzan en una carrera, los organizadores deben buscar todas las herramientas disponibles para garantizar la seguridad tanto de los participantes como de los asistentes.

En el mercado de las herramientas de seguridad, es tan importante la fiabilidad de los datos como la buena presentación de los mismos, ya que una buena presentación facilita que un software destaque sobre los demás.

Con eso en mente, desde la empresa Tap Consultoría se buscaba una aplicación que utilizase la realidad aumentada para mostrar los resultados del informe de seguridad sobre el propio circuito analizado. Puesto que ya contaban con un software de seguridad para circuitos de motociclismo y con el dispositivo ORA-1, unas "*smartglasses*" con pantalla semitransparente, se ideo una aplicación que permitiese ver las trayectorias y el alcance de los posibles accidentes cuando el usuario estuviese mirando en la dirección de estos. Gracias al gps, giroscopio y demás sensores del dispositivo, se puede calcular la orientación de la pantalla y ajustar su contenido a lo que se esta viendo a través de ella.

Ilustración 1: Muestra de las lineas de escape en una curva, escaladas para ajustarse al terreno.

El simulador calcula las trayectorias de un piloto y su vehículo tras una caída en cualquier punto del circuito. Para esto se introducen los datos de la trazada del circuito, dividiéndolo en tramos y guardando las coordenadas que definen cada tramo, se ajusta a la trayectoria óptima de los pilotos por el mismo y a la velocidad que intentan llevar en cada punto. A partir de aquí, se elige un punto en la trazada y se simula una caída del piloto en ese punto. Se tienen en cuenta los distintos terrenos que puede recorrer en la caída (asfalto, grava, hierba, etc) y se calcula en que punto se detendrá, o en el caso de haber una barrera, a que velocidad se produce la colisión.

Este proyecto nos permite pasar de un informe en papel, con los resultados dibujados sobre un mapa del circuito a una vista directa del circuito real, resaltando las consecuencias que tendría un accidente que ocurriese en el punto donde esta situado el usuario. Ademas permite interacción directa con la simulación, permitiendo añadir barreras simplemente recorriendo a pie la ubicación que tendría la misma, lo que permite ver las posibilidades de una colisión contra una barrera inmediatamente. Esto permite revisar sin necesidad de volver a un puesto con un ordenador y rehacer las simulaciones.

La aplicación se desarrollara en Android, para que se pueda utilizar desde una amplia variedad de dispositivos, no obstante, tanto la interfaz de usuario como las interacciones con esta se diseñan con el objetivo de ser usadas desde un dispositivo ORA-1. Las ORA-1 son unas “smartglasses”, un par de gafas que soporta una pantalla “see-through” o semitransparente, lo que nos permite ver lo que se muestra en la pantalla superpuesto a lo que hay tras ella.

Para facilitar la sincronización entre estas dos escenas (la pantalla y lo que hay detrás), se buscara un motor gráfico, o kit de desarrollo que simplifique el uso de la realidad aumentada, buscando ademas que la aplicación no solo sirva para las gafas ORA-1 sino para cualquier dispositivo Android con los sensores necesarios, que a día de hoy podemos encontrar en cualquier smartphone. Basta con una cámara trasera, un GPS y un giroscopio para permitir a la aplicación saber cuando estamos sobre el circuito, cuando estamos mirando en la dirección de una simulación y cuando debe mostrar los detalles acerca de una moto que ha salido despedida en un accidente.

Ilustración 2: Dos muestras de la aplicación vista a través de una pantalla see-trough montada en unas smartglasses.

El principal problema que surge al utilizar este tipo de dispositivos es la interacción con el usuario. Mientras que navegar por la simulación se convierte en algo trivial, puesto que se enlaza a los movimientos del usuario, decidir que mostrar y que ocultar para no sobrecargar la vista se puede convertir en un problema. Las pantallas de los dispositivos wearables no suelen ser muy grandes, y el hecho de estar viendo a través de ella dificulta la lectura de información. Por esto se ha decidido filtrar la información que se vera en la aplicación a la mas esencial, y se mantiene oculta hasta que el usuario la necesita, para lo cual solo ha de pulsar sobre la simulación que quiere resaltar.

Tabla de requisitos

Funcionales

- La aplicación debe permitir visualizar los datos del informe superpuestos al circuito, manteniendo una representación fiable del informe cuando el usuario se mueva y desapareciendo cuando salgan de su campo de visión.
    - Los datos del informe que se desean ver son: los puntos de riesgo, las trayectorias de escape tras un accidente, tanto del piloto como de su vehículo, las colisiones contra las protecciones y la velocidad estimada del objeto en todos los elementos anteriores.
- La aplicación debe permitir la visualización de varios circuitos y cargar unicamente los datos del circuito en el que se encuentre el usuario.
- No es necesario que la aplicación realice la simulación, pero, de usar un simulador externo, deberá permitir actualizar los datos del circuito.
- La aplicación permitirá colocar barreras en la posición del usuario. Estas barreras se podrán añadir a la simulación para posterior uso.
    - Este caminara a lo largo de la ubicación de la nueva barrera y mediante sus coordenadas se añadirá a la simulación.
    - Se mostraran las colisiones, si las hubiera, con las trayectorias de la simulación actual.
- Un nuevo usuario, estando sobre un circuito, no debe tardar mas de un minuto en visualizar el informe del circuito en el que se encuentra.

No funcionales

- La aplicación deberá funcionar de forma optima en un dispositivo ORA-1 (Smartglasses con S.O. Android)
- (Opcional) La aplicación se podrá ejecutar en cualquier dispositivo Android, aunque la funcionalidad de realidad aumentada se vea mermada.
- (Opcional) La aplicación permitirá un sistema alternativo a la localización por GPS para que el usuario se mueva a lo largo del circuito.

Contenidos

En esta memoria se expondrán los principales elementos que forman parte de este proyecto, que se pueden dividir en 3 apartados.

Simulador de accidentes

Este programa crea las simulaciones y calcula los resultados a mostrar en la aplicación.

Puesto que la aplicación muestra los datos generados por este simulador, se diseño e implemento un método de exportación y comunicación de dichos datos. De esta forma podemos obtener los últimos resultados de cada circuito en cualquier parte y aplicar los cambios realizados desde la aplicación.

Realidad aumentada

La realidad aumentada consiste en visualizar una porción del mundo real a través de un dispositivo digital, generalmente añadiendo algún tipo de contenido virtual que se superpone a la imagen original y se actualiza en tiempo real. En este proyecto vamos a mostrar sobre un circuito de motociclismo una serie de simulaciones que representan las trayectorias de los vehículos y pilotos tras un accidente.

Al estar orientada hacia dispositivos wearables, hay que tener en cuenta que la interacción con el usuario esta limitada, puesto que no disponemos de una gran pantalla táctil que se pueda llenar de botones. Esto obliga a buscar una forma de mostrar la información al usuario sin saturar la pantalla y prácticamente sin requerir interacción por parte del usuario.

Kit de desarrollo de realidad aumentada

Se deben crear los objetos a representar tal y como se verían en un mundo tridimensional y luego llevarlos a la pantalla de tal forma que solo se vean cuando esta este orientada correctamente. Aunque existen librerías básicas como OpenGL que permiten crear todo tipo de aplicaciones gráficas, se buscara una librería que de una capa superior sobre las utilidades de OpenGL, para facilitar el proceso de desarrollo. Esta librería debe facilitar la sincronización de la cámara virtual de OpenGL con la pantalla real del dispositivo, donde se deben representar los objetos perfectamente sincronizados con el mundo real.

Estado de las tecnologías

Vamos a dedicar una pequeña sección a recordar el *estado del arte*, es decir, el estado de las tecnologías utilizadas en el desarrollo de este proyecto, en el momento en que se planteó y realizó el mismo.

- Realidad Aumentada.

  Pese a que este concepto existe desde hace varias décadas, su uso se reducía a sistemas de navegación o a la televisión. La mayor publicidad que ha recibido esta tecnología fue la llegada de las Google Glass, lo que atrajo una enorme inversión en este sector y promovió el desarrollo de herramientas para crear sistemas con realidad Aumentada.

  No obstante, cuando se planteo este proyecto en 2015, las gafas de Google ya habían caído en el olvido y no se encontraron grandes usos para la RA. Los automóviles de gamas altas empezaban a proyectar información sobre el parabrisas, comenzaron a surgir aplicaciones que permitían usar la cámara del smartphone para ofrecer información sobre el lugar donde nos encontrásemos, pero no fue hasta el lanzamiento del videojuego de Niantic *Pokemon Go* que la realidad aumentada llegase a millones de personas.

