TRABAJO FIN DE GRADO

Grado en Ingeniería de Sistemas Audiovisuales

Realidad Aumentada en dispositivos iOS como sistema de ayuda a los alumnos

Autor: Diego Villarán Molina
Tutor: Carlos García Rubio
Director: Jorge Ruiz Magaña

Universidad Carlos III de Madrid
www.uc3m.es

TÍTULO: *REALIDAD AUMENTADA EN DISPOSITIVOS iOS COMO SISTEMA DE AYUDA A LOS ALUMNOS*

AUTOR: *DIEGO VILLARÁN MOLINA*

TUTOR: *CARLOS GARCÍA RUBIO*

DIRECTOR: *JORGE RUIZ MAGAÑA*

(Hoja en blanco)

Agradecimientos

Aquí termina una aventura que empezó hace cinco años. Cinco años, que parecen haber sido cinco días, en los que muchas personas han contribuido a que esta haya sido la mejor etapa de mi vida.

Gracias a mi familia, en especial a mis padres Luis y Carmen; gracias porque desde la distancia habéis sido partícipes tanto de mis buenos como de mis malos momentos. Siempre habéis estado ahí aguantándome y siempre me habéis dado todo vuestro apoyo.

Gracias a mis amigos de Miranda, los de toda la vida: Asier, Míkel, Gustavo, Miguel, Paloma, Sandra, Adriana, Nerea y todas las demás; gracias porque desde pequeños habéis hecho que en gran medida sea lo que soy.

Gracias a todas las personas que habéis formado parte de mi vida estos años en Madrid. A la gente que he conocido en la Universidad; toda, la que de algún modo ha sido parte de esta aventura. A los compañeros de clase, tanto de la Técnica como del Grado: Paloma, María, Víctor, Sergio, Miguel Ángel, Kike, Nacho… (perdonad si me dejo a alguien), gracias. Gracias a Jaime, Vero y Marta; amigos con los que he pasado muy buenos ratos y alguno que otro no tanto… A mis compañeros y amigos de "vida" Edu y Diego, habéis sido mi familia aquí, gracias a vosotros me he sentido como en casa. Especialmente a ti, Edu; muchas gracias por todo. A mis chicas: María, Natalia, Belén y Susana; habéis sido y sois un pilar muy importante para mí; hemos pasado juntos muy buenos e inolvidables momentos y espero seguir añadiendo más de ellos a la lista… Con vosotras y el resto del Ghetto, por supuesto; muchas gracias por haberme hecho uno más.

También me gustaría agradecer a mis compañeros de RNE, gracias a vosotros he aprendido mucho y habéis hecho que este verano haya sido mucho más llevadero. Os voy a echar de menos.

Por último, agradecer a mis tutores Jorge y Carlos el haberme permitido realizar este Proyecto y ayudarme con la Memoria.

De nuevo, gracias a todos.

(Hoja de blanco)

Resumen

Este Proyecto consiste en la realización de una aplicación destinada a los alumnos de la Universidad para obtener ayuda e información del Campus Universitario. La aplicación se ejecuta en dispositivos móviles iOS: iPhone, iPad e iPod Touch y emplea tecnologías de Realidad Aumentada proporcionadas por el SDK Vuforia de Qualcomm.

Esta aplicación servirá como nexo de unión entre soportes, a priori incompatibles, como una simple hoja de papel impresa y el propio dispositivo electrónico haciendo uso de Realidad Aumentada; que proporcionará al soporte impreso de contenidos multimedia e interacción a través de la pantalla del dispositivo móvil.

Para llevar a cabo este Proyecto se estudian las diferentes herramientas que se necesitan para poder desarrollar una aplicación para la plataforma de Apple, así como adquirir los conocimientos teóricos necesarios a tal efecto.

Además se hace un estudio sobre cuál es el estado actual de la Realidad Aumentada y de las herramientas capaces de proporcionarla. Se utilizará una de estas tecnologías, Vuforia, para dotar a la aplicación a desarrollar de capacidades de Realidad Aumentada y se explica con detalle cómo es el funcionamiento y arquitectura de esta tecnología.

Se estudiarán técnicas de modelado en tres dimensiones para incorporar Realidades Virtuales propias a la aplicación a desarrollar y se expone con detalle cómo ha sido el proceso de desarrollo de la aplicación, así como su funcionamiento.

Palabras clave

Smartphone, Realidad Aumentada, Realidad Virtual, Apple, iOS, Objective-C, iPhone, iPad, Vuforia, Trackable, Target, Image Target, Frame Marker, Multi Target, Virtual Button, detección, seguimiento, modelo 3D, OpenGL.

Abstract

This Project consists in the realization of an application for students of the University to get help and information from the University Campus. The application runs on iOS mobile devices: iPhone, iPad and iPod Touch and employs Augmented Reality technologies provided by Qualcomm Vuforia SDK.

This application will serve as a link between supports, at first sight incompatible, as a single printed sheet of paper and the electronic device itself using Augmented Reality, which provide the printed support interaction and multimedia content through the mobile device screen .

To carry out this Project different tools needed to develop an application for the Apple platform will be studied and the knowledge necessary for this purpose will be acquired.

It also makes a study about the current state of Augmented Reality and tools capable of providing it. One of these technologies will be used, Vuforia, to provide Augmented Reality capabilities to the application and explains in detail how is the operation and architecture of this technology.

Three dimensions modeling techniques will be studied to incorpórate specific virtual realities to the application to develop and set out in detail how was the process of the application development and operation.

Keywords

Smartphone, Augmented Reality, Virtual Reality, Apple, iOS, Objective-C, iPhone, iPad, Vuforia, Trackable, Target, Image Target, Frame Marker, Multi Target, Virtual Button, detection, tracking, 3D model, OpenGL.

(Hoja de blanco)

Índice general

Índice de Figuras

Índice de Tablas

(Hoja en blanco)

1. Introducción y Objetivos

1.1. Introducción

Cuando un estudiante accede por primera vez al mundo universitario, es habitual que durante el periodo de tiempo previo a su incorporación a la Universidad y sus primeras jornadas de vida en el Campus se vea abrumado por el cambio que supone adentrarse en la Universidad.

El alumno se ve inmerso en un entorno y una dinámica totalmente diferente a la que se encontraba en el instituto ya que, por lo general, la Educación Secundaria y Bachillerato se imparten en centros que constan de un edificio y en un horario continuado; mientras que en la Educación Superior en la Universidad es habitual que las instalaciones se compongan de uno o varios campus con mayor cantidad de edificios y los horarios, como consecuencia de las prácticas, disponibilidad de laboratorios, de aulas, etc. hace que estos horarios estén más fragmentados. Además, en la Universidad a parte de la docencia en sí misma, también se ofrecen cursos deportivos, culturales, etc. que pueden ser del interés del alumno.

Por todo ello es habitual que, durante este periodo, el estudiante acumule gran cantidad de información, fundamentalmente a través de dos tipos de soporte bien diferenciados: por una parte folletos y guías informativas impresas en papel, y por otra parte, a través de internet en el sitio web de la Universidad.

Mientras que el soporte tradicional en papel proporciona una información limitada al contenido que tenga impreso, recabar información a través de internet ofrece al estudiante un valor añadido en forma de contenido multimedia (audio, vídeo, galerías de imágenes, etc.) que lo hace más atractivo y le proporciona un mayor grado de conocimiento. Sin embargo, el hecho de usar internet y buscar la información, supone mayor esfuerzo y tiempo en encontrar la información deseada que el visualizar un folleto ya impreso con información concreta.

Por tanto se tienen dos soportes informativos de ayuda al estudiante totalmente distintos como son el papel impreso tradicional e internet a través de un dispositivo electrónico. Estos dos soportes son, a priori, incompatibles en el sentido de que el alumno no puede conseguir una interacción entre ambos que le proporcione mayor información; son soportes que se suplementan, es decir, el alumno los utiliza de forma independiente y es él mismo quien abstrae la información obtenida de cada uno de ellos y la compone para lograr un mayor conocimiento complementando la información obtenida de uno y otro soporte.

El grado de penetración de los dispositivos tipo Smartphone en nuestra sociedad actual hace que gran cantidad de alumnos dispongan de teléfonos inteligentes (Smartphones); mediante los cuales se puede tener acceso a internet y visualizar contenidos multimedia. Uno de los terminales más populares de estas características hoy en día es el iPhone de Apple, el cual además, cuenta con cámara digital incorporada.

Aprovechando este hecho, sería interesante ver cómo se podría utilizar uno de estos dispositivos tan habituales actualmente para que sirviera como nexo de unión entre ambos soportes; tratar de ver cómo se podría emplear un dispositivo de estas características para acercar el mundo del papel impreso al mundo de la interactividad, al mundo de internet y lograr esa interactuación entre ambos soportes en forma de sinergia que proporcione al estudiante una mayor cantidad de información, de forma novedosa y atractiva.

La Universidad Carlos III de Madrid ha publicado recientemente un paquete de aplicaciones para Smartphones iOS[1] (también disponibles para la plataforma Android) que, a grandes rasgos, proporcionan servicios de información institucional, agenda, mapas, ayuda al estudiante, etc. pero todo ello desde una perspectiva de aplicación convencional que nada tiene que ver con lo que se plantea en este Proyecto.

1.2. Objetivos

Se pretende realizar una aplicación para dispositivos móviles iOS de Apple (iPhone, iPad y iPod Touch), a modo de prueba de concepto, que junto con un folleto impreso tradicional de información sobre la Universidad Carlos III de Madrid y haciendo uso de Realidad Aumentada, le sirva al estudiante como sistema de ayuda en la Universidad.

El **objetivo principal** que se persigue con la realización de este Proyecto es:

- Dotar a un folleto informativo de más contenido del que se presenta de forma impresa mediante el uso de funcionalidades de Realidad Aumentada a través de un dispositivo móvil inteligente, en este caso un dispositivo iOS, que proporcione al alumno ayuda e información sobre la Universidad de forma interactiva haciendo uso del propio folleto y junto con el dispositivo móvil, de manera que disponga de una experiencia de usuario novedosa, atractiva y con contenidos multimedia.

[1] Aplicaciones UC3M: http://hosting01.uc3m.es/semanal3/blog/

Para alcanzar este objetivo se establecen los siguientes **objetivos parciales**:

- Conocer cómo es la arquitectura y funcionamiento de la plataforma iOS de Apple.
- Aprender a programar en lenguaje de programación Objective-C, el cual es el lenguaje principal de iOS.
- Manejar el entorno de desarrollo de aplicaciones Xcode.
- Estudiar los entornos de desarrollo de Realidad Aumentada disponibles en iOS.
- Comprender el funcionamiento y arquitectura del SDK Vuforia de Qualcomm.
- Integrar las funcionalidades de Realidad Aumentada proporcionadas por Vuforia en una aplicación para iOS.
- Aprender a modelar objetos 3D, así como OpenGL, para crear y representar las Realidades Virtuales que aparecerán en la aplicación.

1.3. Motivación

La motivación principal para realizar este es realizar una aplicación que sirva para resolver el problema sobrecarga informativa que sufren los estudiantes de nuevo ingreso a la Universidad, debida a la gran cantidad de información y soportes que tienen a su disposición. Además, el hacer uso de un dispositivo móvil conectado a internet, para dotar de más contenidos a un soporte tan limitado como el papel impreso; que aunque en la realización de este Proyecto se centre en la ayuda al estudiante, se trata de una solución con mucho potencial aplicable a una gran variedad de ámbitos (marketing, publicidad, videojuegos, etc.) y proporciona infinidad de posibilidades.

Por otra parte, una segunda motivación es aprender a programar una aplicación para dispositivos iOS de Apple; una plataforma para la cual no se enseña a programar en la Carrera y que tiene una cuota del 65,27% en el mercado mundial de sistemas operativos móviles[2], hace que sea una plataforma muy atractiva para el desarrollador y con perspectivas de futuro, ya que la cuota de mercado tiene una tendencia ascendente.

1.4. Fases de desarrollo

Para la consecución del presente Proyecto se han llevado a cabo varias fases de desarrollo, las cuales se describen a continuación en orden cronológico ascendente:

[2] Cuota total del mercado de sistemas operativos móviles, incluyendo tabletas, en Junio de 2012, según NetMarketShare.

FASE 1: Introducción a la programación en Objective-C

Puesto que se desarrolla una aplicación para dispositivos iOS, se comenzó por investigar los requerimientos necesarios para poder llevarla a cabo.

El lenguaje de programación principal para programar aplicaciones iOS es Objective-C, y puesto que no se disponía de conocimiento alguno acerca de dicho lenguaje, hubo que estudiar cuáles eran las características del mismo, así como su sintaxis, etc. Para aprender a programar en Objective-C hubo que documentarse mediante la lectura del libro "Objective-C for Absolute Begginers" (Bennett, Fisher, & Brad, 2010) y sendos tutoriales y páginas web.

FASE 2: Despliegue del entorno de desarrollo

Apple pone a disposición de los desarrolladores el entorno de desarrollo integrado Xcode, el cual se instaló para aprender su funcionamiento y familiarizarse con las herramientas de desarrollo.

Fue preciso acceder al *iOS Developer Program*, ya que dada la naturaleza del Proyecto, se necesitaba una cuenta de pago de Desarrollador de Apple para poder probar la aplicación a desarrollar sobre dispositivo físico; ya que una cuenta gratuita sólo permite ejecutar aplicaciones en desarrollo sobre simulador, desde el cual no se puede acceder funcionalidades de cámara.

Para dotar a la aplicación de las funcionalidades de Realidad Aumentada, se instaló el SDK Vuforia de Qualcomm, como complemento a Xcode, lo cual precisa de una cuenta de desarrollador de Qualcomm para poder descargar e instalar la extensión, si bien en esta ocasión todo se ofrece de forma gratuita.

FASE 3: Documentación para programación de aplicaciones para iOS

Una vez desplegado el entorno de desarrollo, se procedió a la documentación para conocer el funcionamiento de las herramientas de desarrollo mediante la lectura del libro "Teach Yourself iOS 5 Application Development" (Ray, 2012) que además sirvió como punto de partida para documentarse sobre cómo es la arquitectura de la plataforma iOS y sus aplicaciones. Paralelamente a la lectura de este libro se fueron desarrollando diversas aplicaciones sencillas que se proponían en el libro a modo de entrenamiento.

FASE 4: Documentación para programar aplicaciones usando Vuforia

Tras la lectura del libro y la programación de las aplicaciones propuestas, se comenzó a estudiar el SDK de Realidad Aumentada Vuforia (versión 1.5.8) proporcionado por Qualcomm. Para adentrarse y comenzar a entender el SDK se procedió a documentarse a través de la propia web de Qualcomm, en la sección de desarrolladores. Ahí se encuentra disponible información relativa al API, etc.

Durante este proceso de documentación se descubrió que la sintaxis de este SDK está escrita en C++, un lenguaje del que tampoco se tenía conocimiento; por ello también hubo que documentarse sobre él.

Se descargó el código de aplicaciones de ejemplo implementadas por Qualcomm para ver el funcionamiento de las mismas sobre el dispositivo físico.

Si bien Qualcomm ofrece bastante información teórica sobre cómo funciona Vuforia, no proporciona ninguna información práctica sobre cómo aplicar su tecnología sobre una aplicación construida desde cero. Por tanto se tuvieron que llevar a cabo labores de Ingeniería Inversa sobre las aplicaciones de ejemplo proporcionadas para comprender el funcionamiento de Vuforia desde la perspectiva del código.

FASE 5: Documentación sobre modelado en 3D

Se descubre que para la representación de las Realidades Virtuales en 3D se emplea OpenGL ES, especificación de la que no se tiene ninguna noción y por tanto es preciso documentarse al respecto empleando tutoriales y diferentes sitios web, además de emplear el libro "Developing Graphical Applications with OpenGL ES, iPhone 3D Programming" (Rideout, 2010).

Puesto que para crear los Modelos 3D, se precisa de herramientas de modelado 3D, se buscó software que se adaptara a las necesidades del Proyecto, en este caso se instalaron las herramientas Cheetah3D, Blender y Adobe Photoshop CS6.