  Actualmente, en 2017, la popularidad de la realidad aumentada ha vuelto a decaer, quizá por la falta de dispositivos dedicados a esta tecnología o porque todavía hay aspectos en los que debe mejorar, no se suelen ver mas aplicaciones de esta tecnología en el día a día que las que ya existían hace 10 años.

- SmartGlasses

  Como se ha indicado antes, este proyecto comenzó poco después del lanzamiento, y olvido, de las GoogleGlass. Pese a que crearon una alta expectación, su alto precio y sus limitadas prestaciones, comparadas con los rápidos avances en smartphones, impidieron que las gafas se hicieran un hueco en el mercado. Hubo muchas otras marcas que también lo intentaron, como las Ora-1 utilizadas en este proyecto, y a día de hoy siguen saliendo nuevos diseños, con mejores procesadores, baterías de mayor capacidad y pantallas de alta resolución. No obstante, el estado de las smartglasses no ha cambiado desde que se comenzó este proyecto, y sin encontrar un hueco en el mercado.

- Android

  Android es el sistema operativo mas usado en smartphones, con millones de usuarios y cientos de versiones, y ya era así cuando se empezó el proyecto. El único aspecto destacable, que afecto al desarrollo del proyecto, fue el lanzamiento de Android Studio, un IDE especifico para el desarrollo de aplicaciones en Android. Junto con el lanzamiento del nuevo IDE, se descatalogó el Plug-in de Android para el IDE Eclipse, que hasta entonces era la

herramienta con soporte oficial. Pese a estar descatalogado, la librería de RA a utilizar se creo con Eclipse y estaba pensada para integrarla desde ese entorno. Tras analizar brevemente lo que habría supuesto migrar una librería ajena a Android Studio para poder desarrollar la aplicación con este nuevo entorno, se descarto esa opción, pese a que personalmente estaba mas familiarizado con el nuevo entorno.

Simulador De Accidentes En Circuito

Puesto que el objetivo de este proyecto no es calcular las zonas de riesgo de accidentes se necesita de un programa externo que genere dichos datos.

Este programa debe simular, a lo largo del circuito, varias caídas de los pilotos, calculando la trayectoria que seguirían, a que velocidad se produciría la caída, donde se detendrían y, en caso de colisión, a que velocidad ocurriría el impacto.

El simulador utilizado ya exportaba los resultados en una serie de ficheros de texto, que contienen instrucciones para recrear el mapa de la simulación en autocar, pero estos carecían de algunos datos que se querían mostrar, como la velocidad del piloto y su vehículo a lo largo de su trayectoria tras una caída o accidente.

Existía pues la necesidad de modificar la exportación de datos del simulador para incluir estos datos.

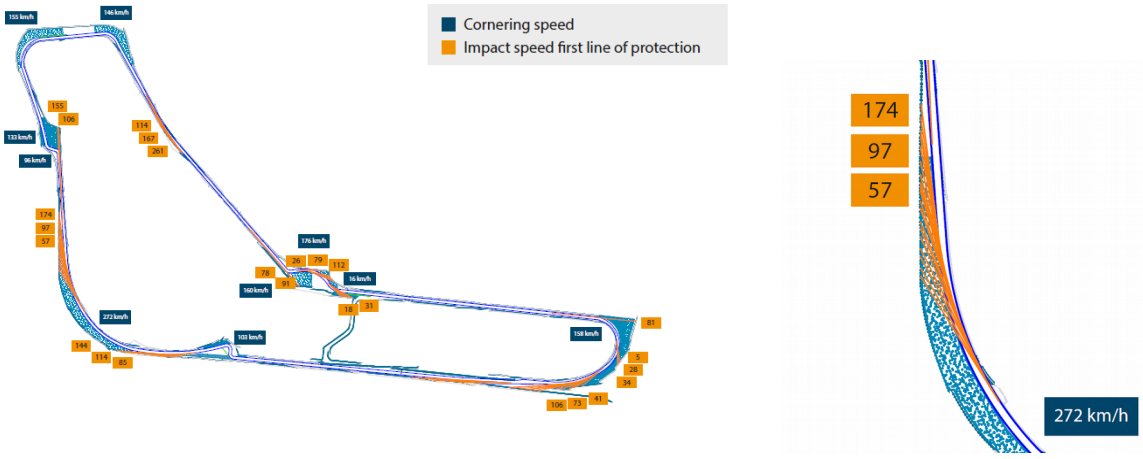


Ilustración 3: Ejemplo de un informe creado con el simulador, con detalle de una curva con varios impactos

Estos son los datos que queremos mostrar en la aplicación, por lo que conociendo las coordenadas del punto inicial del circuito y las coordenadas relativas de cada dato, podemos hacer que estas simulaciones solo se vean cuando el punto representado entre en el campo de visión del usuario.

Características de la aplicación

No todos los elementos presentes en los informes, o en las simulaciones, se han utilizado en la aplicación. La aplicación debía ser sencilla de usar para los usuarios y mostrar un resumen de los resultados de una simulación, modificarla o intervenir en el proceso no estaba contemplado. Además, gracias a la realidad aumentada, se pueden obviar una gran cantidad de datos topográficos del circuito.

Las trayectorias de escape eran el objetivo principal de este programa, y se utilizan sus coordenadas de inicio y fin para que su representación en el mundo virtual se superponga con el circuito real.

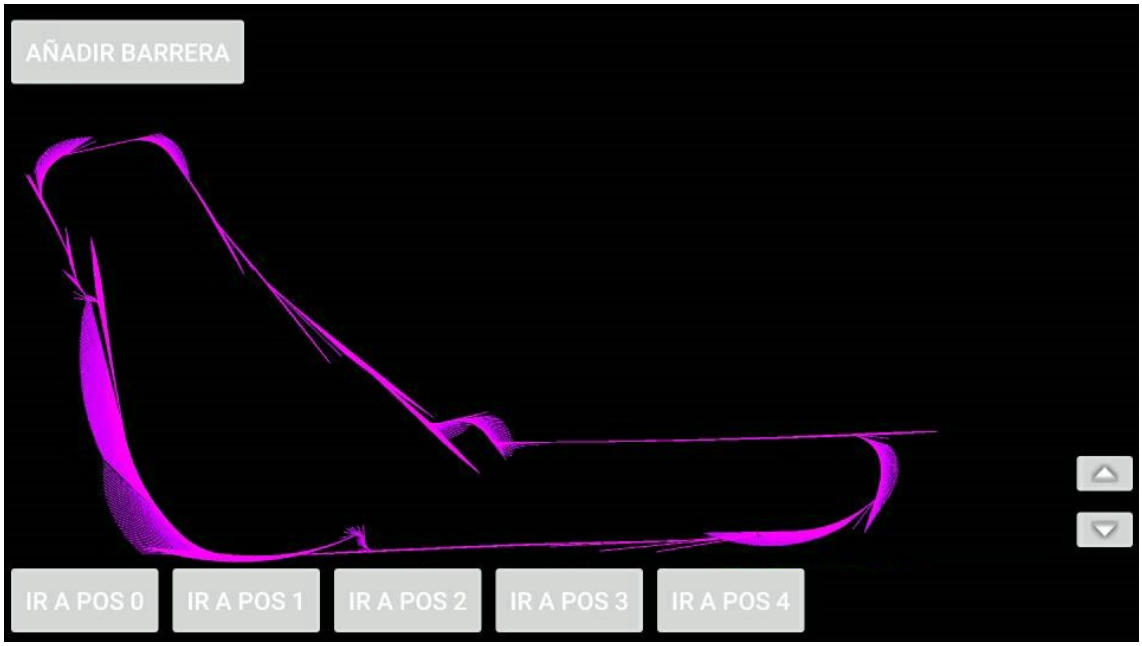


Ilustración 4: Vista aérea desde la aplicación del mismo circuito mostrado anteriormente.

Dado que se eliminan todos los datos del circuito, la necesidad de elegir que capas se representan se reduce, esto sumado a la reducida capacidad interactiva de los dispositivos *wearables*, llevo a la decisión de no incluir esta característica en el programa.

Por el mismo motivo se descarto la posibilidad de crear y modificar circuitos y simulaciones en la aplicación, con la excepción de los muros de contención, los cuales se pueden añadir usando la posición del usuario para marcar la ubicación de los mismos.

Puesto que los cambios en el circuito iban a ser menores, y dada la limitación, aunque cada día menor, de la potencia computacional de los *wearables*, también se descarto la posibilidad de realizar nuevas simulaciones desde la aplicación, por lo que esta obtiene los datos de las simulaciones ya realizadas.

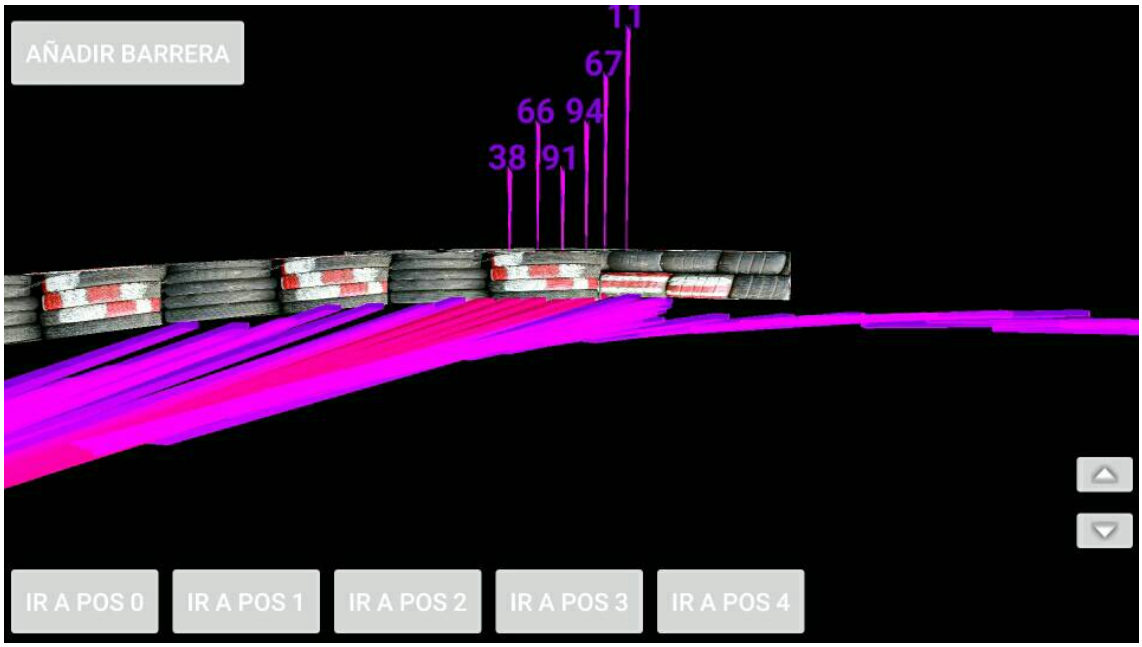


Ilustración 5: Ejemplo de una barrera añadida "in situ". Esta barrera implica un riesgo de impacto a diferentes velocidades que podría resultar letal.

Geoposicionamiento

Por supuesto, la precisión de este sistema depende de la del GPS del dispositivo en el que se muestra la aplicación, por ello su único propósito es el de mostrar los datos del informa, y no es de definir circuitos o realizar las simulaciones.

Las variaciones que se pueden realizar sobre el informe, esto es, añadir nuevas barreras de seguridad al circuito, se permiten con el objetivo de mostrar una aproximación a las consecuencias que tendría.

Al volver a un puesto de trabajo con el simulador, se comprobaría la posición de las barreras añadidas, se ajustaría si fuese necesario, y se realizarían las comprobaciones necesarias antes de llevar a cabo la construcción de dichos elementos.

Exportación de datos

Para poder ver los informes de seguridad desde la aplicación desarrollada, debe existir alguna forma de exportarlos desde el simulador.

En el simulador que se utilizo se realizaba la exportación de las trayectorias necesarias en ficheros .scr, que se pueden utilizar para generar el mapa del circuito en el programa de dibujo "autocad". Se decidió aprovechar este método para importar los datos en la aplicación, utilizando el sistema de ficheros de Android para separar los distintos circuitos en carpetas.

Estos se actualizarían en el dispositivo conectando con el servidor por Internet y descargando los nuevos circuitos generados previamente. La aplicación no interactuara directamente con el simulador, los circuitos deberán ser exportados a ficheros y las barreras creadas en la aplicación se importaran en el simulador de la misma manera.

No obstante, el simulador utilizado solo exportaba elementos visuales, como la posición de las trayectorias de escape o las curvas isocinéticas[1] de la velocidad tras una caída. Esta exportación servia para crear imágenes que mostrar en un informe, pero no se exportaba la velocidad a la que ocurrían las salidas de pista.

En la aplicación se necesitaba mostrar la velocidad a lo largo de la trayectoria, por lo que fue necesario modificar el programa para que se guardasen y enviasen al exportador los detalles de velocidad necesarios, así como las barreras incluidas en la simulación.

Se añadieron dos ficheros nuevos:

- Un fichero contenía varias lineas por cada "accidente", para complementar a la trayectoria, que incluía la velocidad en varios puntos a lo largo de esta. Esto permite mostrar mas detalles en cada trayectoria y calcular con mayor precisión la velocidad de impacto con las barreras añadidas desde la aplicación.
- El segundo fichero contiene las barreras o protecciones que existen actualmente en el circuito. Aunque la mayoría de datos del terreno no sean necesarios, dado el carácter de realidad aumentada del proyecto, es posible que se hayan añadido nuevas barreras al simulador, planificadas con anterioridad. Exportarlas permite facilitar la revisión del informe de seguridad del simulador desde el circuito inspeccionado.

[1] Las lineas isocinéticas unen todos los puntos del plano del circuito en el que la velocidad de escape es la misma. (P.E. una linea isocinetica uniría los puntos de todas las trayectorias que serian recorridos a 60 Km/h)

Realidad Aumentada

El concepto “Realidad aumentada”, también conocido como realidad mixta, hace referencia a la observación, directa o indirecta, del mundo real cuyos elementos han sido “aumentados” o modificados digitalmente. Esto engloba desde simples añadidos como el marcador en la retransmisión de los deportes, hasta gafas cuyos cristales nos muestran información de lo que tenemos delante.

Ilustración 6: Ejemplo de realidad aumentada

Si solo quisiéramos ver una representación tridimensional de la simulación, se crearía un mundo virtual en el que se colocaría una “cámara” que grabe imágenes en ese mundo y las muestre en la pantalla.

En este proyecto se necesita unir dicho mundo virtual con el real. Situar los objetos en sus coordenadas correspondientes es fácil, pero también hay que ajustar la cámara a la posición del usuario.

Para ello utilizan diferentes sensores, integrados hoy en día en la mayoría de teléfonos móviles, para detectar la posición y orientación de la pantalla, y se transmiten estos datos a la cámara virtual. Otro de los elementos mas importantes es el campo de visión, que define la amplitud del área que estamos viendo, y que debe ser exactamente el que abarca la pantalla real, para que lo que se dibuja al borde de la pantalla coincida exactamente con lo que se ve a través de la misma.

Esto supone un problema con los dispositivos con pantalla tradicional, ya que se necesita usar una cámara para ver lo que hay detrás, pero el campo de visión de esta no suele coincidir con la pantalla, por lo que algunos elementos se ven duplicados o distorsionados. Como en la mayoría de aplicaciones que utilizan realidad aumentada partiendo de las imágenes tomadas por una cámara en tiempo real, los mejores resultados se obtienen cuando el usuario se centra únicamente en lo que se muestra en la pantalla.

Además, la visualización tenia que servir tanto para dispositivos con pantallas tradicionales (led, lcd) como para pantallas "see-through", que permiten ver a través de ellas superponiendo su contenido.

Limitaciones de la realidad aumentada

La aplicación se ha diseñado para el propósito específico de visualizar el informe de una simulación. Aunque se podría haber creado una interacción mas profunda con el simulador, se consideró que esto podría reducir la usabilidad de la aplicación.

El pasar de un programa de escritorio a una aplicación Android significa cambiar el teclado y el ratón por un touchpad, o una pantalla táctil se se usa un smartphone con cámara. Estos medios de entrada son mas lentos y el usuario medio esta menos acostumbrado, lo que implica un aumento del tiempo en las interacciones con el programa.

Introducir datos de circuitos, configurar simulaciones o filtrar los datos que se quieren visualizar tomarían demasiado tiempo como para que la aplicación fuese de utilidad durante un informe de seguridad. Por ello se ha optado por un menú sencillo que nos permita cargar los resultados de un informe en unos segundos, y que sea al hacer el informe cuando se configuren los parámetros del mismo.

La realidad aumentada, aunque ayuda a ubicar las simulaciones sobre el terreno, también limita lo que podemos visualizar. Pasamos de una vista de pájaro del informe a vista de suelo, por lo que no podemos ver todo el circuito y los accidentes simulados de un vistazo.

Por supuesto hay muchos datos que se vuelven irrelevantes con la realidad aumentada.

No necesitamos ver la trazada del circuito en la pantalla, ni las zonas de frenado o las gradas del público, puesto que estamos viendo los auténticos.