FASE 6: Desarrollo de la aplicación final

Adquiridos todos los conocimientos necesarios, se comienza la implementación de la aplicación final junto con la elaboración del folleto informativo. Durante la fase de implementación se realizaron consultas a los expertos de Vuforia a través del foro de la página de desarrolladores de Qualcomm.

FASE 7: Redacción de la Memoria

Una vez desarrollada la aplicación final, se procede a la elaboración y redacción de la presente Memoria.

1.5. Medios empleados

A continuación se realiza una enumeración de los medios que han sido necesarios emplear para la realización de este Proyecto:

Hardware:

- Ordenador portátil Apple MacBook Pro 6,2

- Teléfono móvil Apple iPhone 4, 32GB
- Cable Base Dock para cable USB de Apple
- Impresora Photosmart 5510 e-All-in-One de HP con AirPrint

Software:

- Apple OS X 10.7.4 (Sistema Operativo MacBook Pro)
- Apple iOS 5.1.1 (Sistema Operativo iPhone)
- Xcode 4.3.2 (Entorno de desarrollo)
- Cuenta Apple Developer Program (Pruebas en dispositivo físico)
- Vuforia SDK 1.5.8 de Qualcomm (SDK Realidad Aumentada)
- Cheetah3D 6.0 (Herramienta modelado 3D)
- Blender 2.63a (Herramienta modelado 3D)
- OpenGL Export Addon (for Blender 2.63) [Exportación de Modelos 3D]
- Adobe Photoshop CS6 (Herramienta diseño gráfico, texturización Modelos 3D)

Otros:

- Conexión a Internet (Descarga de herramientas de desarrollo, documentación, etc.)
- Folios DIN-A 4 (folleto)

1.6. Estructura de la Memoria

El desarrollo de esta Memoria comenzará por el *Planteamiento del Problema*; en el que se hará un *Análisis del Estado del Arte* abordando los campos que abarca el Proyecto desde una perspectiva general hasta las tecnologías más específicas que dan cuerpo a este Proyecto. Posteriormente se abordará el *Diseño de la Solución Técnica*, tras lo cual en el capítulo *Resultados y evaluación* se expondrá cómo es el funcionamiento que ofrece la aplicación desde la perspectiva del usuario final.

La Memoria prosigue con una explicación de cómo extender la funcionalidad de la aplicación desarrollada o de nuevas aplicaciones que hagan uso de las tecnologías empleadas en ésta en el capítulo *Futuras líneas para ampliar el Proyecto*. A continuación se presenta el *Presupuesto y planificación* del Proyecto y para concluir con la Memoria se presentarán las *Conclusiones* obtenidas de la realización de este Proyecto.

2. Planteamiento del problema

2.1. Análisis del Estado del Arte

En anteriores puntos de esta Memoria se ha indicado que la característica clave de este Proyecto es el empleo de Realidad Aumentada; es preciso conocer en qué consiste. La Realidad Aumentada consiste en la definición de una visión del mundo real a la cual se añaden componentes virtuales que no existen en la realidad; por tanto se crea una Realidad Mixta en la cual conviven elementos reales y elementos virtuales. No se debe confundir la Realidad Aumentada con la Realidad Virtual puesto que esta última se aísla de lo real y es puramente virtual, como consecuencia, no existe coexistencia de mundo real y mundo virtual.

2.1.1. Realidad Aumentada

En la actualidad se aceptan dos definiciones para explicar qué es la Realidad Aumentada. La primera, (Milgram & Kishino, 1994) fue realizada por Paul Milgram y Fumio Kishino en 1994. Definieron un concepto que llamaron Continuo Virtualidad –*Virtuality Continuum*- que consiste en una escala continua entre lo completamente virtual (Entorno Virtual), y lo completamente real (Entorno Real). El continuo realidad-virtualidad por tanto, abarca todas las posibles variaciones y composiciones de objetos reales y virtuales. El área que está entre los extremos, donde tanto lo real como lo virtual se mezcla, recibe el nombre de Realidad Mixta –*Mixed Reality*-; ésta a su vez se dice que consta de Realidad Aumentada –*Augmented Reality*-, donde lo virtual aumenta lo real; y la Virtualidad Aumentada –*Augmented Virtuality*-, donde lo real aumenta lo virtual. Mientras que el término Virtualidad Aumentada raramente es utilizado, Realidad Aumentada y Mixta se suelen usar como sinónimos.

Figura 2.1: Continuo Virtualidad - Virtuality Continuum-

La segunda de las definiciones la hizo el Doctor Ronald T. Azuma en 1997. Según él (Azuma, 1997), para poderse considerar Realidad Aumentada, ésta ha de mezclar elementos de la realidad con elementos inexistentes en la misma, es decir Realidades Virtuales; el individuo que la experimente ha de poder interactuar con ella en tiempo real y ha de estar en tres dimensiones.

Las tecnologías existentes en la actualidad para producir Realidad Aumentada, se pueden englobar en dos categorías: Tecnología hardware dedicada que proporcione capacidades de Realidad Aumentada y software aplicable a soluciones hardware ya existentes y cuyo propósito general no es el de proporcionar Realidad Aumentada, por ejemplo Smartphones, tabletas, etc. Por tanto, para simplificar se dirá que existen soluciones hardware y soluciones software.

2.1.1.1. Soluciones Hardware

En rasgos generales, los dispositivos dedicados de Realidad Aumentada están formados por unos cascos o auriculares que se acoplan a la cabeza del usuario (u otro tipo de soluciones que se expondrán a continuación) y un sistema de visualización para mostrarle la Realidad Virtual que se incluye al mundo real. Los cascos o auriculares pueden ser susceptibles de llevar incorporado sistemas de posicionamiento GPS para permitir determinar con exactitud la localización del usuario.

Los principales sistemas de visualización que se vienen empleando son pantallas de óptica transparente (Optical See-through Display) y pantallas de mezcla de imágenes (Video-mixed Display).

A continuación se explicarán las tres técnicas más destacadas para la representación de la Realidad Aumentada.

Display montado sobre la cabeza del usuario

Este tipo de pantallas se conoce por el acrónimo HMD (head-mounted display). Estos displays representan tanto imágenes del mundo real que está percibiendo el usuario como imágenes de Realidades Virtuales en tres dimensiones sobre la vista del mundo real.

Se trata de un dispositivo de visualización parecido a un casco en el que sobre una lente muy próxima al ojo del usuario, que le permite ver el mundo real, se proyectan los elementos virtuales que, con la combinación de la visión del mundo real, genera la Realidad Aumentada. La proyección de la Realidad Virtual, en lugar de proyectarse sobre una lente, puede ser proyectada directamente sobre la retina

de los ojos del usuario; en este caso el HMD recibe el nombre de *monitor virtual de retina*.

Los HMD se pueden clasificar atendiendo a dos parámetros; según su configuración física sobre la cabeza del usuario y según el campo de visión del mundo real que se le permita ver. Según el primer parámetro, los HMD pueden ser:

- Monocular: Las imágenes virtuales sólo pueden ser percibidas por un ojo.
- Binocular: Las imágenes virtuales pueden ser percibidas por ambos ojos, obteniendo de este modo una imagen estereoscópica.

Según el segundo parámetro, existen HMD destinados a Realidad Virtual y destinados a la representación de Realidad Aumentada. Cuando el dispositivo HMD reduce tanto el campo de visión del usuario, éste se ve totalmente inmerso en una Realidad Virtual, ya que solamente será capaz de ver las imágenes creadas artificialmente por ordenador sobre el display.

Mientras, los Head-Mounted Displays que permiten ver al usuario el entorno en el que está inmerso y añadir a esa visión objetos virtuales son los destinados a la representación de Realidad Aumentada.

Un ejemplo de actualidad de dispositivos dedicados de Realidad Aumentada es el PGlass que está realizando Google (All Things D., 2012). Las gafas prototipo fueron presentadas en Abril de 2012 en Google+, si bien la idea no es nueva, lo que ha sorprendido de este Proyecto es que las gafas son mucho más pequeñas y delgadas que todas las soluciones existentes hasta el momento; además Google ha indicado que el siguiente paso será llevar el diseño del dispositivo a uno nuevo que permita la integración con cualquier gafa convencional existente. Por todo esto, el Proyecto Glass de Google ha tenido tanta repercusión, debido a las posibilidades reales de implantación en el mercado de las que dispone, así como de su potencial.

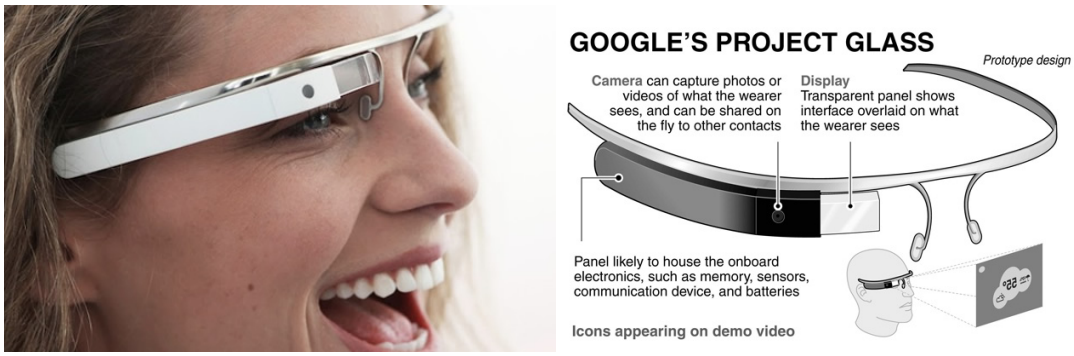


Figura 2.2: Dispositivo perteneciente al Proyecto Google Glass (FUENTE: Google)

Display de mano

Constan de pantallas portátiles cuyo tamaño se ajusta a la mano del usuario. Todas las soluciones disponibles de este tipo hasta la fecha consisten en la visualización de la realidad a través de éstas (a través de la cámara digital de la que dispone el dispositivo) y sobre ellas se montan las diferentes Realidades Virtuales para dotar al usuario de la percepción de Realidad Aumentada.

Al principio, los dispositivos de Realidad Aumentada con display de mano empleaban para realizar el seguimiento del dispositivo, y por tanto, del usuario; marcadores fiduciales, para más adelante usar unidades de geoposicionamiento GPS y sensores MEMS como brújulas digitales y acelerómetros/giroscopios de seis grados de libertad. Posteriormente, sistemas como ARToolKit, permitieron añadir información digital a las secuencias de vídeo en tiempo real. Actualmente, sistemas de visión como SLAM[3] o PTAM[4] son los que se están empezando a emplear para realizar el seguimiento.

Figura 2.3: Ejemplo de dispositivo Display de mano
(FUENTE: zbexodus.com)

Este tipo de dispositivo parece que va ser el que mayor éxito comercial tenga de entre los diferentes tipos de dispositivos que proporcionan Realidad Aumentada; de hecho hoy en día son los dispositivos de Realidad Aumentada por antonomasia en la sociedad; puesto que las ventajas más significativas que ofrece son el carácter portátil de los dispositivos de mano y la posibilidad de ser aplicada en los teléfonos con cámara, tan habituales en la sociedad de hoy. Sin embargo, tienen como desventaja las limitaciones físicas, por el hecho de que los usuarios han de sostener el dispositivo por delante de ellos en todo momento, así como el

[3] Simultaneous Localization and Mapping, Localización y Mapeado Simultáneos.

[4] Parallel Tracking and Mapping, Seguimiento y Mapeado Paralelo.

efecto distorsionador de ángulo típico de las cámaras de los teléfonos móviles en comparación con el mundo real visto a través de los ojos humanos (Feiner, 2011).

Display espacial

En lugar de que el usuario tenga que llevar puesto o tenga que sostener tanto un dispositivo Head-Mounted Display o Display de mano, los dispositivos de Realidad Aumentada Espacial (SAR, Spatial Augmented Reality) hacen uso de proyectores digitales para mostrar información gráfica sobre los objetos físicos (Raskar, Greg, & Henry, 1998).

2.1.1.2. Soluciones Software

Se trata de proporcionar software a dispositivos electrónicos tales como Smartphones o tabletas, los cuales disponen de las tecnologías hardware necesarias para ofrecer funcionalidades de Realidad Aumentada. El software ha de ser capaz de realizar una fusión coherente entre las imágenes obtenidas con la cámara del dispositivo del mundo real y las imágenes virtuales en 3D; estas imágenes en 3D han de posicionarse en lugares del mundo real. Con las imágenes capturadas por la cámara, el mundo real ha de ser situado en un sistema de coordenadas; esto recibe el nombre de *registro de imágenes*. Este proceso involucra diversos métodos de visión computacional, gran parte de ellos relacionados con el seguimiento de secuencias de vídeo. Gran número de métodos de visión por ordenador de Realidad Aumentada se heredan de modo similar a los métodos de odometría visual.

En el mercado se encuentran disponibles multitud de soluciones software de Realidad Aumentada que se pueden englobar en software de código abierto -*open source*- y software de código cerrado gratuito -*free closed source*-.

Software open source

A continuación se citan algunos ejemplos existentes de herramientas de código abierto.

- **Argon**: Se trata de un navegador de Realidad Aumentada que utiliza una combinación de KML y HTML, JavaScript, CSS para permitir el desarrollo de aplicaciones de Realidad Aumentada. Cualquier contenido web puede convertirse en contenido apto para ser representado dentro de la Realidad Aumentada. Disponible para iPhone.
- **ArUco:** Biblioteca reducida para el desarrollo de aplicaciones de Realidad Aumentada basada en OpenCV. Licencias disponibles para BSD, Linux y Windows.

- **Mix Augmented Reality Engine (mixare):** Motor de Realidad Aumentada para Android y iPhone; funciona como una aplicación autónoma y para desarrollar otro tipo de implementaciones.
- **ARToolKit:** Biblioteca que emplea funcionalidades de seguimiento de vídeo, para calcular la ubicación de la cámara y la orientación relativa respecto a la posición de los marcadores físicos que se sitúan en la Realidad; todo en tiempo real, para crear aplicaciones de Realidad Aumentada.
- **ATOMIC Authoring Tool:** Se trata de una herramienta que permite la creación de aplicaciones de Realidad Aumentada para que sean desarrolladas, especialmente, por no programadores.
- **ATOMIC Web Authoring Tool**: Se trata de una herramienta que permite la creación de aplicaciones de Realidad Aumentada para que puedan ser exportadas a cualquier sitio web.

Software gratuito closed source

A continuación se citan dos ejemplos de herramientas gratuitas de código cerrado.

- **Metaio Mobile SDK:** Solución gratuita que proporciona funciones de seguimiento natural. Disponible para iPhone y Android.
- **Qualcomm AR SDK (QCAR):** Rebautizado como Vuforia, es una solución gratuita para la detección y seguimiento se imágenes de referencia y marcadores usando características de detección natural. Se encuentra disponible para iPhone y Android. Se trata de la solución elegida para el desarrollo de la aplicación de este Proyecto.

2.1.1.3. Smartphones

(PC Magazine, 2011) El término Smartphone hace referencia a "teléfono móvil inteligente"; el hecho de llamarlo "inteligente" es porque esta clase de dispositivos ofrecen más funciones que los móviles tradicionales. En la actualidad, para que un teléfono móvil se considere "inteligente" ha de disponer de una serie de características entre las que destacan:

- Permiten la instalación de aplicaciones.
- Deben contar con algún sistema operativo.
- Cuentan con GPS.
- Permiten el acceso a internet.
- Soporte a sistemas de correo electrónico.
- Deben disponer de cámara digital.

- Funciones de agenda digital, gestión de contactos, calendarios, etc.
- Lectura de documentos de distinto formato.
- Reproducción de audio y vídeo.

Todas estas características implican que este tipo de dispositivos tengan una gran capacidad de cómputo, en forma de potentes procesadores, memoria RAM, almacenamiento, etc.; que además permitan el uso simultáneo de varias de estas funciones.

Sobre su diseño, es habitual que los Smartphones posean un tamaño significativamente mayor al de un móvil tradicional, esto se debe a la necesidad de incorporar ciertas características especiales como pantallas táctiles de grandes dimensiones.