El principal objetivo de la aplicación es ver donde hay peligro de accidentes en un circuito, por lo que lo primero que se muestra son las salidas de pista. Podemos centrarnos en una salida para ver detalles sobre la velocidad a la que se iniciaría, y su progresión hasta que el piloto y el vehículo se detienen. Estas marcas de velocidad se sitúan sobre la propia salida y se pueden volver a ocultar para no superponerse con las demás.

La frecuencia con la que se calculan estas salidas se puede configurar tanto en el simulador como en la aplicación. También la diferencia de velocidad necesaria para que se muestre la siguiente marca en una salida, o si estas deben colocarse a la misma altura o a diferentes, para poder ver todas las marcas desde cualquier ángulo.

Además de mostrar detalles sobre las salidas de pista, la aplicación también permite realizar modificaciones a la simulación, añadiendo barreras o protecciones al circuito, mostrando luego las posibles colisiones contra dicha barrera en caso de accidente.

Las barreras se colocan simplemente pulsando un botón, que sitúa el punto inicial de la barrera en la posición del usuario. Tras moverse a lo largo de la barrera que se desea colocar, se vuelve a pulsar el botón para crear una barrera entre la posición inicial y final del usuario, tras lo que se calculan y muestran las colisiones.

Por supuesto, la clave del proyecto es la visualización de la información clave para evitar accidentes, por lo que si algún elemento de la simulación molestase, se pueden eliminar de la visualización. Si se ha hecho una simulación de accidente cada pocos metros, pero en un momento dado nos queremos centrar solo en una salida de pista que podría dar lugar a un accidente, se pueden ocultar las salidas cercanas simplemente pulsando en ellas, dejando una vista mas clara de la que realmente interesa.

Motor gráfico

Para facilitar la integración de la realidad aumentada, se opto por buscar y usar una librería de desarrollo de aplicaciones con RA. Esta debía permitir el uso de geolocalizacion para unir la representación virtual con la visión del mundo real, además de reconocer la interacción del usuario con los objetos virtuales. También era recomendable que la librería utilizase una licencia gratuita y código abierto.

La mayoría de librerías de realidad aumentada que se podían encontrar a mediados del 2015 se centran en el uso de marcadores (DART[2], AndAR[3]), o en el reconocimiento de objetos (windage[4], wikitude[5]) para alinear las imágenes virtuales al mundo real. La "baja" precisión del GPS (5-25m) había relegado su uso en la realidad aumentada a aplicaciones de navegación o de información sobre puntos de interés (restaurantes, museos, monumentos…).

	Type	Unity (3D)	Marker	3D Object Tracking	GPS	IMU Sensors	Visual Search	ContentAPI
AndAR	Free		✓					
DroidAR	Free + Commercial SDK option		✓	✓	✓	✓	⊖ can be added via opencv	✓ OpenGL or jMonkey Engine
Wikitude	Free + Commercial SDK option	✓ 3D Tracking Included	✓ Advanced	✓ Beta	✓	✓	✓ Cloud Recognition and Offline (on device)	✓ with Wikitude Studio and Cloud Recognition
windage	Other		✓					
Designers ARToolkit (DART)	Free		✓					✓

7- Comparativa de las librerías mencionadas

Pero aunque no fuesen muy populares, si existían algunas librerías de RA basadas en GPS, y algunas de las anteriores también incluían integración para ello.

Junto con el GPS, es necesario el uso de los sensores de orientación del dispositivo en el que se esta ejecutando la aplicación. Las aplicaciones de realidad aumentada utilizan estos sensores (giroscopio, acelerómetro…) para determinar la orientación y el movimiento del usuario y poder calcular que se debe mostrar en la pantalla.

[2] DART - Web de la librería: http://ael.gatech.edu/dart/aboutdart.htm
[3] AndAR - Proyecto en Google Code: https://code.google.com/archive/p/andar/
[4] Windage - Proyecto en Google Code: https://code.google.com/archive/p/windage/
[5] Wikitude - http://www.wikitude.com/

DroidAR[6], Arlab[7], Vuforia y Wikitude son algunas de las librerías mas completas y conocidas, y todas facilitan el uso del GPS y de los sensores del dispositivo.

Vamos a ver un poco mas sobre estas 4 librerías.

- **DroidAR:** Es una librería centrada en la navegación y la interacción con el usuario. Permite crear objetos situados en unas coordenadas geográficas especificas y que el usuario pueda ver el objeto si se encuentra lo bastante cerca. También permite la interacción con dichos objetos, tanto táctil como por posición, y tiene código abierto, por lo que se puede modificar para justarse a las necesidades del proyecto.
- **Arlab:** Arlab es un conjunto de librerías, apoyado por la Empresa Nacional de Innovación (EINSA) que ofrece diversas soluciones para proyectos de realidad aumentada. La que nos interesaría en este proyecto es la librería **AR BROWSER**, que promete facilitar la creación de aplicaciones para navegación por ciudades. Aunque esta mas orientada hacia puntos de interés (POI) que pura geolocalizacion.
- **Vuforia**: Vuforia esta centrada en el reconocimiento de imágenes y marcadores. Como también integra localización por GPS, se planteo la posibilidad de usar Vuforia para reconocer la trazada del circuito y mejorar la precisión del GPS al mostrar las salidas de pista.
- Wikitude: Wikitude es una librería centrada en aplicaciones para “smartglasses” como las Google Glass o las ORA-1. Puesto que la aplicación se va a usar principalmente en este ultimo, Wikitude era una solida opción. Facilita la interacción del usuario con los elementos del entorno y la visualización en pantallas transparentes. Se descarto por estar mas centrada en reconocimiento de objetos mediante tecnologías cloud, el uso de la geolocalizacion es principalmente para obtener lugares de interés cercanos e información de la zona.

DroidAR

DroidAR, desarrollado por bitstars, es un framework para realidad aumentada en Android. Esta preparado tanto para la realidad aumentada basada en marcadores como para la basada en localización *(Ver Ilustración 2)*.

[6] DroidAR - http://droidar.blogspot.com./
[7] ARlab - http://www.arlab.com/arbrowser

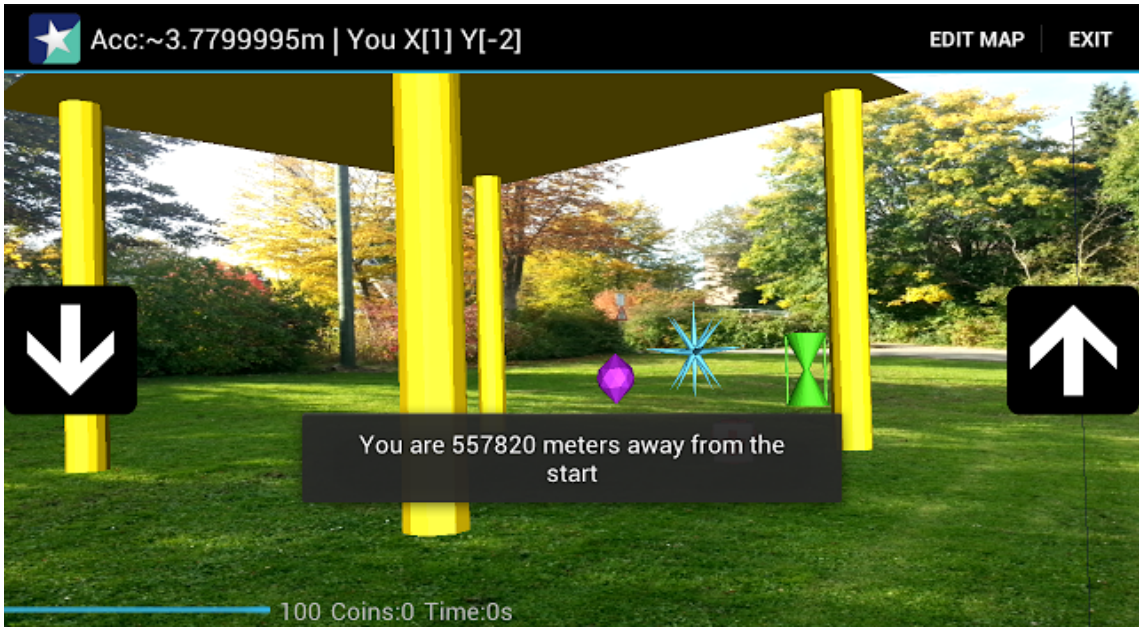


Ilustración 8 : Ejemplo de realidad aumentada basada en localización.

Encapsula la librería gráfica OpenGL2 añadiendo elementos que se pueden ubicar en base a coordenadas geográficas, o relativos al usuario. También se encarga de sincronizar la cámara virtual de OpenGl con la pantalla del dispositivo para dar la sensación de que lo que se esta viendo en la pantalla, realmente esta detrás de la pantalla. Facilita la carga de texturas para dar mas detalles y realismo a los objetos.

También permite generar texturas a partir de vistas, de las utilizadas para cualquiera de las aplicaciones tradicionales, para mostrar textos, botones, o cualquier otro elemento en el mundo virtual. Técnicamente se podría cargar la funcionalidad completa de otra aplicación Android en el mundo virtual, e interactuar con ella.

El mayor problema encontrado con esta librería fue en la integración con la aplicación. DroidAR esta preparada para el entorno de desarrollo, o IDE, Eclipse, y implementar la aplicación Android en ese mismo entorno. Para cuando se comenzó a realizar este proyecto, Android había lanzado su propio IDE, Android Studio, y descatalogado el Plug-in de Android para Eclipse. Puesto que el proyecto se centra en la interfaz de realidad aumentada, se decidió utilizar Eclipse sacrificando las facilidades para diseño de interfaces de Android Studio para evitar la migración de la librería DroidAR.

Descripción Y Desarrollo De La Aplicación

Una vez expuestos los sistemas de los que partía este proyecto, se va a detallar la aplicación que se ha implementado.

Como se indicaba en la introducción, el objetivo de este proyecto era crear una aplicación que, mediante realidad aumentada, representase una simulación de accidentes en una carrera de motociclismo, proyectados sobre el propio circuito del que se ha hecho el informe.

Concretamente, se quería representar las trayectorias de escape de un piloto y su vehículo tras un accidente, o una caída, en una competición de motociclismo.

Esta aplicación nos permitirá situarnos físicamente en el circuito y ver, mediante una pantalla semitransparente, una linea a lo largo de la trayectoria que seguiría un piloto que sufriera un accidente a nuestros pies. También nos permitirá ver detalles sobre la velocidad que tendría el piloto en el punto inicial del accidente, y la progresión de la misma hasta que se detiene por completo, o colisiona contra una barrera.

Otro requisito era poder realizar modificaciones al circuito. Cuando un inspector este revisando el informe de seguridad, con la ayuda de esta aplicación, para comprobar donde se deben mover las protecciones o donde se pueden colocar nuevas. Por tanto la aplicación debía permitir añadir estas protecciones al circuito, calcular si habrá colisiones, y en caso afirmativo estimar la velocidad del impacto. Esto permitirá agilizar las modificaciones al circuito, puesto que ya no se requiere el uso de un ordenador para mostrar una estimación del riesgo.

Funcionamiento del programa

Al iniciar la aplicación, se muestra al usuario la lista de circuitos disponibles. En esta vista también hay un menú contextual que permite al usuario realizar varias acciones:

- Activar o desactivar la cámara. En dispositivos con pantalla opaca, también se puede conseguir un efecto de realidad aumentada utilizando la cámara del dispositivo, aunque si el área de visión de la cámara no coincide con el área cubierta por la pantalla el efecto se vera distorsionado.

- Cambiar el campo de visión (FOV). Por defecto es 22º, que se corresponde con la pantalla de las gafas ORA-1.
- Actualizar. Enviar una petición al servidor para comprobar si hay circuitos nuevos, y en tal caso, descargarlos.
- Sincronizar. Conectar con el servidor y enviar las barreras creadas en el circuito seleccionado.

Todas las opciones se guardan utilizando la API Preferences de Android, que 0 permite crear y recuperar registros clave/valor dejando al sistema operativo encargarse de su almacenamiento.

Por el contrario, los circuitos se almacenan como ficheros en el almacenamiento externo del dispositivo, para facilitar su acceso en caso de querer cargar los circuitos sin depender de la conexión a Internet o de la propia aplicación desarrollada.

Tras elegir el circuito, se lanzan varios hilos de ejecución para agilizar la carga del programa. En primer lugar se crea el mundo virtual de OpenGl, en el que se cargaran las trayectorias que queremos visualizar y que nos permitirá visualizar estos objetos en la pantalla. Por otro lado, se inicia la lectura de los ficheros que contienen los informes del circuito seleccionado, y se crean los objetos tridimensionales que representaran las trayectorias de los accidentes y las barreras del circuito.

El ultimo proceso necesario, y el que mas tiempo lleva, es la inicialización del GPS. De esto se encarga el propio Android del dispositivo, por lo que este proceso solo debe esperar hasta que se haya alcanzado una precisión suficiente y entonces permite continuar a la aplicación.

Cuando estos procesos hayan acabado, la pantalla del dispositivo se comportara como una ventana a este mundo virtual. Gracias al GPS conocemos la ubicación del usuario, por lo que todos los objetos se han colocado respecto a este de tal forma que solo se vean cuando el usuario este en las coordenadas correctas. Esto significa que si estamos en el circuito y orientamos la pantalla hacia él, podemos ver las trayectorias de los accidentes exactamente en la posición que les corresponde.

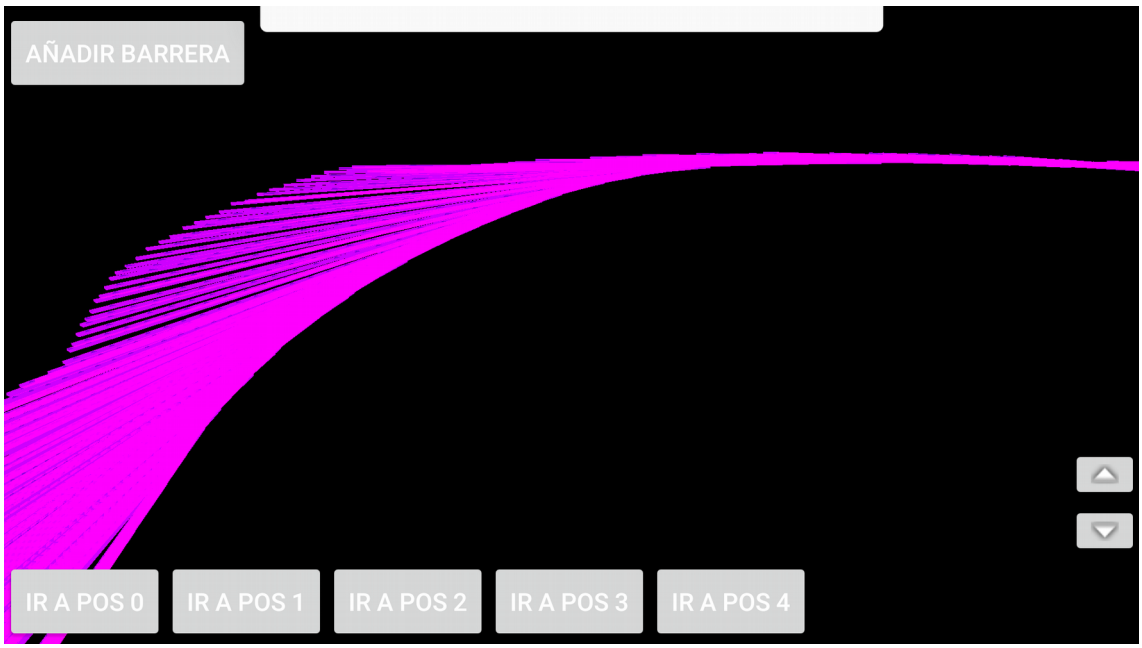


Ilustración 9: Ejemplo de una curva abierta, sin velocidades.

Para mantener la pantalla lo mas limpia posible, por defecto solo se muestran las lineas que delimitan las trayectorias y las barreras. En caso de haber colisiones, también se muestra la velocidad a la que se produce con un texto flotante sobre el punto de colisión. Un detalle que se puede apreciar es que no se muestran las unidades, esta decisión se tomo para simplificar lo que se veía en la pantalla, puesto que si los textos se solapan dificultan su lectura. Ademas, la aplicación esta pensada para usuarios que conocen el simulador, el circuito y el informe, por lo que saben que los datos están representados en kilómetros por hora, lo que permite evitar la redundancia de esta información.

En caso de que el usuario quisiera ver detalles sobre una salida de pista en concreto, este puede pulsar sobre ella, lo que resaltara esta trayectoria y mostrara, con textos flotantes, la velocidad a lo largo de varios puntos, tal y como se haya configurado en el simulador al crear el informe. Una vez haya terminado, puede volver a pulsar sobre la trayectoria para ocultarlos. Ademas, si quisiera centrarse sobre una única trayectoria y le molestasen las demás, puede realizar una pulsación larga sobre la misma, lo que hará que desaparezca junto con los detalles que pudiese estar representando.