De todo lo anterior se concluye que, las características de los teléfonos inteligentes les convierten en dispositivos potenciales para proveer funciones de Realidad Aumentada; en concreto son "Displays de mano" sin el software necesario para proporcionar Realidad Aumentada. En la actualidad, estos dispositivos se han convertido en los dispositivos de Realidad Aumentada por antonomasia, gracias a las aplicaciones que proporcionan funciones de Realidad Aumentada, que además han sido las causantes de acercar este mundo a una gran cantidad de público.

2.1.1.4. Aplicaciones

Como se ha visto hasta ahora, la Realidad Aumentada proporciona una nueva forma de interacción que hace que esta tecnología se aplique en diferentes campos:

- **Educación**: Puesto que el coste que supone realizar soluciones de Realidad Aumentada todavía no es lo suficientemente bajo; su uso está básicamente restringido a museos, parques temáticos, etc. En estos lugares es habitual la presencia de redes wi-fi, que junto con las aplicaciones de Realidad Aumentada, permiten que los usuarios obtengan información sobre lugares, obras o puedan visualizar recreaciones en 3D de obras arquitectónicas, artefactos científicos, etc.
- **Entretenimiento**: El campo de aplicación por antonomasia de la Realidad Aumentada en el mundo del entretenimiento son los videojuegos, ya que proporciona una forma muy novedosa a la hora de jugar; sobre todo teniendo en cuenta que en los últimos años han ido ganando popularidad plataformas de juegos con controles no tradicionales, que se escapan del mando tradicional, como es el caso de la Wii de Nintendo y los juegos táctiles de las plataformas

móviles. Por ello, esta nueva forma de jugar puede gozar de gran éxito en los próximos años.

Un proyecto que goza de bastante éxito es ARQuake Project, que permite jugar al conocido Quake pero en exteriores disparando a enemigos virtuales.

- **Arquitectura**: Para la recreación de edificios deteriorados o derruidos, así como para la representación de edificios de nueva construcción y la generación de planos de planta en 3D que proporcionan una visión más realista de los mismos.
- **Dispositivos de navegación**: Facilitan el uso de navegación dentro de un edificio mostrando dónde se ubican cada una de las estancias del mismo; además de, en automóviles emplear la luna para superponer información de tráfico, destino, etc.
- **Publicidad**: En la actualidad se emplea la Realidad Aumentada en este campo, sobre todo, para llamar la atención del usuario mediante anuncios interactivos, susceptibles de proporcionar intercatividad.
- **Turismo**: Existen plataformas que no requieren prácticamente conocimientos técnicos para desarrollar aplicaciones a terceros; un par de ejemplos son Junaio o Layar; que han fomentado el lanzamiento de gran número de aplicaciones sobre turismo, etc.

A continuación se exponen con más detalle las características del SDK Vuforia de Qualcomm, puesto que se trata de la herramienta de desarrollo que dotará a la aplicación a implementar en este Proyecto de las funcionalidades de Realidad Aumentada.

2.1.2. Vuforia

Una aplicación basada en el SDK Vuforia de Realidad Aumentada hace que la pantalla del dispositivo móvil funcione como unas "lentes mágicas" que al mirar a través de ellas, el usuario percibe un mundo aumentado en el que el mundo real y virtual coexisten. La aplicación superpone objetos virtuales 3D sobre la visión en directo capturada por la cámara del dispositivo, de modo que estos objetos virtuales parecen que realmente se encuentran en la escena real.

La figura que se muestra a continuación muestra una vista general de cómo se realiza el desarrollo de una aplicación con la plataforma Vuforia. Esta plataforma consta del SDK Vuforia y un *Target Management System (Sistema de gestión de Imágenes de Referencia)* alojado en el portal de desarrolladores de Qualcomm, QDevNet[5]. El desarrollador puede subir imágenes para generar la imagen de

[5] http://ar.qualcomm.at

referencia (trackable) para realizar el seguimiento y una vez generada por el sistema, descargarse los recursos necesarios que serán incluidos en la aplicación a desarrollar. El SDK Vuforia proporciona una biblioteca (objeto compartido "libQCAR.so" en Android, y biblioteca estática "libQCAR.a" en iOS) que debe ser vinculada a la aplicación.

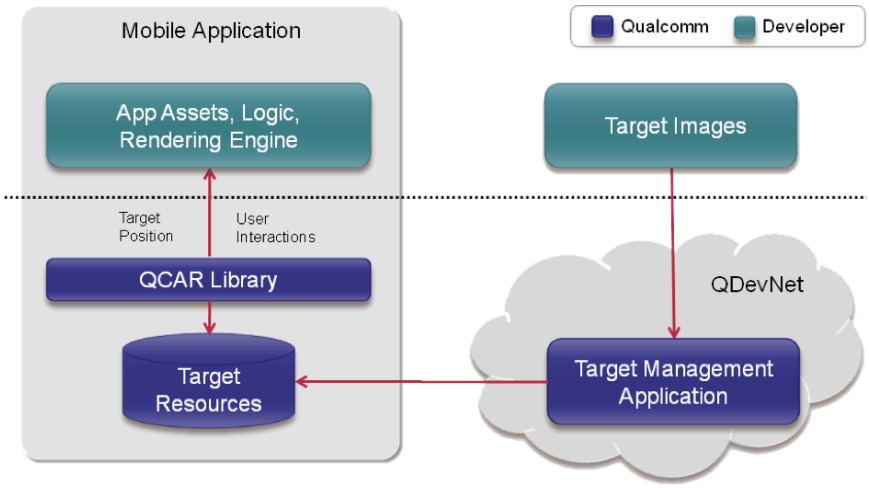


Figura 2.4: Diagrama que muestra el proceso de desarrollo de una aplicación Vuforia (FUENTE: Qualcomm)

2.1.2.1. Arquitectura

Una aplicación de Realidad Aumentada basada en Vuforia, consta de una serie de componentes esenciales. A continuación se describirán términos referentes a la arquitectura de clases básicas de Vuforia, muchos de estos componentes se tratan de *singletons*; cuya traducción literal sería "hijo único", y que consiste en una clase diseñada para tener sólo una instancia (o un número limitado de ellas). Conociendo esto, y por comodidad a la hora de emplear el lenguaje, se llamará instancia a un *singleton*.

- **Cámara** (camera, *singleton*): La instancia de la cámara se encarga de que cada fotograma capturado por la cámara digital se pase de forma eficiente al tracker. El desarrollador ha de decir a la instancia de la cámara, con los métodos oportunos, cuando debe comenzar y detenerse la captura de fotogramas. Cada fotograma es enviado automáticamente a un dispositivo dependiente del formato de imagen y dimensiones.
- **Convertidor de imágenes** (Image Converter, *singleton*): La instancia del conversor de formato de pixel realiza la conversión entre el formato con el que trabaja la cámara (por ejemplo, YUV12) a un formato adecuado para el renderizado en OpenGL ES (por ejemplo, RGB565) y para el seguimiento (por ejemplo, luminancia). Esta conversión implica un submuestreo para tener la

imagen capturada por la cámara en diferentes resoluciones disponible en la pila de fotogramas convertidos.

- **Tracker** (*singleton*): El tracker contiene los algoritmos de visión computacional para detectar y seguir (detect & track) los objetos en los fotogramas capturados por la cámara. Basado en la imagen tomada por cámara, diferentes algoritmos se ocupan de detectar nuevas imágenes de referencia (Targets) o marcadores (Markers), y evaluar los botones virtuales (Virtual Buttons). Los resultados se almacenan en un objeto de estado que es utilizado por el procesador de vídeo de fondo y al que puede accederse desde el código de la aplicación. El tracker puede cargar múltiples conjuntos de datos (datasets), pero sólo uno puede estar activo a la vez.
- **Procesador de vídeo de** fondo (Video Background Rendered, *singleton*): La instancia del procesador de vídeo de fondo procesa la imagen capturada por la cámara que se encuentra almacenada en el objeto de estado. El rendimiento del renderizado del vídeo de fondo está optimizado para dispositivos específicos.
- **Código de la aplicación** (Application Code): El desarrollador ha de inicializar todos los componentes anteriores y llevar a cabo tres pasos fundamentales en el código de la aplicación. Por cada fotograma procesado, el objeto de estado se actualiza y se llama al método de procesamiento de la aplicación. El desarrollador debe:
  - Consultar el objeto de estado para los Targets y/o Markers nuevos que puedan aparecer en escena o si se actualiza su estado.
  - Actualizar la lógica de la aplicación con nuevos datos de entrada.
  - Renderizar la capa de Realidad Aumentada.
- **Recursos de imágenes de referencia** (Target Resources): Los Target Resources se crean mediante el Target Management System, que se encuentra disponible on-line. El dataset descargado contiene un fichero de configuración XML que permite al desarrollador configurar ciertas características de los trackables y un fichero binario que contiene la base de datos de los trackables. Estos elementos (XML y binario) son compilados por la aplicación a desarrollar en el paquete de instalación de la aplicación y los usa el SDK Vuforia en tiempo de ejecución.

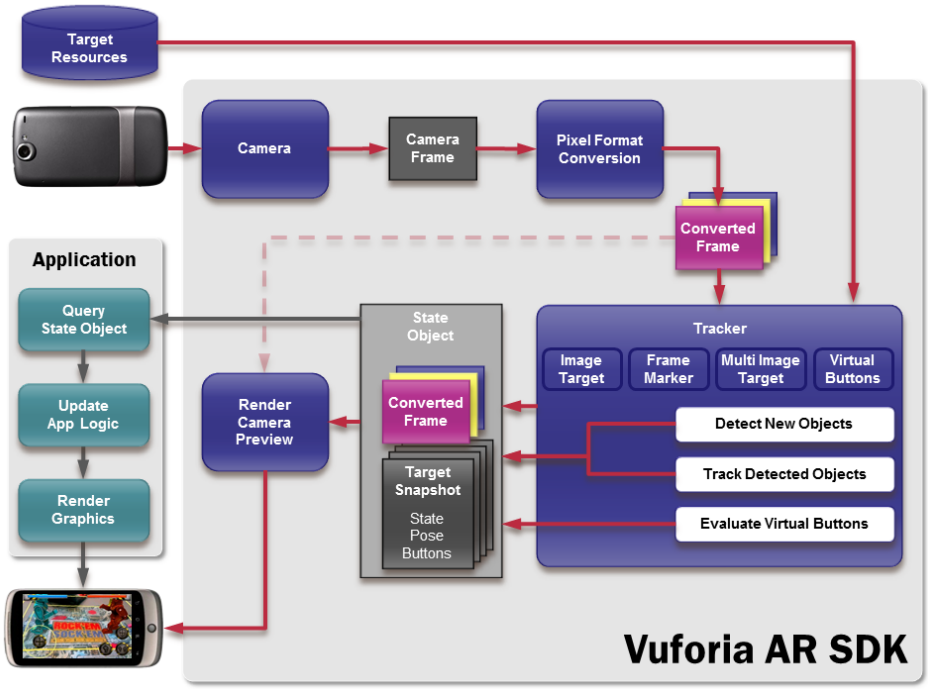


Figura 2.5: Diagrama que muestra el flujo de datos del SDK Vuforia en un entorno de aplicación (Fuente: Qualcomm)

2.1.2.2. Trackables

Los "trackables" constituyen la clase base que representa todo objeto real que puede ser seguido en seis grados de libertad por el SDK Vuforia. Cada trackable, cuando es detectado y seguido (detected & tracked), tiene un nombre, un identificador, estado e información sobre su posición (pose information). Tanto Image Targets, Multi Targets como Markers son trackables que heredan propiedades de esta clase base. Los Trackables se actualizan cada vez que se procesa un fotograma y los resultados se pasan a la aplicación en el objeto de estado (State).

Parámetros

- **Tipo de trackable:** Enumeración que define el tipo de trackable:
  - UNKNOWN_TYPE: Trackable de tipo desconocido.
  - IMAGE_TARGET: Trackable de tipo ImageTarget.
  - MULTI_TARGET: Trackable de tipo MultiTarget.
  - MARKER: Trackable de tipo Marker.
- **Nombre/identificador de trackable:** String que define de forma única al trackable dentro de la base de datos de Targets. Su longitud máxima es de 25 caracteres. A-z, A-Z, 0-9, [- _ .]

- **Estado del trackable:** Cada trackable tiene una información de estado asociada con él en el objeto State, el cual se actualiza cada vez que se procesa un fotograma. El estado viene caracterizado por una enumeración (enum):
  - UNKNOWN: El estado del trackable es desconocido. Este estado es devuelto normalmente antes de la inicialización del tracker.
  - UNDEFINED: El estado del trackable no está definido.
  - NOT_FOUND: El trackable no se encuentra, por ejemplo, esto sucede cuando el trackable no está en la base de datos.
  - DETECTED: El trackable se ha detectado en el fotograma actual.
  - TRACKED: El trackable se ha seguido en el fotograma actual.
- **Posición del trackable:** La posición actual válida de un trackable en estado DETECTED o TRACKED se devuelve como una matriz 3x4 en orden de filas mayor. El SDK Vuforia proporciona herramientas para convertir estas matrices de posición en matrices modelo-vista GL (GL model-view matrix) y para proyectar puntos en tres dimensiones desde la escena 3D a la pantalla del dispositivo.
  El SDK Vuforia emplea un sistema de coordenadas cartesianas a derechas. Cada Image Target y Frame Marker define un sistema de coordenadas local con (0,0,0) en el centro (medio) del Target, constituyendo el origen del sistema de coordenadas. +X va hacia la derecha, +Y hacia arriba y +Z hacia fuera del trackable (en la dirección desde la cual puede ser visto).

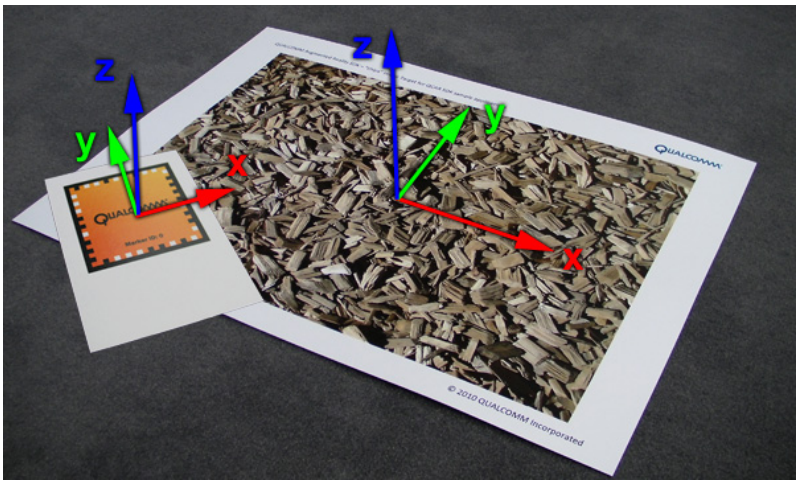


Figura 2.6: Sistema de coordenadas de dos Trackables
(FUENTE: Qualcomm)

El origen del sistema de coordenadas local de un Multi Target viene definido por sus componentes. Sus partes, constituidas por Image Targets son transformadas respecto a este origen. La posición del Multi Target devuelta por el sistema, es la posición de este origen, independientemente desde qué parte

individual del Multi Target esté siendo seguida. Esta característica permite que un objeto geométrico (por ejemplo, una caja) pueda ser seguida continuamente con las mismas coordenadas, incluso si otras partes de un Image Target están visibles en la escena y son capturadas por la cámara.

Como se ha visto, existen tres tipos de trackables: Image Targets, Multi Targets y Frame Markers. A continuación se detallan las características de todos ellos.

2.1.2.2.1. Image Targets

Los Image Targets, son imágenes que el SDK Vuforia es capaz de detectar y seguir. A diferencia de marcadores tradicionales, matrices de códigos de datos y códigos QR, los Image Targets no necesitan ningún tipo de regiones en blanco y negro especiales o códigos para poder ser reconocidos. El SDK Vuforia emplea algoritmos avanzados para detectar y seguir las características naturales que se encuentran en la propia imagen. El SDK reconoce al Image Target comparando esas características naturales de la imagen con una base de datos de Targets conocidos. Una vez el Image Target es detectado, el SDK realizará el seguimiento de dicho Target siempre que, al menos parcialmente, se encuentre visible en el campo de la escena capturado por la cámara.