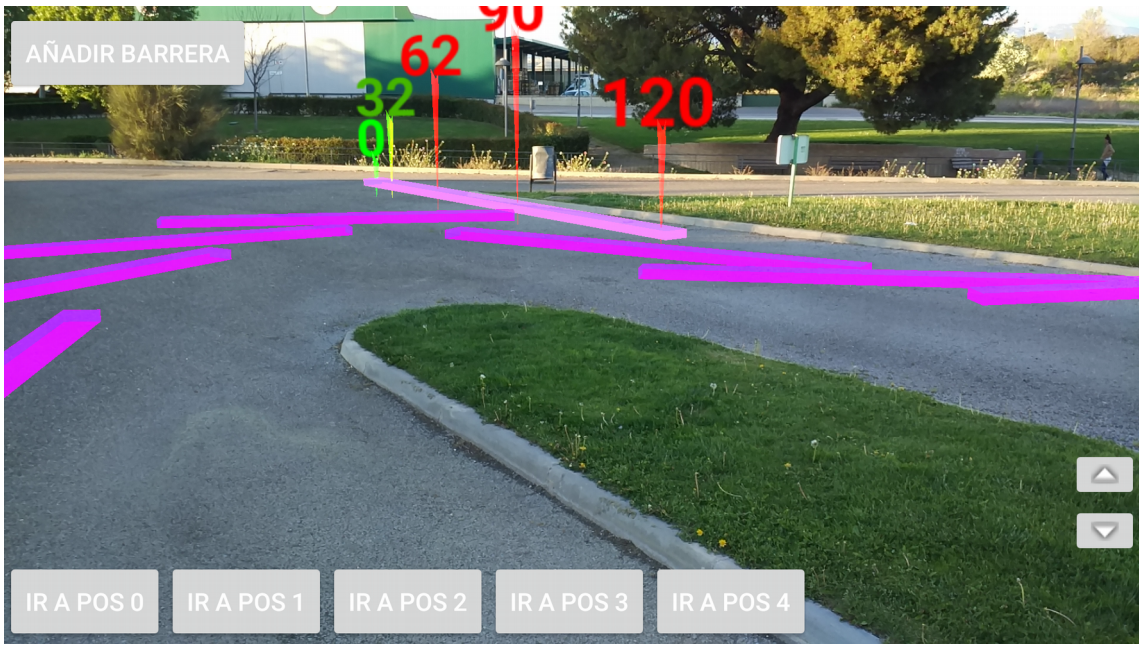


Ilustración 10: Ejemplo de una curva cerrada, la última de un slalon. Se muestran las velocidades de la trayectoria seleccionada.

La aplicación debía permitir al usuario añadir barreras, o protecciones, al circuito, incluyendo estas en la simulación y mostrando si se detectaba riesgo de impacto.

Para crear las barreras, se utiliza la posición del usuario como punto de partida. Pulsando un botón que se muestra en la pantalla, se guardan las coordenadas actuales del dispositivo como punto inicial de la barrera. Cada pulsación al botón añade un nuevo punto, por lo que para crear una barrera solo es necesario recorrer la zona que esta ocupara, marcando los puntos que se consideren necesarios. Generalmente bastara con añadir dos o tres tramos para crear una barrera que se adapte lo suficiente para la simulación.

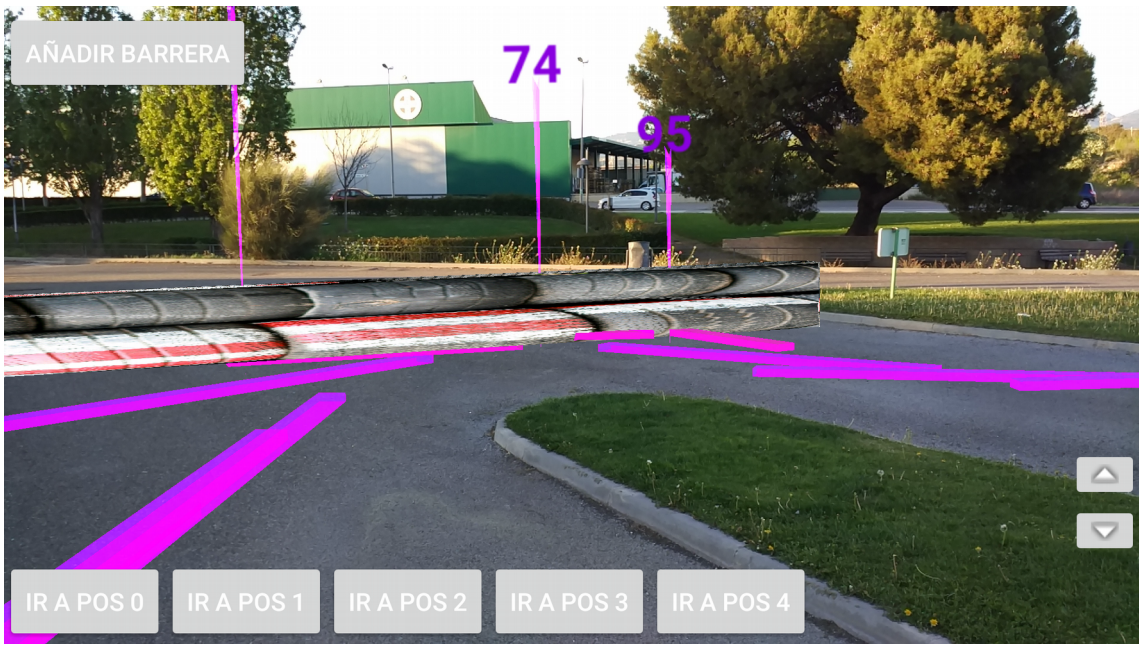


Ilustración 11: Ejemplo de una barrera recién añadida, situada al borde del circuito, por lo que implica colisiones.

Entonces el programa representara la barrera en la pantalla y calculara si intercepta alguna de las trayectorias existentes. En caso de impacto, se mostrara sobre la barrera un texto con la velocidad a la que se estima que se produciría el choque.

Determinar si este impacto supone un riesgo o no queda fuera de lo que se buscaba en este proyecto, puesto que entran en juego mas factores, incluyendo el error que pueda introducir el propio GPS al crear la barrera. El objetivo es simplemente mostrar una primera impresión de lo que la nueva barrera podría suponer.

Metodología

Interacción con DroidAR

La integración de la librería DroidAR se centra en una sola clase que extiende de "*Setup*". La clase abstracta *Setup* de DroidAR es una clase abstracta que funciona como una plantilla[8]. La implementación base crea una instancia de OpenGL, añade una cámara, inicializa un gestor de eventos para interactuar con el mundo virtual y prepara la vista de OpenGL para mostrarla en la pantalla. También crea varias capas, una bajo la de OpenGL para la cámara, y tantas como sean necesarias para la interfaz de usuario sobre ella. Cuando ha terminado, el entorno OpenGL creado se muestra en la *Activity* en la que se haya creado el *Setup*, junto con las demas capas que se hayan configurado y con los eventos activados a la espera de interacción por parte del usuario.

Tras cada paso, se realiza una llamada a un método abstracto, que permite modificar el mundo virtual que se esta creando para ajustarlo a nuestras necesidades. Cada uno de estos métodos provee los parámetros necesarios para realizar una tarea distinta, correspondiente a la fase en la que es invocado.

Por ejemplo, el primer método se ejecuta nada mas crear la clase, no tiene parámetros y esta pensado para inicializar lo que podamos necesitar en nuestra implementación. Aquí se crea un *World, un* objeto que representan un mundo virtual y contendrá toda la información de la escena que queremos mostrar en pantalla. También se crea una cámara y una serie de eventos que detectaran el movimiento del dispositivo.

En las fases intermedias de la inicialización, se crean las instrucciones a ejecutar cuando se activen los eventos, se lanza un hilo que ejecutara un *observador*[9] para todos los eventos, puesto que no son eventos convencionales de Android y no podemos delegar en él para gestionarlos.

Estos métodos son lo mínimo que se debe implementar para utilizar la librería DroidAR, pero no se ha mencionado como se añaden las trayectorias, barreras y velocidades que se quieren mostrar en este proyecto. Además de los ya mencionados, hay varios métodos que se pueden sobrescribir, valiéndose del funcionamiento de la herencia de Java, que nos permiten personalizar aún mas el mundo virtual que estamos creando.
En uno de estos métodos se define la iluminación global a utilizar en la escena, puesto que los elementos a representar tienen el único objetivo de mostrar información, la mejor opción es una luz homogénea, que ilumine todos los elementos por igual sin causar sombras.

---

[8] El patrón de método de la plantilla es un patrón de diseño de comportamiento que define el esqueleto de programa de un algoritmo en un método, llamado método de plantilla, el cual difiere algunos pasos a las subclases.

[9] El patrón de diseño Observador define una dependencia del tipo 1-n entre objetos, en nuestro caso eventos, de manera que cuando uno se active se procesen todas las funciones asociadas a él.