Figura 2.7: Ejemplo de una Realidad Virtual sobre un Image Target
(FUENTE: Qualcomm)

Los Image Targets se crean on-line con el Target Management System, a partir de imágenes JPG o PNG (sólo se soportan imágenes en RGB o en escala de grises). Las características que se extraen de estas imágenes se almacenan en una base de datos, que se usa en tiempo de ejecución para realizar las comparaciones.

El SDK Vuforia, además de permitir múltiples imágenes de entrada en la base de datos, puede detectar y seguir hasta cinco Targets de forma simultánea en el campo de escena capturado por la cámara. Nótese que el rendimiento en este supuesto, podría variar en función de la carga del procesador y de la GPU. El SDK puede soportar aproximadamente hasta cincuenta Image Targets en la base de datos de referencia; además permite intercambiar datasets en tiempo de ejecución; por lo que una aplicación puede soportar un número de Targets más elevado todavía.

Datasets

Un dataset comúnmente se refiere a un grupo de trackables que se descargan del Target Management System. El SDK permite que las aplicaciones carguen, activen, deshabiliten y descarguen grupos de trackables (datasets) en tiempo de ejecución. Estos datasets pueden contener información relativa tanto de Image Targets como de Multi Targets.

Parámetros

- **Tamaño (Size):** Este parámetro define el tamaño real del Image Target en unidades 3D de la escena. El desarrollador ha de especificar el valor de este parámetro durante la creación on-line del trackable o en el fichero XML de Configuración del Dataset (Dataset Configuration XML file). Este fichero XML es generado por el Target Management System, pero el desarrollador puede modificarlo en cualquier momento. Este parámetro en muy importante, ya que la información de ubicación que se devuelve durante el seguimiento, tendrá también esta escala. Por ejemplo, si un Image Target tiene 16 unidades de ancho, mover la cámara desde el borde izquierdo del Target hasta su borde derecho, cambiará la posición devuelta 16 unidades sobre el eje X[6].

Hay que tener en cuanta que el tamaño de un Image Target se puede cambiar en tiempo de ejecución; lo que proporciona más flexibilidad a la hora de definir o actualizar la escala del Target en la escena 3D.

- **Botones Virtuales (Virtual Buttons):** Los Image Targets pueden contener uno o más botones virtuales. El desarrollador puede consultar al Target el número de botones virtuales asociados al mismo, recorriendo una lista y accediendo a cada Botón Virtual individualmente y comprobar su estado. Los botones virtuales se pueden añadir y eliminar en tiempo de ejecución. Más adelante, se explicarán con más detalle los botones virtuales.

[6] (x, y) – Tamaño del Target en unidades de escena medidas a lo largo de los ejes horizontal y vertical de un Target rectangular

Ejemplo

Las dos figuras que se muestran a continuación sirven para ilustrar un ejemplo en el que se puede ver un sistema de coordenadas y las características naturales de una imagen que emplea el SDK Vuforia para detectar un Image Target.

Figura 2.8: Sistema de coordenadas de un Image Target[7]
(FUENTE: Qualcomm)

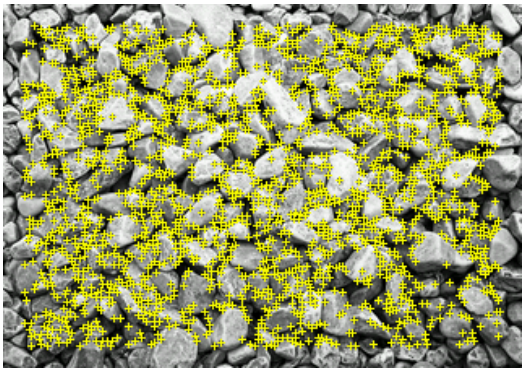


Figura 2.9: Características naturales de una imagen que Vuforia emplea para detectar el Image Target (FUENTE: Qualcomm)

Aspectos a tener en cuenta

- **Imágenes de entrada**: Los Image Targets se detectan basándose en las características naturales que son analizadas en la imagen del Target, almacenadas en la base de datos y luego comparadas en tiempo de ejecución con las características de la imagen capturadas por la cámara. Los Targets para que sean detectados con precisión deberían ser ricos en detalles, tener buen contraste (local); deben ser, por lo general, bien iluminados y no con poco brillo o colores apagados, y no deberían tener patrones repetitivos como un tablero de ajedrez o la fachada de un edificio, con ventanas similares, etc.

7 Las unidades para el tamaño están en unidades de escena 3D y el sistema de coordenadas es local al Image Target. En este ejemplo, el tamaño = [10,7] es en la escena 3D real.

- **Tamaño real**: El tamaño de la imagen de entrada del Target depende en su totalidad de la aplicación. Se recomienda que para entornos de uso en campo cercano, la imagen real del Target tenga un tamaño aproximado de 20 a 100 cm; aunque tamaños mayores y menores son perfectamente posibles. Nótese que se está refiriendo al tamaño real del Target impreso.
- **Parámetro Size** (Tamaño): El tamaño del Image Target definido en el fichero XML de configuración puede ser definido arbitrariamente. Se recomienda que el tamaño sea definido en unidades de escena, que se relacionan con el tamaño real. Por ejemplo, 1 unidad de escena es 1 cm en el mundo real. Usando esta convención es más sencillo que los elementos del mundo virtual coincidan con el mundo real cuando se desarrolle la aplicación de Realidad Aumentada.

2.1.2.2.2. Multi Targets

Un Multi Target no es más que un conjunto de Image Targets con una relación espacial fija. Una vez que una o más partes de un Multi Target se detecta, todas las partes restantes pueden seguirse debido a que su posición y orientación espacial relativa es conocida. Siempre que una de las partes del Target está, al menos parcialmente, visible en la imagen capturada por la cámara, el Multi Target puede ser seguido. La diferencia entre varios Image Targets Individuales y un Multi Target es que, siendo ambos un conjunto de Targets, el último se trata como si sólo fuera uno (una sola información de posición) a diferencia de múltiples Image Targets, que proporcionan varias informaciones de posición, estado, etc. de forma individual.

Figura 2.10: Ejemplo de una Realidad Virtual al lado un Multi Target
(FUENTE: Qualcomm)

Los Image Targets que componen un Multi Target se crean con el Target Management System on-line, del mismo modo que se explica a la hora de crear

Image Targets individuales. La relación espacial de cada una de las partes se almacena en el fichero de configuración XML usando transformaciones simples.

Los Multi Targets se pueden crear de dos formas; directamente con el Target Management System on-line o en tiempo de ejecución mediante un conjunto de APIs. Las partes se pueden añadir o eliminar, y su configuración espacial se puede modificar. Esta flexibilidad se puede usar para crear una información de posición a partir de múltiples Image Targets o para seguir objetos que cambian, pero con configuraciones espaciales conocidas. Los Multi Targets son parte de los datasets, y por tanto, pueden cargarse, descargarse, activarse y desactivarse en tiempo de ejecución.

Parámetros

La partes de un Multi Target se transforman en un sistemas de coordenadas común. En tiempo de ejecución, el origen de este sistema de coordenadas, dentro del Multi Target, se devuelve como información de posición. Así todas las partes individuales compuestas por Image Targets se sitúan con rotaciones y translaciones, dentro de este sistema. Este tipo de target tienen los siguientes parámetros:

- **Nombre/identificador de parte (Part name/id):** String que define unívocamente cada una de las partes del Multi Target. Este nombre ha de ser idéntico al empleado para definir el Image Target dentro del dataset de Targets. Esta referencia define el conjunto de características que se usa para detectar la parte en la visión en directo proporcionada por la cámara en tiempo de ejecución. Su longitud máxima es de 25 caracteres. A-z, A-Z, 0-9, [- _ .]
- **Translación (Translation):** Este parámetro traslada el origen de una parte constituida por un Image Target a lo largo de los ejes X, Y, Z las unidades de escena deseadas[8].
- **Rotación (Rotation):** Este parámetro rota el origen de una parte constituida por un Image Target el ángulo deseado. Las rotaciones se definen como series de parejas eje-ángulo que se aplican en orden. Para definir las rotaciones se permiten varios formatos:
  - *(x, y, z, ángulo dec.)* rotación en grados decimales sobre los ejes definidos en el vector (x, y, z).
  - *(x, y, z, ángulo rad.)* rotación en radianes sobre los ejes definidos en el vector (x, y, z).
  - *(qx, qy, qz, qw)* cuaternio para definir la rotación.

[8] (x, y, z) – translación en unidades de escena medidas a lo largo de los tres ejes espaciales

Figura 2.11: Ejemplo de Multi Target con sus correspondientes ejes de coordenadas
(FUENTE: Qualcomm)

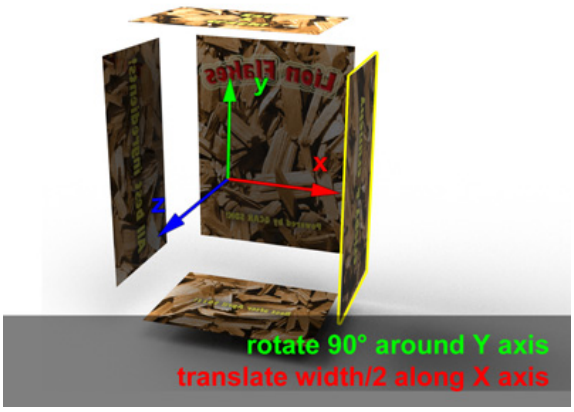


Figura 2.12: Transformación de la parte derecha de la caja; las partes individuales se sitúan empleando pares de translación-rotación
(FUENTE: Qualcomm)

Aspectos a tener en cuenta

- **Imágenes de entrada, tamaño real, parámetro Size**: Los Multi Targets se detectan y siguen como combinaciones de Image Targets. Todas las recomendaciones sobre la elección de imágenes de entrada, tamaño, etc. que aplican a Image Targets también son válidas para los Multi Targets.
- **Forma** (Shape): La herramientas de creación on-line de Targets permiten crear Multi Targets de forma cúbica. En todos estos objetos, todos los ánulos son rectángulos y las caras opuestas son del mismo tamaño. De acuerdo a esto, surge una restricción adicional para las imágenes de entrada para crear el Multi Target, las imágenes deben ser de las mismas dimensiones dos a dos.

2.1.2.2.3. Frame Markers

Además de la detección y seguimiento basada en características, el SDK Vuforia puede seguir un tipo especial de marcadores fiduciales llamados Frame Markers. El identificador único de un Frame Marker se encuentra codificado en un patrón binario a lo largo del borde de la imagen del marcador. Un Frame Marker permite que cualquier imagen se pueda situar dentro de este marco binario. Así se consigue conseguir un efecto más natural que el aspecto que presentan otros marcadores fiduciales más tradicionales. El SDK Vuforia requiere que el marco binario se encuentre totalmente visible en el fotograma para poder ser reconocido correctamente.

Figura 2.13: Ejemplo de cuatro Frame Markers sobre los que se sitúan cuatro Realidades Virtuales (FUENTE: Qualcomm)

A diferencia de los Image Targets, los Frame Markers no son generados por el Target Management System. Todos los 512 Frame Markers se distribuyen dentro de un directorio (*assets*) de instalación del SDK. Para escalarlos al tamaño deseado de impresión, se ha de usar la opción de escalado de cualquier editor de imágenes (como Adobe Photoshop) sin la opción de filtrado o "vecino más próximo"; para que al cambiar las dimensiones se mantenga definido el marco y pueda ser detectado correctamente.

Debido al relativamente bajo coste computacional requerido para decodificar la ID del marcador, todos los 512 Frame Markers pueden usarse en una aplicación y simultáneamente pueden detectarse y seguirse 5 de ellos.

Con la versión 1.5 del SDK Vuforia, la anchura del marco ha sido cambiada, por lo que el desarrollador ha de asegurarse de actualizar las plantillas de los Frame Markers localizadas en el directorio *assets*\

Parámetros

- **Tamaño del marcador (Marker Size):** Como en los casos anteriores, el tamaño del Target es el tamaño real del marcador en unidades de escena 3D. El desarrollador necesita especificar este tamaño en el momento de su creación. El parámetro Size es importante, como la información de posición que es devuelta durante el seguimiento se relacionará con la misma escala. Por ejemplo si el marcador tiene una unidad de ancho, mover la cámara desde el borde izquierdo del Target hasta el borde derecho, la posición que se devuelve habrá cambiado una unidad a lo largo del eje X[9].
- **Identificador del marcador (Marker ID):** Cada Frame Marker codifica un ID en el patrón binario que recorre el borde del marcador. Este ID tiene un rango de [0..512] dando 512 posibles marcadores diferentes y objetos de Realidad Aumentada 3D asociados a ellos.
- **Tipo del marcador (Marker Type):** Enumeración que define el tipo de marcador, cada uno será:
  - INVALID: Marcador de tipo inválido.
  - ID_FRAME: Marcador cuya ID está codificada en su marco.

Ejemplo

Un ejemplo de marcador es el mostrado en la Figura 2.14, este marcador codifica ID=0.

Figura 2.14: Ejemplo de Frame Marker (FUENTE: Qualcomm)

El marcador consta de cuatro áreas, cada una de las cuales cumple una función determinada:

[9] (x, y) – Tamaño del marcador en unidades de escena medidas a lo largo de los ejes horizontal y vertical de un marcador

- Un área alrededor del marco negro del marcador debe dejarse libre de elementos gráficos. Esta área, es típicamente el doble de ancho que el marco negro y debería ser brillante para proporcionar un buen contraste con el marco.
- El marco negro se usa para reconocer el marcador en el entorno que es visto por la cámara en tiempo de ejecución.
- El área ID en el interior del marco codifica la ID del marcador en un patrón binario.
- El área de diseño (naranja) en el que el desarrollador puede incluir cualquier diseño para hacer el Frame Marker más atractivo visualmente y hacer que contenga información adicional para el usuario final. Esta área del marcador no es evaluada por el SDK Vuforia.

Aspectos a tener en cuenta

- **Tamaño físico** (Physical Size): El tamaño del marcador depende del entorno de la aplicación. Se recomienda que para escenarios de uso en campo cercano, el marcador real tenga unas dimensiones de 3-10 cm; estas dimensiones se refieren al tamaño del marcador impreso. El pequeño tamaño que se puede aplicar a los marcadores permite que se puedan emplear para ser usados como piezas o cartas de juegos.
- **Forma del Marcador** (Shape of the Marker): En el uso real, el marco o el patrón binario podría ser parcialmente obstruido, evitando así que el marcador trabaje correctamente. Para evitar este problema, es importante prestar especial atención al diseño del marcador impreso. A continuación se muestran algunos ejemplos de diseños físicos de marcadores que buscan evitar oclusiones causadas por el usuario.

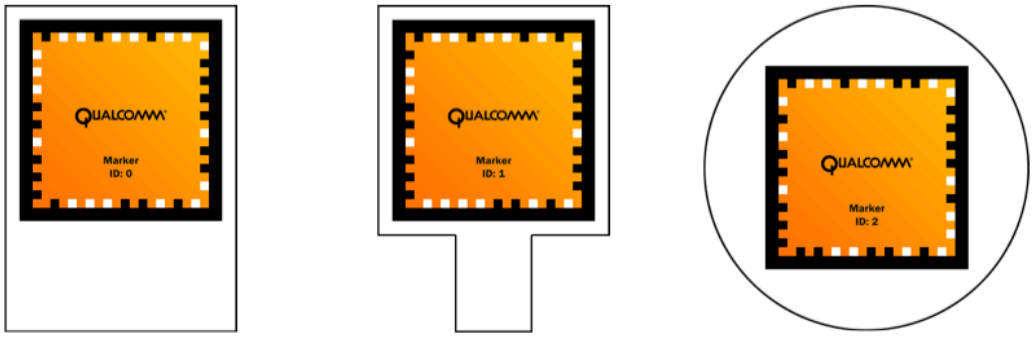


Figura 2.15: Ejemplos de forma de tres Frame Markers (FUENTE: Qualcomm)

- **Parámetro Tamaño del marcador** (Size Parameter): El tamaño del Marker puede establecerse arbitrariamente. Se recomienda que el tamaño sea definido

en unidades de escena, que se relacionan con el tamaño real. Por ejemplo, 1 unidad de escena es 1 cm en el mundo real. De esta manera la superposición de elementos virtuales sobre el entorno real es más fácil de conseguir.