El último método debe crear y añadir a la escena los objetos que se quieren representar, no obstante en este proyecto se modifica ligeramente el comportamiento. Puesto que la ubicación de los objetos depende de la posición y orientación del usuario, los únicos elementos que se pueden añadir en esta fase son los botones inmóviles que se ven en la pantalla.

Para añadir los elementos del informe, en la fase de creación del mundo se ejecuta una función que, de forma simultanea a la creación de la escena, activa el GPS del dispositivo y comienza a leer la ubicación. El GPS de un dispositivo Android, una vez activado, busca satélites a los que conectarse y con una llamada al sistema permite conocer la localización. Esta localización contiene la ordenada, la coordenada, la altitud y una estimación de la precisión de la medida.

La función que se ha creado solicita la ubicación al sistema cada cierto tiempo, y solo cuando la precisión ha alcanzado un valor aceptable y la creación del entorno OpenGL ha concluido, crea los objetos que representaran las barreras y las trayectorias y los añade a la escena.

Para gestionar la creación de objetos que representasen los elementos del informe, se implemento una *Factoria*, una clase que implementa todos los métodos necesarios para crear objetos renderizables en una escena a partir de los que contienen los datos del informe, que consisten principalmente en contenedores de coordenadas. Para las trayectorias se crean hexaedros rectangulares, de pequeña anchura y menor altura, ya que daban un mejor resultado en pantalla que los rectángulos bidimensionales. Estos se sitúan con una minúscula diferencia de altura para evitar la superposición. Como barreras se usan hexaedros mas grandes, de 2m de altura y texturizados para representar una típica barrera de neumáticos.

Todos los objetos de la escena incorporan al Observer una función a ejecutar cuando el usuario pulsa sobre ellos, permitiendo mostrar u ocultar la velocidad del piloto a lo largo de una trayectoria y eliminar las trayectorias o barreras que desee.

En la sección de realidad aumentada se ha comentado que uno de los aspectos que se buscaban era la disponibilidad del código fuente de la librería, y fue en esta parte de la implementación donde esa precaución se convirtió en una necesidad, puesto que DroidAR añade la imagen de la cámara bajo la imagen de OpenGl siempre que el dispositivo tenga una disponible, lo cual suponía un problema con las smartGlasses, puesto que el área de visión de la cámara no coincide con el área que hay tras la pantalla. La cámara debía ser desactivada en estos casos, puesto que el resultado era confuso y podía llegar a producir mareos al usuario.

Carga de ficheros y conexión con el simulador

Como se ha mencionado, el almacenamiento de los datos de los circuitos se hace en el propio dispositivo, mediante el sistema de ficheros de Android.

Se designa o crea un directorio para la aplicación, y un subdirectorio por cada circuito. Por cada circuito se guardan los diferentes ficheros exportados desde el simulador.

Actualmente se exportan hasta 5 ficheros por circuito, dos pares de ficheros con las coordenadas y las velocidades de las lineas de escape del piloto y el vehículo, y otro con las coordenadas de las barreras. Estos ficheros no son obligatorios, si para un circuito no hay barreras, datos de velocidad o trayectorias de piloto o vehículo, simplemente se mostraran los datos que existan.

Para actualizar estos ficheros, se envía una petición http a un servicio REST conectado al simulador. En la petición que envía la aplicación se incluye la fecha de la última actualización como un parámetro de la url. La respuesta debe ser otro JSON con los datos de todos los circuitos creados o actualizados a partir de la fecha enviada. En el anexo 2 se ha incluido el esquema de definición de este JSON.

Aunque se implementó un sencillo webservice en php para un servidor apache, con el objetivo de testear la sincronización durante el desarrollo, este simplemente lee los ficheros de una carpeta designada y los envía. Queda a la responsabilidad de quien gestione el simulador desarrollar un webservice REST que cumpla con las especificaciones.

Conclusiones

Se ha creado una aplicación que, ejecutada en un dispositivo como las smartglasses ORA-1, nos permite recorrer un circuito de motociclismo mientras se muestran los resultados de accidentes que podrían suceder a nuestros pies.

Las trayectorias se muestran de forma clara y distinguible del entorno, y con un simple toque podemos ver a que velocidad atravesaría nuestra posición un piloto accidentado, donde se detendría y/o a que velocidad chocaría contra la pared. O simplemente nos puede asegurar que los asistentes que se sitúan en la primera fila están a varios metros del punto máximo que podría alcanzar un piloto que no frenase a tiempo.

El publico en general esta mucho mas acostumbrado a los mapas y los informes en papel tradicionales que a la realidad aumentada, las reacciones han sido positivas en la mayoría de usuarios que han probado la aplicación, aunque todavía no se ha puesto a prueba por ningún usuario potencial, que fuese a usar la aplicación para análisis de seguridad en circuitos. Si que se realizaron pruebas con usuarios ajenos al entorno de motociclismo, algunos de los cuales son los desarrolladores del simulador utilizado para los circuitos de ejemplo.

Cronograma

Al comenzar a implementar el proyecto en el verano de 2015, tras acabar el cuarto año de carrera y con dos asignaturas pendientes de aprobar, detuve el desarrollo para centrarme en la convocatoria de septiembre. Tras esto, comencé a trabajar como becario en Tap Consultoría y, aunque retome el proyecto nada mas acabar exámenes, pronto paso a segundo plano. En el siguiente cronograma no se reflejan avances desde noviembre de 2015 hasta octubre de 2016, aunque si hubo algún progreso, en ningún momento fue constante, y se me hace demasiado difícil concretar que se logro en ese tiempo. Algo similar pasa en diciembre de 2016, tras no completar esta memoria a tiempo para la ultima convocatoria de 2016, una vez mas el proyecto paso a segundo plano, esta vez solo un mes, retomando y completando esta memoria a finales de enero de 2017.

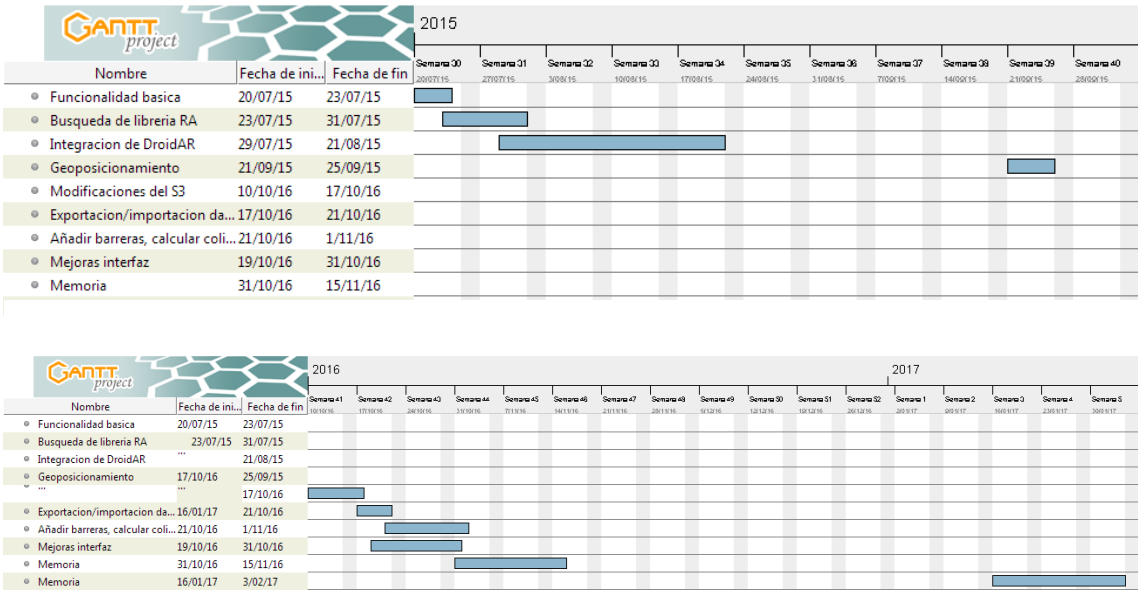


Ilustración 2: Diagrama de gantt: 2016-2017

Opinión personal

A nivel personal, he tenido la oportunidad de trabajar con una tecnología en desarrollo como es la realidad aumentada. He expandido mi conocimiento de Android, de su sistema de archivos y la configuración de aplicaciones. Pero quizá la experiencia mas importante sea el proceso de búsqueda y uso de un código ajeno. Investigar las opciones existentes para averiguar si cubre las necesidades del proyecto y, sobre todo, descubrir por mi cuenta el funcionamiento de un código desconocido.