- **Contraste de la imagen interna del diseño**: Toda la parte interior del marcador se puede diseñar libremente, siempre que la integridad de los bits (marco binario) no se vea comprometida. Se recomienda el uso de imágenes con alto contraste/brillo, patrones o rellenos que permitan la detección y seguimiento del marco y del patrón binario.
- **Contraste cercano al borde**: Un error común es recortar los marcadores por el marco negro del borde. El marco necesita integridad con un fondo brillante para que sea reconocido con precisión. Como regla general, el espacio libre alrededor del marcador debe ser el doble de ancho que el grosor del borde del propio Frame Marker.

2.1.2.3. Virtual Buttons

Los Botones Virtuales se tratan de regiones rectangulares, definidas por el desarrollador, que se encuentran en los Image Targets. Cuando son tocados u ocluidos en la visión proporcionada por la cámara, disparan un evento. Los Botones Virtuales se pueden emplear para implementar eventos como la pulsación de un botón o para detectar si áreas específicas del Image Target se cubren por un objeto. Sólo se evalúan si el área del botón se encuentra visible por la cámara y la cámara se encuentra estable. La evaluación de los Botones Virtuales se deshabilita durante movimientos bruscos de cámara.

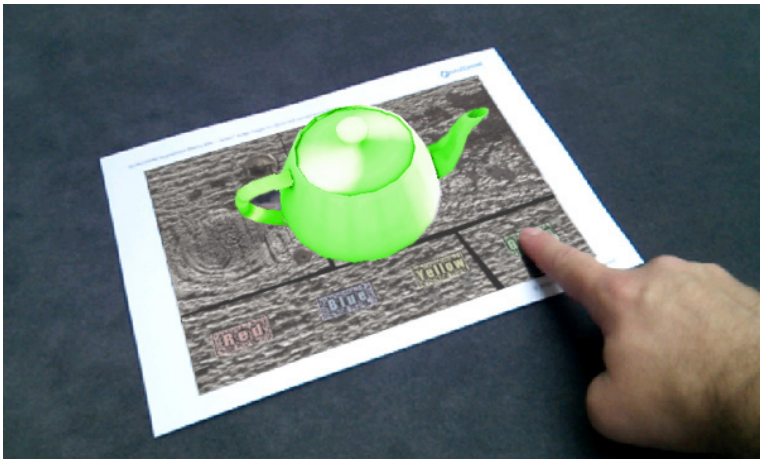


Figura 2.16: Ejemplo de cuatro Virtual Buttons sobre un Image Target en el que se encuentra una Realidad Virtual (FUENTE: Qualcomm)

Los Botones Virtuales se pueden crear, bien definiéndoles en el fichero XML de configuración como propiedad de Image Targets, o bien pueden ser añadidos o eliminados en tiempo de ejecución mediante un conjunto de APIs.

Parámetros

- **Nombre/identificador del botón (Button name/identifier):** String que identifica unívocamente al botón dentro del Target. Su longitud máxima es de 25 caracteres. A-z, A-Z, 0-9, [- _ .]
- **Coordenada (Button Coordinates):** Los botones se definen como regiones rectangulares. El desarrollador necesita especificar:
  - (x, y) de la esquina superior izquierda del rectángulo.
  - (x, y) de la esquina inferior derecha del rectángulo.

  Nótese que las unidades del área del botón están en unidades de escena 3D – el sistema de coordenadas local del Image Target. El origen del sistema de coordenadas es el medio (centro) del Image Target.
- **Sensibilidad del botón (Button sensivity):** "Sensibilidad" se refiere a la "respuesta" del botón:
  - HIGH: Rápida detección del evento de oclusión, el botón será muy sensible, pero puede disparar mayores falsos positivos.
  - MEDIUM: Compromiso entre una respuesta rápida y una detección robusta.
  - LOW: El botón necesita ser ocluido durante mayor tiempo para que se genere un evento.

Ejemplo

En la Figura 2.17 se muestra un ejemplo de Botones Virtuales sobre un Image Target. Consta de cuatro botones dispuestos horizontalmente en la parte inferior del mismo, y enmarcados con diferentes colores. Por ejemplo, presionando u ocluyendo un botón se podría cambiar el color de un objeto virtual.

Los botones tienen un tamaño de 200x100 pixeles en la imagen original de 1500x1050 pixeles. La esquina superior izquierda del botón rojo se sitúa en [90,850]; el origen es la esquina superior izquierda de la imagen.

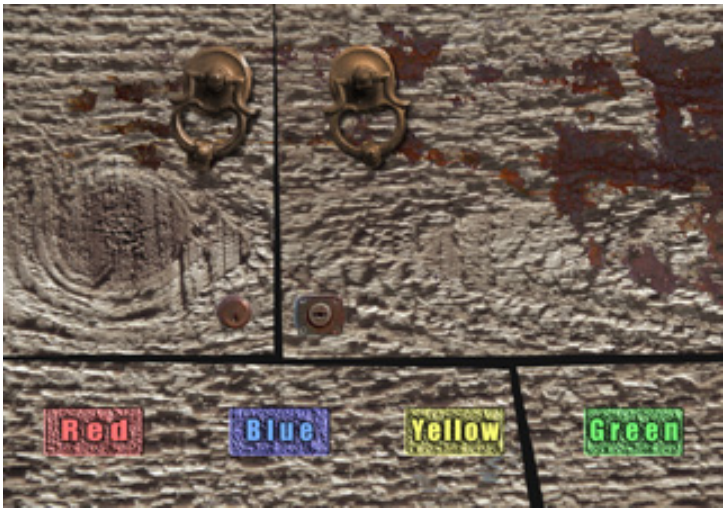


Figura 2.17: Ejemplo de cuatro Virtual Buttons sobre un Image Target (FUENTE: Qualcomm)

Las unidades del área de los botones están en unidades de escena 3D – El sistema de coordenadas local del Image Target. En este ejemplo se está usando un tamaño (Size)=[247, 173] como tamaño del Image Target en la escena real 3D – igual al tamaño impreso en milímetros.

Por lo tanto todos los tamaños y las coordenadas de ubicación deben escalarse en consecuencia. Los botones se definen con el formato de coordenadas *(esquina superior izquierda) (esquina inferior derecha)*:

Aspectos a tener en cuenta

- **Tamaño** (Size): Se deben elegir áreas en la imágenes que tengan unas dimensiones de, aproximadamente el 10% del tamaño del Image. Los botones más pequeños no se podrán identificar con facilidad si el Image Target aparece pequeño, bien porque así lo sea o bien se encuentre a gran distancia del objetivo de la cámara, en el campo de visión captado por la cámara.
- **Forma** (shape): Los usuarios finales deberían saber dónde se encuentran los botones dentro del Image Target. Por ello, se recomienda que los botones destaquen del resto de la imagen del Image Target.
- **Textura/contraste**: Los botones no deberían ser monocromáticos, sino que deberían proporcionar suficiente contraste para ser evaluados. En función de la aplicación, se debería elegir un diseño de botón que tenga una textura diferente que el objeto que causa la oclusión; no tiene por qué emplearse sólo el dedo para producir la oclusión.
- **Disposición dentro del Target**: Cuando se disponen los botones en un Image Target ha de tenerse en cuenta el uso que va a tener el usuario sobre el mismo, para evitar oclusiones involuntarias, etc. Además y en la misma línea, se ha de evitar que éstos se encuentren demasiado juntos.

Expuestas las características de Vuforia, es preciso conocer la arquitectura de la plataforma donde se ejecutará la aplicación a desarrollar, iOS de Apple.

2.1.3. iOS

iOS se trata de un sistema operativo de Apple Inc. Actualmente se usa en dispositivos como iPhone, iPad, iPod Touch y Apple TV, aunque originalmente fue desarrollado expresamente para el iPhone y recibía el nombre de *iPhone OS*. Apple Inc. no permite que iOS se ejecute en hardware de otros fabricantes. Este sistema operativo proporciona una interfaz de usuario basada en el concepto de manipulación directa sobre la pantalla, usando gestos multitáctiles[10] .

iOS deriva del sistema operativo de Apple, OS X, destinado a equipos de sobremesa, ordenadores portátiles, servidores, etc. Está desarrollado con lenguajes de programación tales como C, C++ y Objective-C. Posee un núcleo híbrido (XNU), Darwin BSD, y por lo tanto se trata de un sistema operativo de tipo Unix.

El lenguaje de programación principal en iOS, además de en OS X, es Objective-C. Se trata de lenguaje orientado a objetos creado como un superconjunto de C para que implementase un modelo de objetos similar al que emplea Smalltalk[11].

Como se trata de una capa muy fina que está por encima de C y además es un superconjunto estricto de C; un compilador de Objective-C es capaz de compilar cualquier programa escrito en C, y también se puede escribir código en C dentro de una clase de Objective-C. Por ello es habitual que en el código de aplicaciones iOS se encuentren partes escritas en Objective-C, C, C++…

iOS presenta cuatro capas de abstracción: Core OS (capa del núcleo del sistema operativo), Core Services (capa de servicios principales), Media (capa de medios), y Cocoa Touch (gestión de la interfaz de usuario).

Cada una de estas capas están formadas por una serie de frameworks, que permiten a los desarrolladores diferentes posibilidades y funcionalidades para implementar aplicaciones para la plataforma.

[10] Esto aplica a dispositivos iPhone, iPad e iPhone Touch; ya que la interfaz que proporciona el Apple TV, está pensada para ser usada con un control remoto de televisión

[11] Lenguaje de programación que permite realizar tareas de computación mediante la interacción con un entorno de objetos virtuales

- **Core OS**: Proporciona las características de bajo nivel: gestión de hilos, matemática compleja, ficheros del sistema, manejo de memoria, seguridad, drivers del dispositivo, etc. mediante una serie de frameworks.
- **Core Services**: Proporciona los servicios fundamentales del sistema que usan todas las aplicaciones.
- **Media**: Provee de la tecnología usada para soportar trazados 2D y 3D, audio y video. Basada en C y Objective-C
- **Cocoa Touch**: es la capa más importante para el desarrollo de aplicaciones iOS. Proporciona un conjunto de frameworks que proporciona el API de Cocoa para desarrollar aplicaciones. Cocoa Touch deriva de Cocoa, el API empleado en la plataforma MAC.

2.2. Requisitos

Los requisitos establecidos para la realización de este proyecto tienen que ver tanto con la plataforma de ejecución de la aplicación así como con la funcionalidad de la propia aplicación.

En cuanto a los requerimientos de funcionalidad solicitados; se precisa de una aplicación que ofrezca al estudiante un sistema de ayuda en el Campus junto con folletos impresos tradicionales de información sobre la Universidad haciendo uso de Realidad Aumentada, a modo de prueba de concepto.

El fundamento de esta aplicación es el uso conjunto con un folleto impreso físico; por ello y teniendo en cuenta que se trata de una prueba de concepto de una funcionalidad concreta (Realidad Aumentada), se solicita una interfaz de usuario sencilla y que sea auto-configurable en función de la página del folleto físico que el alumno esté visualizando. La intención del cliente es realizar una prueba piloto con esta tecnología, para si tiene éxito, solicitar la implementación de una aplicación más avanzada y/o incorporar estas nuevas funcionalidades a algunas de sus aplicaciones ya disponibles.

Se requiere que la aplicación proporcione al estudiante información institucional de la propia Universidad, información académica (horarios, aulas, etc.) e información de servicios e infraestructuras (deporte, cultura, instalaciones, etc.)

La aplicación debe ser implementada para ser ejecutada, al menos, sobre la plataforma iOS de Apple, tanto para la versión 4 como para la 5 del sistema operativo en dispositivos iPhone 4/4S.

2.3. Restricciones y Marco Regulador

Para la implementación de la solución requerida por el cliente, éste ha solicitado que no se empleen SDKs cuyo uso implique el pago de licencias, no importándole si se trata de código libre o no, mientras éste sea gratuito. Por lo tanto el SDK Vuforia de Qualcomm cumple esta restricción, puesto que aunque se trate de una solución de código cerrado, no es preciso pagar una licencia para poder utilizarlo, si bien hay que respetar el Contrato de Licencia de Vuforia[12].

La intención del cliente es poner a disposición de la Comunidad Universitaria esta aplicación, mediante descarga gratuita a través de la plataforma de distribución de software para dispositivos portátiles de Apple, App Store; bien como aplicación dedicada o bien como un complemento de funcionalidad a alguna de sus aplicaciones disponibles.

Apple impone una serie de restricciones a las aplicaciones para que puedan ser publicadas en su sistema de distribución de aplicaciones, estas restricciones tienen que ver con aspectos de diseño, éticos, aceptación de términos y condiciones, el cumplimiento de la legislación vigente, etc. Por ello, se deben de tener en cuenta y respetar dichas restricciones durante el desarrollo de la aplicación, para que en la fase de supervisión de la aplicación, Apple apruebe el software y lo publique.

Será necesario cumplir las restricciones de Apple, para cumplir uno de los requisitos del cliente que es que la aplicación pueda publicarse en el App Store.

Apple pone a disposición de los desarrolladores una serie de directrices que sirven de ayuda para conocer qué aspectos se tienen en cuenta para aceptar/denegar aplicaciones en el App Store. Todas las directrices se encuentran disponibles en el sitio web de Apple[13]; si bien, a continuación se describen las directrices que aplican de forma más significativa al desarrollo del presente Proyecto. Se hará referencia al número de directriz específico entre paréntesis.

- **En el capítulo de Términos y Condiciones:** Como desarrollador de aplicaciones para el App Store, se tiene la obligación de aceptar los términos del Programa de Acuerdo de Licencia[14], de las Directrices de Interfaz Humana[15] y cualquier otro tipo de licencias o contratos entre el desarrollador y Apple.

12 "Vuforia License Agreement for iOS" disponible en https://ar.qualcomm.at/legal/license/

13 https://developer.apple.com/appstore/resources/approval/guidelines.html (Inscripción necesaria)

14 "Program License Agreement (PLA)" disponible en https://developer.apple.com/membercenter/index.action

15 "Human Interface Guidelines (HIG)"

- **En el capítulo de Funcionalidad:** Se rechazarán aquellas aplicaciones que presenten bugs (2.2) y cuya ejecución se interrumpa y se cierren de forma inesperada (2.1). Se rechazarán aquellas aplicaciones que hagan uso de APIs no públicas. (2.5). Las aplicaciones de iPhone deben ejecutarse, sin necesidad de ser modificadas, también en un iPad, con la resolución de iPhone, y con el doble de resolución de un iPhone 3GS. (2.10). Las aplicaciones para navegar por internet o que presenten esta funcionalidad, deben usar el framework iOS WebKit y WebKit Javascript (2.17). Las aplicaciones deben seguir las Directrices de Almacenamiento de Datos en iOS[16], sino se rechazarán (2.23).
- **En el capítulo de Marcas registradas e imagen comercial:** Las aplicaciones deben cumplir con todos los términos y condiciones explicados en la Guía para el uso de marcas y derechos de autor de Apple[17] y la Lista de marcas comerciales de Apple[18] (8.1).
- **En el capítulo de Privacidad:** Las aplicaciones no pueden transmitir datos de un usuario sin obtener permiso previo de éste y proporcionarle acceso a la información sobre cómo y dónde serán utilizados los datos (17.1)
- **En el capítulo de Requisitos legales:** Las aplicaciones deben cumplir con todos los requisitos legales en cualquier lugar donde se ponen a disposición de los usuarios. Es obligación del desarrollador entender y cumplir con todas las leyes locales. (22.1) Es por ello que en el caso de España hay que respetar la Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal[19]. Esta Ley Orgánica tiene por objeto garantizar y proteger, en lo referente al tratamiento de datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, especialmente su honor, intimidad y privacidad personal y familiar. Su objetivo principal es regular el tratamiento de los datos y ficheros, de carácter personal, independientemente del soporte en el cual sean tratados, los derechos de los ciudadanos sobre ellos y las obligaciones de aquellos que los crean o tratan.