Aunque disponía de documentación acerca del funcionamiento de DroidAR, como crear un proyecto y como utilizar las diferentes funciones, parece que los desarrolladores tenemos tendencia a obviar en la documentación funciones, opciones o pequeños detalles que nos parece innecesario mencionar, pero que a alguien que desconoce nuestro trabajo puede resultarle confuso. Tras pasar varias horas siguiendo un hilo de ejecución por decenas de clases, todas desconocidas, para tratar de averiguar porque algo tan sencillo como dibujar una linea no funciona correctamente en determinadas condiciones se comprende la gran importancia que tiene documentar nuestro trabajo correctamente.

También tuve la oportunidad, o mas bien necesidad, de adentrarme en el entorno de desarrollo Eclipse, el cual había utilizado de una forma mucho mas simple, puesto que los proyectos que había necesitado hasta ahora como máximo necesitaban incluir alguna librería en forma de .jar. Gracias a este proyecto he podido conocer las diferentes formas de trabajar con diferentes proyectos a la vez, ya sea importando el código y compilando como parte de un mismo proyecto o importando una librería externa generada específicamente para asociarla con una aplicación Android.

Por otra parte, también he aprendido el esfuerzo que supone trabajar en un proyecto después de una jornada completa de trabajo, y la facilidad que se quede relegado a un segundo plano. Después de casi un año sin volver al proyecto, también comprendí por las malas la importancia de documentar todo, desde el trabajo realizado, hasta la información obtenida, en ocasiones incluyendo porque era necesaria.

Bibliografía

[1] Documentación oficial de Android (https://developer.android.com/index.html)

[2] Documentación y foros de las smart glasses Ora-1 (http://optinvent.proboards.com/thread/20/ora-1-faq)

[3] Proyecto DroidAR, de Bitstars (http://bitstars.github.io/droidar/)

[4] Documentación para OpenGL en Android (https://developer.android.com/training/graphics/opengl/index.html)

[5] Understanding SOAP and REST Basics and Differences (http://blog.smartbear.com/apis/understanding-soap-and-rest-basics/)

[6] Augmented Reality SDKS (http://augmera.com/?p=461)

[7] Android SDK Augmented Reality: Location & Distance (https://code.tutsplus.com/tutorials/android-sdk-augmented-reality-location-distance--mobile-8004)

[8] Michele Gattullo, Antonio E. Uva, Michele Fiorentino, Joseph L. Gabbard (2015). Legibility in Industrial AR: Text Style, Color Coding, and Illuminance)

[9] Historia de la realidad aumentada http://www.augment.com/blog/infographic-lengthy-history-augmented-reality/

[10] Informacion sobre wearables y SmartGlasses https://www.wareable.com/

[11] Foros de consultas de programación StackOverflow (http://stackoverflow.com/)

[12] Wikipedia, la enciclopedia online (en.wikipedia.org)

ANEXO 1

CASOS DE USO

Visualización del informe de un circuito

#	USUARIO	SISTEMA
1	Inicia la aplicación	
2		Se carga la lista de circuitos y las opciones
3	Selecciona el circuito	
4		Se leen los datos del circuito, se activa el GPS y se crea el entorno OpenGL
5		El sistema sitúa al usuario como centro del mundo virtual y coloca los objetos que representan las trayectorias a su alrededor, convirtiendo las coordenadas geográficas en coordenadas virtuales según la posición del usuario.
6	El usuario se sitúa físicamente en el circuito y orienta el dispositivo hacia la carretera.	
7		Cuando una barrera o trayectoria entra en el área que cubre la pantalla del dispositivo, ésta aparece en la pantalla, de tal forma que el usuario la ve en sus coordenadas.
8	El usuario pulsa sobre la trayectoria deseada para conocer la velocidad a la que se produce la caída.	
9	Cuando ha terminado, cierra la aplicación.	

Creación de una barrera

Se parte del paso 6 del caso de uso anterior, cuando el usuario esta visualizando un circuito.

#	Usuario	Sistema
1	El usuario se sitúa en el punto donde desea comenzar la barrera	
2	El usuario pulsa el botón “Crear barrera”	
3		El sistema coloca un marcador en la posición del usuario.
4	El usuario camina a lo largo de la barrera. Cuando ha recorrido un tramo recto, pulsa el botón “Añadir tramo”	
5		El sistema crea una barrera desde el marcador hasta la posicion del usuario.
6		El sistema mueve el marcador hasta la posición del usuario.
7	Repite los pasos 4-6 hasta que ha completado la barrera.	
8	El usuario pulsa el botón “Finalizar barrera”	
9		El sistema guarda la barrera y calcula las colisiones.
10		El sistema crea y muestra marcadores que indican las colisiones y la velocidad a la que se producen.
11	El usuario puede seguir inspeccionando el circuito	

ANEXO 2:

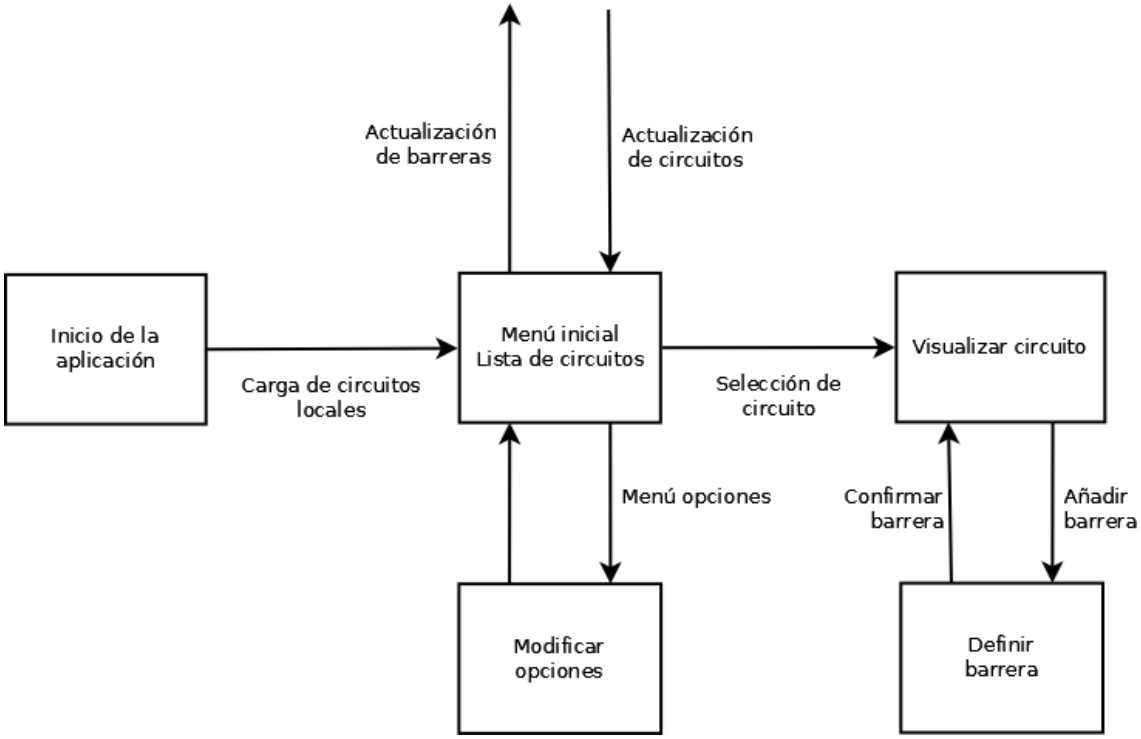
Definición del JSON utilizado en la comunicación con el simulador.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "ListaCircuitos",
    "description": "Una lista de circuitos actualizados",
     "type": "array"
     "items": {
         "title": "Circuito",
         "type": "object",
         "properties": {
              "nombre": {
                 "description": "El nombre unico del circuito",
                 "type": "String"
                 },
              "salidasMoto":{
                 "description":    "Contenido    del    fichero
salidasMoto_enteras.scr",
                 "type": "String"
                 },
              "velocidadMoto": {
                 "description":    "Contenido    del    fichero
velicidadFranjasSalidasMoto_enteras.txt",
                 "type": "String"
                 },
              "salidasPiloto":{
                 "description":    "Contenido    del    fichero
salidasPiloto_enteras.scr",
                 "type": "String"
                 },
```

```
            "velocidadPiloto": {
              "description": "Contenido del fichero velicidadFranjasSalidasPiloto_enteras.txt",
              "type": "String"
              },
            "barreras": {
              "description": "Contenido del fichero barreras.scr",
              "type": "String"
              }
        }
     "required": ["nombre"]
    },
    "required": ["circuitos"]
}
```

ANEXO 3

Diagrama de bloques funcionales



Glosario:

- **Activity:** De la guia de Android: *Una Activity es un componente de la aplicación que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción, como marcar un número telefónico, tomar una foto, enviar un correo electrónico o ver un mapa.*
  En este proyecto se usan dos, una que conforma el menú y otra para la visualización de circuitos.