Finalmente y a modo de resumen, decir que tanto los diferentes Términos y Condiciones ofrecidos por Qualcomm y Apple; así como las leyes locales a las que remiten, en este caso la legislación española (LOPD); conforman los requisitos y el marco regulador del presente Proyecto.

16 "iOS Data Storage Guidelines" disponible en https://developer.apple.com/icloud/documentation/data-storage/

17 "Guidelines for Using Apple Trademarks and Copyrights", disponible en http://www.apple.com/legal/trademark/guidelinesfor3rdparties.html

18 "Apple Trademark List", disponible en http://www.apple.com/legal/trademark/appletmlist.html

19 Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal (LOPD), disponible en http://www.boe.es/aeboe/consultas/bases_datos/doc.php?coleccion=iberlex&id=1999/23750

3. Diseño de la solución técnica

3.1. Procedimiento

Para la creación de una aplicación basada en Realidad Aumentada haciendo uso del SDK Vuforia de Qualcomm, su desarrollo se puede realizar siguiendo dos procedimientos bien diferenciados, a través de Xcode o mediante una extensión de Unity. A continuación se describirán estas dos formas de proceder y se argumentará el por qué de la elección de una de ellas.

La primera forma de proceder, consiste en emplear el entorno de desarrollo Xcode para programar la aplicación del modo tradicional, igual que se hace habitualmente para la creación de aplicaciones para iOS: teniendo en cuenta su arquitectura, su lenguaje de programación principal Objective-C, su API y frameworks, etc. Junto con ello, el empleo del SDK Vuforia, que establece su propio API en C++, arquitectura, etc.

La segunda de ellas es a través de Unity. Unity es un Motor 3D, propiedad de Unity Tecnologies, que se emplea para el desarrollo de videojuegos para Windows, OS X, iOS, Android, Xbox 360, PlayStation 3, y Wii.

Qualcomm ofrece una extensión para Unity que permite crear aplicaciones basadas en Aumentada con Vuforia para dispositivos iOS. Si bien, como se ha dicho en apartados anteriores el SDK Vuforia es gratuito, al igual que la extensión para el Motor 3D; para poder cargar aplicaciones en iOS, Unity requiere de una licencia de pago[20]. Además, la programación de la aplicación se realiza a través de Unity, que actúa como una capa de abstracción por encima de Objective-C, de la arquitectura de iOS y del API en C++ de Vuforia. La programación en Unity se realiza de un modo más visual, situando posiciones de cámara, planos (Targets), Realidades Virtuales sobre los Targets, etc. a modo de un editor 3D convencional, combinada con lenguajes de programación JavaScript, C# o un dialecto de Python llamado Boo.

De estas dos maneras de proceder, Qualcomm recomienda la programación a través de la extensión de Unity debido a que, como se dijo antes, se evita la programación directa contra Objective-C, conocer la arquitectura de iOS, etc. Haciendo que esta forma de programar sea, a priori, más sencilla. Además, como Unity es un Motor 3D, se puede dotar a los Modelos 3D, que se presentarán como Realidades Virtuales, de animaciones complejas, empleo de física de partículas

[20] Unity Pro (1.500$) + iOS Pro complement (1.500$)

para la representación de elementos tales como agua, humo, fuego, etc. y en definitiva todas las posibilidades de modelado y animación 3D disponibles en los videojuegos comerciales actuales.

Sin embargo, para la realización de este Proyecto, se ha optado por la programación directamente contra código en Xcode, haciendo uso del API y frameworks de Apple, así como con el SDK Vuforia. El fundamento de esta elección radica en uno de los objetivos que se quieren alcanzar con este Proyecto y que consiste en aprender cómo es la arquitectura y funcionamiento de iOS, así como aprender a programar en su lenguaje de programación Objective-C. El hándicap que presenta esta forma de proceder, a parte de la dificultad extra que supone, se produce en la representación de las Realidades Virtuales; y es que si se sigue este procedimiento; las animaciones de los Modelos 3D sólo pueden ser animaciones simples: escalados, translaciones y rotaciones del Modelo completo. A pesar de ello, se ha considerado más provechosa esta forma de proceder, por lo expuesto anteriormente.

Puesto que comenzar a programar una aplicación basada en Vuforia desde cero es muy complicado y no hay suficiente documentación para realizarlo; Qualcomm pone a disposición de los desarrolladores varias aplicaciones simples de ejemplo con su código fuente disponible para que, a partir de ellas y a modo de plantilla, comenzar a programar aplicaciones propias sea más sencillo. Gracias a estas aplicaciones de ejemplo se puede comprobar cuáles son la posibilidades básicas y características que proporciona este SDK.

La primera de estas aplicaciones de ejemplo, *ImageTargets*, plantea sobre un Image Target el Modelo 3D de una tetera. La aplicación *FrameMarkers* sitúa las letras Q, C, A y R en 3D sobre los Frame Markers con ID 0, 1, 2 y 3 respectivamente; pudiendo representar las cuatro letras de forma simultánea si se encuentran los cuatro marcadores fiduciales en el campo de visión de la cámara. La tercera aplicación proporcionada por Qualcomm, llamada *MultiTargets*, representa el Modelo 3D de un bol de cereales girando alrededor de un Multi Target. La aplicación *VirtualButtons,* emplea un Image Target que consta de cuatro Botones Virtuales para representar el Modelo 3D de la misma tetera que en *ImageTargets*; la diferencia radica en que cuando se pulsa alguno de esos cuatro Botones Virtuales, el color de la tetera cambia. Por último, en la aplicación *Dominoes* que emplea un Image Target, el usuario ha de tocar la pantalla física del dispositivo, sobre dicho Image Target, para ir colocando fichas de domino 3D sobre el Target para generar un efecto dominó. Para comenzar a tirar las fichas el usuario puede "empujar" la primera de ellas desde el mundo real (cuando el usuario pone la primera ficha de dominó sobre el Target, en esa posición se crea un Botón Virtual,

que más adelante permitirá que se pueda empujar dicha ficha) o bien mediante la pulsación de un botón que aparece en la interfaz de la pantalla del dispositivo.

Existen dos aplicaciones más: *BackgroundTextureAccess* y *OcclusionManagement*, pero la funcionalidad que ofrecen no se utilizará para la realización de este Proyecto.

En este punto, conocida la arquitectura de iOS y todo lo que involucra (conocer cómo funciona Xcode, Objective-C, etc.), la arquitectura del SDK Vuforia y la funcionalidad con la que se quiere dotar a la aplicación a desarrollar, y visto el funcionamiento de las aplicaciones de ejemplo; se comienzan ha realizar labores de Ingeniería Inversa sobre el código fuente de estas aplicaciones para comprender cómo es su funcionamiento desde la perspectiva del código.

3.2. Importación de Modelos 3D

Lo primero que se intenta averiguar es cómo se importan los Modelos 3D a la aplicación para representar las Realidades Virtuales. Para ello se parte del código fuente de la aplicación de ejemplo más sencilla, *ImageTargets*, y se intenta buscar el o los ficheros que permiten que se represente la tetera en 3D. Se esperaba encontrar algún tipo de fichero que contuviera el Modelo 3D completo en algún formato tradicional de los empleados para guardar figuras 3D con algún editor de modelado 3D existente en el mercado; pero lo que se encontró fue un fichero de cabecera (.h) que contiene infinidad de datos numéricos almacenados en diferentes arrays de vértices, aristas, etc.

En el código, se importa este fichero de cabecera y a partir de unas funciones de OpenGL ES, que se explicarán con más detalle más adelante, se representa el Modelo 3D. Además de este fichero de cabecera, existe una imagen cuadrada en formato PNG, de lo que parece ser la textura del Modelo.

Llegados a este punto, se intenta documentarse o buscar algún tipo de procedimiento para la creación e importación de Modelos 3D a las aplicaciones Vuforia, para intentar sustituir la figura de la tetera con una creada personalmente en el ejemplo *ImageTarget*. Lamentablemente, Qualcomm no ofrece ningún documento donde se explique o documente un procedimiento a tal efecto, por lo que se tuvo que investigar en el foro de desarrolladores de Qualcomm.

En un primer momento, se encontró una solución que hacía uso de un script escrito en Perl, creado por Heiko Behrens. Este script, llamado obj2opengl, convierte Modelos 3D almacenados en ficheros OBJ (.obj) en arrays compatibles con la versión de OpenGL ES que se encuentra en el iPhone.

El formato de fichero OBJ, desarrollado por Wavefront Technologies, es de código abierto y ha sido adoptado por multitud de editores 3D del mercado. Este formato de ficheros no es más que un documento de texto plano en el que, con un formato determinado, se representa únicamente la geometría 3D; a saber, la posición de cada vértice, la posición UV de cada vértice de coordenadas de textura, normales, y las caras que hacen cada polígono definido como una lista de vértices, y vértices de textura.

Por tanto, el script obj2opengl se encarga de transformar la información almacenada en este tipo de ficheros a un fichero de cabecera, con la estructura de datos con la que es capaz de trabajar OpenGL ES.

Con todo lo anterior en conocimiento se concluye que, para importar un Modelo 3D se necesita de un fichero que contenga únicamente la información tridimensional del Modelo 3D, es decir, su malla; y de una imagen que contenga la textura para vestir a dicho Modelo 3D. Para probar el script obj2opengl, se procedió a descargar de un sitio web Modelos 3D gratuitos y libres de derechos de autor[21].

Antes de emplear el script, se instaló el editor 3D Cheetah 3D para OS X, con el primer objetivo de visualizar los Modelos 3D descargados, y con el objetivo posterior de crear Modelos 3D propios en etapas más avanzadas del desarrollo del Proyecto. Cuando se descargaron varios Modelos 3D en formato OBJ, se comprobó que no todos se presentaban la dupla fichero OBJ-imagen de textura, que es lo que precisa Vuforia. En muchos de ellos, además del fichero OBJ propiamente dicho, en lugar de una textura PNG, se encontraba un fichero MTL (.mtl). Este tipo de fichero, cuyas siglas provienen de Material Template Library (Biblioteca de plantillas de materiales) es un formato de archivo complementario a OBJ que describe las propiedades de sombreado de superficies (material) de objetos contenidos en uno o más ficheros OBJ. Un fichero OBJ referencia a uno o más ficheros MTL (llamados "bibliotecas de materiales"), y desde allí, referencia a una o más descripciones de materiales por nombre. El problema es que no hay forma de que Vuforia trate con ficheros MTL, por ello las primera pruebas se realizaron con Modelos 3D que estuvieran compuestos por malla (OBJ) y textura (fichero imagen).

Para realizar las pruebas, se transformaron todos los Modelos 3D en formato OBJ a ficheros de cabecera (.h) empleando el script obj2opengl, se importó tanto el fichero de cabecera como la textura correspondiente a ese Modelo desde el código y se comprobó el funcionamiento para ver si los Modelos se renderizaban correctamente en la aplicación. Los resultados fueron desiguales; ya que, a pesar

[21] http://free.turbosquid.com/

de proceder del mismo modo con todos los Modelos, unos se representaban correctamente, mientras que otros no. El problema parecía estar acotado, ya que todos los Modelos que no se renderizaban adecuadamente, parecían presentar los mismos fallos. Estos fallos se manifestaban como polígonos triangulares que no se representaban en el Modelo final en una proporción de 50% de polígonos mostrados y 50% de polígonos no mostrados y que, aunque la percepción espacial del Modelo se intuía perfectamente, les daba la apariencia de un colador. Se investigaron cuáles eran las causas de este fenómeno, concluyéndose que la versión de OpenGL ES existente en el iPhone, para representar correctamente los Modelos 3D, necesita que las caras (polígonos) de los Modelos 3D sean triangulares.

Para entender esto se expone un ejemplo muy sencillo. Supóngase que se quiere representar un plano cuadrado en tres dimensiones. Para hacer esto, el plano se puede modelar como una cara cuadrangular o como dos caras triangulares, pues bien; si se pasa el script obj2opengl al plano formado por una sola cara cuadrangular, al verlo renderizado en la aplicación ejecutándose en el iPhone, éste se verá como un triángulo (al plano completo le faltaría la mitad); mientras que si se le pasa el plano formado por dos caras triangulares, el plano se visualizará correctamente. Este sencillo ejemplo se puede extender a Modelos mucho más complejos y de ahí la apariencia de colador que adoptan los Modelos formados por polígonos cuadrangulares al faltarles la mitad de cada cara.

Para solucionar esto, habría que abrir el Modelo 3D en el editor (en este caso Cheetah 3D para OS X) e ir cambiando cada una de las caras cuadrangulares para transformarla en dos caras triangulares. Por esta razón se desestimó el script obj2opengl para que formara parte del proceso de importación de Modelos 3D a la aplicación y se buscaron otras alternativas.

Se encontró una alternativa diferente que supuso la instalación del software de modelado tridimensional Blender, el cual es gratuíto y de código abierto, y de un plug-in para éste llamado *OpenGL / C header exporter*, también de carácter gratuito. Este plug-in, permite exportar cualquier clase de formato de archivo que es capaz de abrir Blender (.dae, .3ds, .obj, etc.) en un fichero de cabecera (.h) compatible con OpenGL ES. Además de la ventaja frente al script de Heiko Behrens que supone el poder exportar mayor cantidad de tipos de fichero, hay que sumarle el hecho de que este plug-in exporta los Modelos transformando todas las caras del mismo a polígonos triangulares si es que no estuvieran triangularizados previamente. Por todo esto, se ha utilizado esta solución para la creación de los ficheros de cabecera de los Modelos 3D que se utilizarán en la aplicación a desarrollar.

En este punto, ya se es capaz de importar la información referente a la malla del Modelo 3D en la aplicación. Como se dijo anteriormente, para que el Modelo se pueda mostrar es necesario que, además de la malla, exista una imagen de textura para pintar el Modelo. Las pruebas se realizaron usando una imagen con un color plano como textura, con el propósito de comprobar únicamente que la malla, es decir, el volumen del Modelo 3D; se renderizaba correctamente. Así los Modelos 3D se veían sin textura; sólo con un color plano. Por tanto, era momento de abordar el problema de cómo texturizar un Modelo 3D, a partir de una imagen plana.

El problema que se plantea es cómo proyectar una imagen plana, de dos dimensiones, sobre un cuerpo tridimensional para recubrirlo. Para solucionar este problema, se emplean los mapas UV. Los UV son coordenadas que se asignan a cada uno de los vértices de un Modelo 3D. Un vértice del Modelo 3D, además de tener tres coordenadas (X, Y, Z) referidas a su posición espacial, también disponen de dos coordenadas más que sirven para determinar su posición en un plano bidimensional que representa la textura. Se podría considerar que los UV son como puntos ocultos que pueden manipularse solamente en un espacio bidimensional. Los UV pueden estar separados de sus caras, a diferencia de los vértices; esto permite crear patrones para aplicar las texturas más adelante. En definitiva, los UV determinan cómo va a proyectarse una textura sobre el Modelo 3D. Los UV se han de definir de forma manual, para evitar la distorsión de la textura. Para ello se tienen que crear shells de UVs, que representen un lado del Modelo a la vez. Una forma de crear estos shells es a través de los seams, que no son más que los bordes de cada shell de un Modelo tridimensional. Por ejemplo, si se tiene un cuadrado el shell sería el propio cuadrado, su área; y los seams, el perímetro que lo delimita, sus aristas.

Para proceder a texturizar un Modelo 3D, y a modo de resumen; se ha de partir de su malla y delimitarla completamente seleccionando de forma manual los seams deseados, para mapear el Modelo 3D y que se genere el mapa UV correspondiente donde pintar la textura en un editor gráfico.

Hasta el momento se conoce cómo diseñar desde cero un Modelo 3D (o a partir de uno ya modelado), texturizarlo y prepararlo para que pueda ser importado correctamente por la aplicación basada en Vuforia para un renderizado adecuado. Por tanto, se está en disposición de crear un procedimiento básico para la creación y/o adaptación de Modelos tridimensionales compatibles con el SDK Vuforia de Qualcomm.

3.3. Procedimiento básico de creación/importación de Modelos 3D

A continuación se expone de forma detallada, mediante un ejemplo práctico, cuál ha sido el procedimiento básico de creación y/o adaptación de Modelos 3D que se ha empleado en la realización de este Proyecto. Las herramientas empleadas son:

- Software de modelado 3D:
  - Cheetah 3D
  - Blender (más plug-in *OpenGL / C header exporter*)
- Software de edición gráfica:
  - Adobe Photoshop CS 6

El ejemplo práctico consistirá en realizar el Modelo 3D del escudo de la Universidad Carlos III de Madrid que se utiliza en la aplicación final de este Proyecto:

PASO I: Creación del Modelo en Cheetah 3D

En la primera fase se creará la malla del Modelo 3D, en este caso, partiendo de un cilindro que se adaptará de forma que pueda contener el escudo de la Universidad.

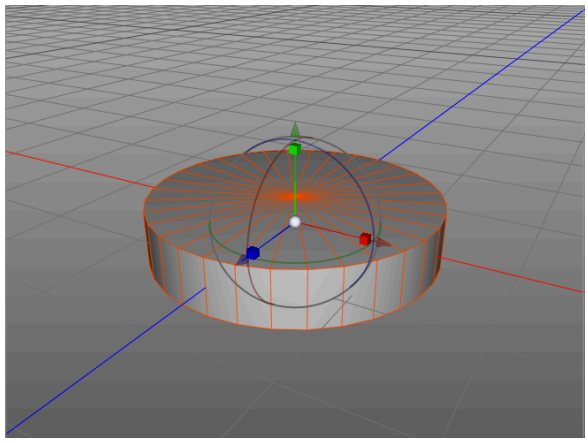


Figura 3.1: Malla de un Modelo 3D en el editor Cheetah 3D

PASO II: Creación del Mapa UV para texturizar el Modelo 3D

Como el Modelo se desea texturizar de modo que sobre las dos caras circulares del cilindro aparezca el escudo de la Universidad, y puesto que éste también es circular; con la herramienta de selección de aristas activada, se seleccionarán las aristas que delimitan la circunferencia inferior y superior del cilindro, así como

una arista de la parte curva del cilindro para realizar el mapeado UV de forma óptima. Para ello, con las aristas indicadas marcadas; se selecciona la opción Selection>Toggle seam. En la siguiente figura se puede comprobar en color verde los seams.

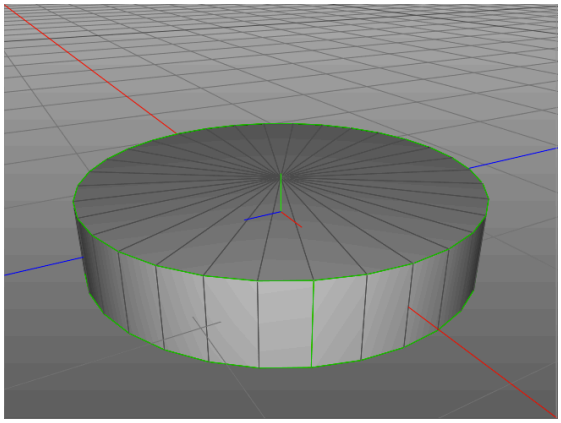


Figura 3.2: Seams sobre la malla de un Modelo 3D

Con los seams definidos, se cambia la vista del editor 3D al modo UV Editing para generar el mapa UV. Para ello se selecciona la opción Unwrap UV sobre el espacio de edición, así se genera el mapa UV de acuerdo a los seams que se han seleccionado y que se pueden adaptar y reorganizar como se desee.

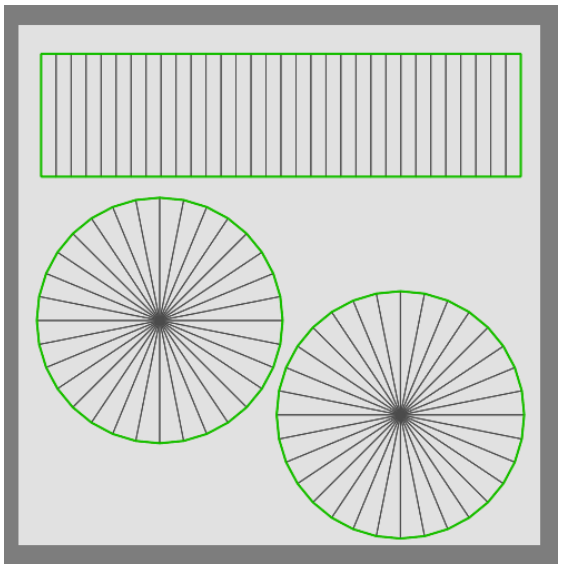


Figura 3.3: Mapa UV de un Modelo 3D

PASO III: Creación de la textura en Adobe Photoshop.

Partiendo del mapa UV recién creado, se crea la textura que pintará el Modelo 3D. Una forma sencilla de proceder para situar y ceñirse del mejor modo al mapa

UV es dejar el mapa como capa superior y hacerla semitransparente para editar la textura en una capa por debajo de ésta. Una vez diseñada la textura se guarda en formato PNG.

Figura 3.4: Textura de un Modelo 3D generada a partir del Mapa UV correspondiente

PASO IV: Aplicación de la textura al Modelo 3D.

Una vez creada la textura en Photoshop, de vuelta a Cheetah 3D, se aplicará la textura sobre el Modelo para comprobar que su renderizado es correcto. Para ello, se ha de crear un nuevo material a partir de la textura que se ha creado y aplicárselo al Modelo. Material>Add Material>Default>Material. Una vez hecho esto, en Diffuse se selecciona Textures>Image y se carga la textura que se acaba de crear. Por último se le aplica al Modelo, arrastrando el material sobre el cilindro.

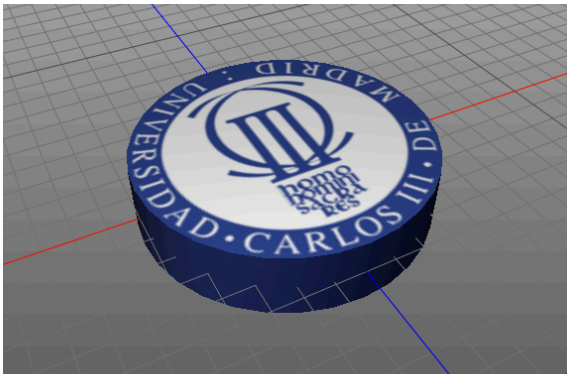


Figura 3.5: Vista de un Modelo 3D sobre el que se aplica una textura

Una vez comprobado el renderizado, se guarda el Modelo 3D en formato .3ds para su posterior transformación a fichero de cabecera (.h)

PASO V: Adaptación del Modelo 3D a fichero de cabecera, usando Blender.

En este último paso, se empleará Blender para transformar el Modelo 3D a un fichero .h compatible con Vuforia gracias al plug-in *OpenGL / C header exporter*. Para ello, se abre Blender y se importa el Modelo guardado con Cheetah 3D: File>Import>3D Studio (.3ds). Una vez abierto, se selecciona el elemento y se exporta a fichero .h: File>Export>OpenGL C include (.h)

Una vez realizados los 5 pasos, se tienen dos ficheros correspondientes a la malla del Modelo 3D y su textura, listos para ser importados desde el código de la aplicación en Xcode.

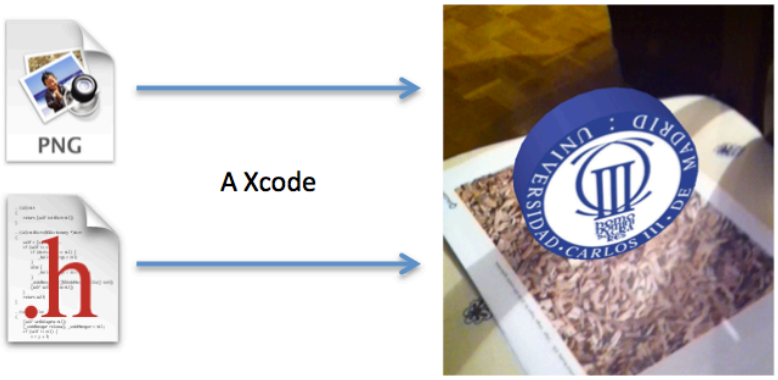


Figura 3.6: Ficheros necesarios para Renderizar un Modelo 3D en Vuforia

3.4. Estructura de la aplicación

Del mismo modo en que las pruebas para la importación de nuevos Modelos 3D se realizaron a partir de la aplicación de ejemplo más sencilla, *ImegeTargets*; para la implementación de la aplicación final del Proyecto también se tomó esta aplicación como punto de partida.

Para explicar la estructura de ficheros y demás componentes de la aplicación, dado que consta de gran cantidad de clases, ficheros y documentos; en primer lugar se listarán todos ellos y se dará una breve descripción de su cometido en la aplicación. Después, se incidirá más profundamente en aquellos ficheros en los que su programación se ha tenido que modificar más profundamente para dar cuerpo a la funcionalidad de la aplicación final, explicando su estructura y funcionamiento. Por último, se citarán de forma breve el resto de ficheros necesarios para la correcta ejecución de la aplicación.

Aunque para desarrollar la aplicación final se haya tomado la aplicación *ImageTargets* como plantilla; tanto ésta como el resto de aplicaciones de ejemplo, comparten una serie de ficheros comunes, clases, que se alojan en el directorio

ARCommon, y de los que heredarán muchas de las clases de la. Alguno de estos componentes han tenido que ser reescritos para satisfacer características de la aplicación a desarrollar. Éstos son:

- **ARParentViewController** (.h/.m): Definición e implementación del View Controller principal de la aplicación. Se encarga de añadir la EAGLView y la Overlay view a la Ventana Principal, manejadores de eventos táctiles, etc. No se ha modificado ningún aspecto de las mismas para la realización del Proyecto.
- **ARViewController** (.h/.mm): Gestiona el ciclo de vida de la cámara y de las Realidades Aumentadas, llamando a QCAR:createAR, QCAR:destroyAR, QCAR:pauseAR y QCAR:resumeAR cuando sea necesario. Asimismo, gestiona los datos necesarios para la Vista, como la carga de texturas que se requieren para pintar las Realidades Aumentadas. Además se encarga de tratar los cambios de orientación del dispositivo rotando la Vista como proceda.
- **AR_EAGLView** (.h/.mm): Superclase que contiene la configuración de OpenGL para su sub-clase, EAGLView. La clase AR_EAGL envuelve el CAEAGLLayer de CoreAnimation en una subclase de UIView. El contenido de la Vista es básicamente una superficie EAGL donde se reproduce la escena OpenGL. Este aspecto no se ha modificado.
  Aquí se define la clase Object3D, que se trata de la estructura que apuntará a un objeto para que se renderice; es decir, un objeto de esta clase se corresponderá con un objeto 3D que se presentará al usuario como Realidad Aumentada.
  Esta clase hereda de NSObject (clase base de un gran número de librerías que se utilizan tanto en OS X como en iOS) y ha sido modificada para satisfacer las características de los ficheros de cabecera de los diferentes Modelos 3D creados siguiendo el procedimiento del apartado anterior:

```
@interface Object3D : NSObject {
    unsigned int numVertices;
    const float *vertices;
    const float *normals;
    const float *texCoords;

    unsigned int numIndices;
    const unsigned short *indices;

    Texture *texture;

    unsigned int faces_count;
    float scale;
    int rotation;
                }
 @property (nonatomic) unsigned int
numVertices;
```

Figura 3.7: Estructura de la clase Object3D

En los atributos de esta clase se guardan el número de vértices del Modelo 3D, los vértices, normales, coordenadas de textura (mapa UV), el número de índices e índices, así como la textura del Modelo, su número de caras, la escala y un flag que servirá para indicar si el Modelo ha de aparecer rotando o no, respectivamente.

- **OverlayViewController** (.h/.mm): Definición e implementación de la clase OverlayViewController. Se encarga de la gestión del ActionSheet de configuración de la cámara así como de la disposición de los elementos en pantalla cuando se gira el dispositivo. No se ha modificado ningún aspecto de las mismas para la realización del Proyecto.
- **OverlayView** (.h/.mm): Definición e implementación de la clase OverlayView. Se encarga de mostrar los elementos de la interfaz de usuario sobre otra Vista. No se ha modificado ningún aspecto de las mismas para la realización del Proyecto.
- **QCARutils** (.h/.mm): Contiene todo el código que inicializa y maneja el ciclo de vida de QCAR (Vuforia); además de algunas funciones útiles para acceder a los Targets, etc. Se trata de una clase singleton que hace que QCAR sea accesible desde cualquier lugar dentro de la aplicación. No se ha modificado ningún aspecto de las mismas para la realización del Proyecto.
- **ShaderUtils** (.cpp/.h): Aquí se definen una serie de métodos que servirán de ayuda para la manipulación de las matrices que representan la posición de los objetos 3D sobre la escena: Translaciones, rotaciones, escalados, etc. No se ha modificado ningún aspecto de las mismas para la realización del Proyecto.
- **Texture** (.h/.mm): Definición e implementación de la clase Texture. En ella se definen una serie de atributos y métodos necesarios para que, partiendo de un fichero de imagen (JPG, PNG), crear una textura y podérsela aplicar de forma conveniente al Objeto 3D. No se ha modificado ningún aspecto de las mismas para la realización del Proyecto.

Una vez vistos los componentes comunes a todas las aplicaciones Vuforia, se citarán los componentes propios de la aplicación final:

- **ImageTargetsAppDelegate** (.h/.mm): Se encarga de gestionar el ciclo de vida de la aplicación de acuerdo al ciclo de vida de las aplicaciones bajo la arquitectura iOS. Se controla si la resolución de la pantalla del dispositivo es *retina* (iPhone 4/4S), mantiene la pantalla de carga de la aplicación hasta que la cámara esté inicializada, etc.
  Las modificaciones necesarias en estos ficheros tienen que ver con especificar a la aplicación qué datasets han de cargarse para poder detectar y realizar el seguimiento de las diferentes páginas del folleto impreso.

- **EAGLView** (.h/.mm): Esta clase es la más determinante puesto que es la encargada de la importación de los diferentes Modelos 3D y su renderización, tanto estática como animada; en función de los Targets que se estén visualizando en el momento. Es en esta clase donde se ha producido la mayor parte del trabajo de este Proyecto y la que más tiene que ver con la arquitectura de Vuforia propiamente dicha. Por ello, se explicará con más detalle esta clase en el siguiente apartado.
- **DomParentViewController** (.h/.mm): Se trata de una sub-clase de ARParentViewController que se encarga de pasar los eventos táctiles que se producen sobre la Ventana a la clase de la que hereda.
- **ButtonOverlay** (.h/.mm/.xib): Definición, implementación y fichero de Interface Builder de la clase ButtonOverlay. Aquí se configuran todos los elementos de la interfaz gráfica: botones, etiquetas, imágenes, WebViews, ActionSheets, cuadros de diálogo, así como los eventos multitáctiles a los que responderán y a que métodos han de llamar cuando éstos se produzcan. Además se definen una serie de métodos que afectan a la interfaz de usuario, mostrando u ocultando botones, etiquetas, etc. que sirven para adaptar la interfaz gráfica al contexto de utilización de la aplicación.
- **Dominoes** (.h/.mm): Gestiona de la interfaz de usuario en función de la página del folleto que se esté visualizando; esto se controla mediante un método principal, *void setActualPage(int i),* que en función del entero que recibe por parámetro, configura la interfaz de usuario mostrando los controles y opciones oportunas de acuerdo a la página del folleto impreso que esté visualizando el usuario. El resto de métodos que están definidos, tanto los que se emplean como complemento a *setActualPage(int i)* como a otros propósitos, se limitan simplemente a llamar a otros métodos del hilo principal de ejecución de la aplicación mediante el método *performSelectorOnMainThread:withObject: waitUntilDone:*
- **uc3m_*(elemento 3D)*.h:** Ficheros de cabecera que contienen la información relativa a la malla de los diferentes Modelos 3D que se emplearán en la aplicación como Realidades Aumentadas, creados siguiendo el procedimiento del apartado 3.3 de la presente Memoria.

En la siguiente figura se muestra un diagrama en el que se puede ver de un modo más gráfico cómo es la jerarquía de clases y ficheros de la aplicación desarrollada. Se representan las relaciones de herencia entre clases así como las interacciones más determinantes entre las mismas que se han tenido que realizar para la consecución del Proyecto.

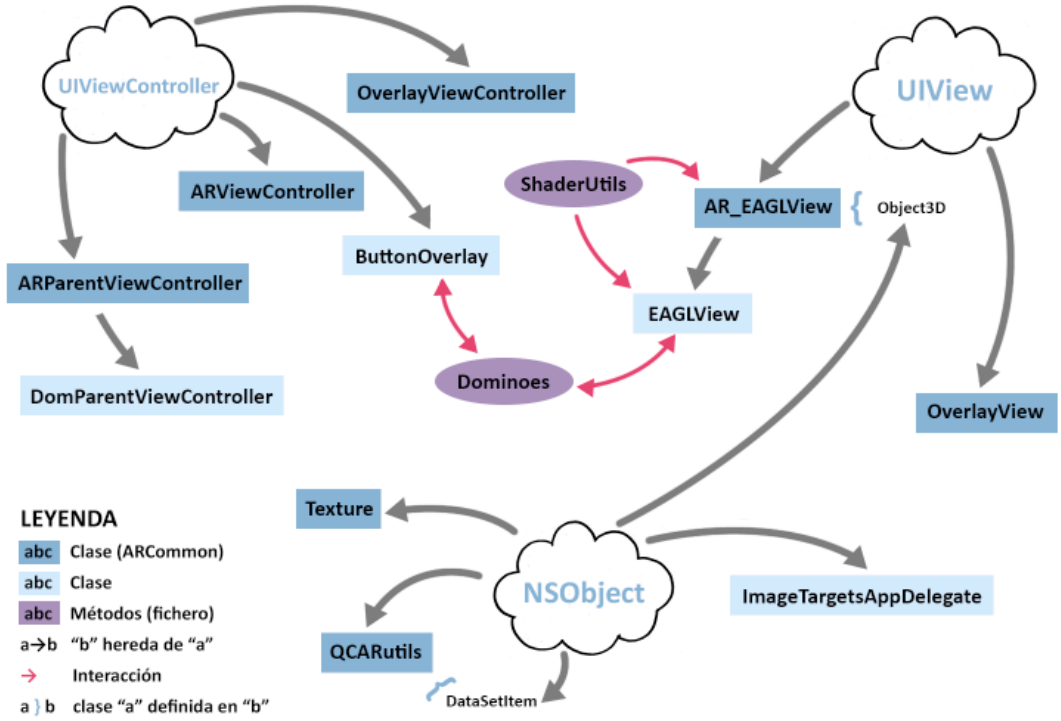


Figura 3.8: Diagrama de Clases

Por último, hay que hacer referencia a los datasets que contienen la información necesaria para que la aplicación pueda identificar y realizar el seguimiento de las diferentes páginas del folleto. Este dataset se crea on-line mediante la herramienta proporcionada por Qualcomm, Target Management System; para ello se han de subir a las diferentes imágenes correspondientes a cada una de las páginas del folleto a dicha herramienta on-line. Una vez son procesadas las imágenes, se procede a la descarga del dataset. En el caso de este Proyecto, los ficheros correspondientes al dataset son: *Folleto.dat* y *Folleto.xml*; es en este último fichero donde se encuentra la información relativa a los diferentes trackables que se emplean en la aplicación, a saber, nombre de la variable que servirá para identificar a un Target dentro del código y su tamaño. Adicionalmente, para establecer Botones Virtuales dentro de un Target, se ha de indicar en este fichero como una etiqueta dentro del Target que alojará dicho Botón Virtual, indicando su nombre (identificador del botón), su tamaño mediante la definición de un rectángulo y si se encuentra habilitado o no. Para una mejor compresión del fichero XML, se muestra a continuación el fichero *Folleto.xml* íntegramente:

```
<?xml version="1.0" encoding="UTF-8"?>
<QCARConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="qcar_config.xsd">
 <Tracking>
  <ImageTarget size="247 172.899994" name="stones"/>
  <ImageTarget size="247 174.75029" name="Pag1">
    <VirtualButton name="botonP1" rectangle="30.0 25.0 70.0 -15.0" enabled="true" />
  </ImageTarget>
  <ImageTarget size="247 174.75029" name="Pag2"/>
  <ImageTarget size="247 174.75029" name="Pag3"/>
  <ImageTarget size="247 174.75029" name="Pag4">
    <VirtualButton name="boton1P4" rectangle="-98.0 -12.0 -78.0 -24.0" enabled="true" />
    <VirtualButton name="boton2P4" rectangle="-35.0 -12.0 -15.0 -24.0" enabled="true"/>
    <VirtualButton name="boton3P4" rectangle="22.0 -12.0 42.0 -24.0" enabled="true"/>
    <VirtualButton name="boton4P4" rectangle="79.0 -12.0 99.0 -24.0" enabled="true"/>
  </ImageTarget>
    <ImageTarget size="247 174.75029" name="Pag5"/>
    <ImageTarget size="247 174.752502" name="Pag6"/>
  </Tracking>
</QCARConfig>
```

Figura 3.9: Fichero *Folleto.xml*

3.5. EAGLView

En este apartado se explicará en profundidad cual es la funcionalidad y cometido de EAGLView dentro de la aplicación, puesto que es aquí donde se ha producido la mayor parte de los esfuerzos de programación, ya que constituye el núcleo de la Realidad Aumentada.

Como ya se indicó, EAGLView se trata de una subclase de AR_EAGLView. Se ha dicho que EAGLView supone el núcleo de la Realidad Aumentada porque es quien se encarga de la gestión y renderizado de todas y cada una de las Realidades Aumentadas de la que consta la aplicación así como de su comportamiento respecto a las acciones del usuario. Para ello y mediante la directiva *import* se importan todos los Modelos 3D, además de sus texturas correspondientes. Para realizar este cometido se define el método:

```
-(void)addMy3DObjectWithTheseVertices:(const float*)vertices
normals:(const float*)normals texcoords:(const float*)texCoords
usingTextureIndex:(NSInteger)textureIndex indices:(const
unsigned short*)indices facesCount:(int)facesCount
scale:(float)scale rotate:(int)rotation
```

El cual se emplea para añadir un Modelo 3D, como objeto de la clase Object3D, en un array. En este array se pretende almacenar todos los Modelos 3D que se emplean en la aplicación (array de objetos de la clase Object3D), por tanto en el método setup3dObjects se realizan tantas llamadas al método que se acaba de declarar, como Modelos 3D se emplean en la aplicación. Nótese que las texturas, en primera instancia, se almacenan en un array independiente al de los Modelos 3D;

ya que puede darse el caso (y de hecho se da en la aplicación) de que un único Modelo 3D pueda disponer de más de una textura. Por eso ambos arrays no tienen necesariamente por qué tener el mismo número de elementos y de ahí que en la llamada al método se emplee un entero, que se corresponde al índice del array de texturas, para designar la textura del Modelo 3D. Así, si por ejemplo se precisa cambiar la textura de un cierto Modelo 3D de la asignada inicialmente en setup3dObjects bastaría con:

```
obj3D.texture = [textures objectAtIndex:n];
```

Donde n sería el índice, es decir, la posición que ocupa la nueva textura que se desea aplicar al Modelo 3D en el array de texturas.

Una vez preparados los arrays de Modelos 3D y texturas; las diferentes Realidades Virtuales están listas para ser renderizadas sobre las diferentes páginas del folleto. A continuación se explicará a fondo el método renderFrameQCAR, ya que se trata del más determinante de toda la aplicación. Se encarga de dibujar el fotograma actual empleando OpenGL. Vuforia llama a este método cuando se desea renderizar el fotograma actual en la pantalla del dispositivo y lo llama en un subproceso (hilo individual) en segundo plano. Es decir, cada fotograma que se le presenta al usuario es resultado de haber llamado a este método.

Cada vez que se llama a este método, lo primero que se hace es vaciar los búferes de color y profundidad de OpenGL:

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Posteriormente, renderizar el vídeo de fondo (es decir, la visión del mundo real, como si sólo se tratara de visualizar el mundo real a través de la cámara)

```
QCAR::Renderer::getInstance().drawVideoBackground();
```

Y recuperar el estado del seguimiento (QCAR::State)

```
QCAR::State state = QCAR::Renderer::getInstance().begin();
```

Gracias a él, se pueden conocer en ese frame qué número de trackables conoce el SDK además de cuántos de ellos se encuentran activos y, por lo tanto, están siendo seguidos por el sistema. También permite el acceso al propio trackable.

El siguiente paso en la ejecución de renderFrameQCAR consiste en determinar qué Targets se encuentran activos; para ello se define el método setTargetsState(QCAR::State state), al cual se le pasa como parámetro el objeto state para determinar qué Targets están activos y en base a ello dar valor a una

serie de variables que funcionan a modo de flag y que indican, en cada Frame, qué Targets se encuentran activos y cuáles no. Después de esto, se establecen una serie de condiciones y se evalúan los flags para representar la Realidad Aumentada adecuada sobre cada uno de los Targets, que tienen que ver con cuales de ellos se encuentran activos, y en función de ello se representará una u otra Virtual o lo hará de un modo u otro.

Cuando el sistema detecta qué pagina del folleto se encuentra visualizando el usuario gracias a los flags que se establecieron, éste tendrá que renderizar la Realidad Virtual y mostrarla sobre el vídeo de fondo para crear la Realidad Aumentada; y además tendrá que adecuar los controles de la interfaz de usuario de acuerdo a la funcionalidad que se preste en esa página. Para esto último, se define el método setActualPage(int i) en Dominoes; que simplemente recibe el número de la página que se está visualizando y en base a él realiza las operaciones oportunas para adaptar los controles y elementos de la interfaz gráfica para que se adecúen a la página que está siendo observada. Desde EAGLView se llama a este método pasándole como parámetro la página que se esté visualizando en un momento dado; si no se visualiza ninguna (o lo que se visualiza no se puede tratar, por ejemplo dos páginas a la vez) el parámetro que se le pasa es i=0.

Para el renderizado de las Realidades Virtuales se parte de la proyección de la vista del Modelo, como un objeto de la clase Matrix44F (QCAR::Matrix44F), que se trata de una matriz de floats de 4 filas y 4 columnas. Modificando ciertos valores de esta matriz, se pueden cambiar determinadas propiedades del Modelo 3D; para ello se definen una serie de métodos en ShaderUtils, como translatePoseMatrix, rotatePoseMatrix o scalePoseMatrix que permiten situar el Modelo 3D sobre las coordenadas deseadas, rotarlo los grados deseados sobre cualquier eje y aumentar o reducir las dimensiones del Modelo 3D, respectivamente. Una vez se encuentra lista esta matriz de posición; se transforman los arrays de vértices, normales, etc. que se importaron de los ficheros de cabecera de los Modelos 3D, actualmente como atributo de la clase Object3D, y se adaptan para que OpenGL ES pueda tratar con ellos; para este propósito se utiliza la función glVertexAttribPointer. Posteriormente a la adaptación a OpenGL ES, los arrays de vértices, normales y coordenadas de textura han de habilitarse, para ello se emplea la función glEnableVertexAttribArray. El último paso, consistente en el renderizado propiamente dicho de la Realidad Virtual, que se realiza con el método glDrawElements.

4. Evaluación y resultados

4.1. Evaluación

Durante el proceso de desarrollo de la aplicación, el procedimiento seguido para evaluar su correcto funcionamiento ha sido el ir ejecutando la aplicación en el dispositivo para comprobar que el código escrito funcionaba correctamente. Por tanto, la evaluación durante el desarrollo inicial ha consistido en ir programando y comprobando lo programado sobre el dispositivo físico.

En una fase más avanzada del desarrollo, se establecieron diferentes pruebas de uso para evaluar la aplicación. Estas pruebas consistieron en dar a probar la aplicación a usuarios no familiarizados con la Realidad Aumentada para que dieran información sobre cómo mejorar la aplicación, cómo disponer los Virtual Buttons sobre el folleto para evitar pulsaciones involuntarias, etc.

4.2. Resultados

Una vez completado el proceso de desarrollo y evaluación; se consigue obtener una aplicación plenamente funcional para el usuario cuyo funcionamiento, desde el punto de vista de éste, se explica a continuación:

La aplicación resultante, *UC3M Virtual*; una vez ejecutada, mostrará una pantalla de bienvenida que mostrará el nombre de la aplicación hasta que se concluya la carga de ésta. Una vez cargada, con la aplicación ya funcional y lista para su uso, se mostrará una vista de lo que capture la cámara en ese momento; y superpuesta a ella, una interfaz de usuario muy simple y poco intrusiva.

Figura 4.1: Pantalla de carga de la aplicación

La aplicación, como se ha venido diciendo a lo largo de esta Memoria, precisa de un folleto impreso para su utilización. En función de la página del folleto que se esté visualizando, la interfaz de usuario se irá modificando automáticamente para adaptarse a la funcionalidad que presente cada una de estas páginas. Por ello, se puede decir que la aplicación consta de N+1 interfaces de usuario, donde N serían las páginas que contiene el folleto (en este caso 6) y el "+1" corresponde a la interfaz mostrada cuando la aplicación no está reconociendo ninguna página del folleto. Toda la interfaz de usuario en conjunto soporta las cuatro orientaciones posibles que puede adoptar el dispositivo físico adaptándose a ellas. En la figura que se muestra a continuación se muestra la interfaz que podrá ver el usuario nada más se complete la carga de la aplicación y que será la misma siempre que no se reconozca ninguna página del folleto, se trata por tanto, de la interfaz inicial.

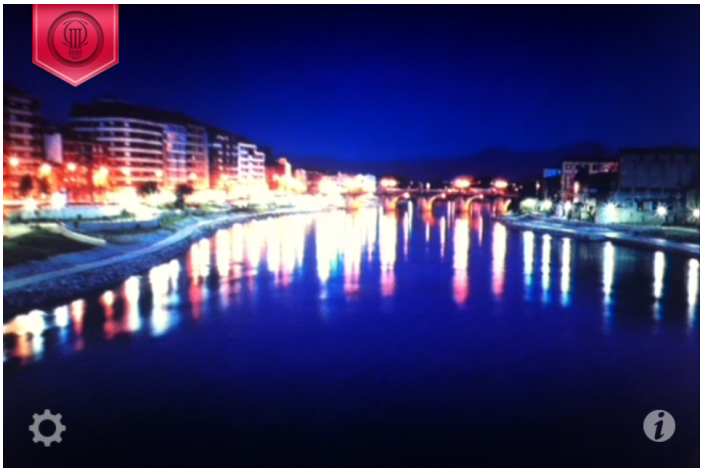


Figura 4.2: Interfaz inicial de la aplicación

En esta pantalla se pueden observar tres elementos de interfaz de usuario; dos de ellos comunes a todas la interfaces que se presenten y uno propio de esta interfaz inicial.

El elemento propio es el botón de información que aparece en la parte inferior derecha de la pantalla; desde él (y sólo desde la pantalla inicial) el usuario accederá a un pequeño cuadro translúcido que se superpondrá sobre la interfaz y que le dará las pautas necesarias para utilizar la aplicación; además desde ahí podrá acceder a un cuadro de información sobre la aplicación.

El primero de los elementos comunes es el marcador rojo que aparece en la esquina superior izquierda. Su funcionalidad es la de informar al usuario sobre qué pagina del folleto se está visualizando. Además de para saber qué página en concreto se está visualizando, este elemento le sirve al usuario para conocer si la

aplicación no está detectando la página, en cuyo caso mostrará el escudo de la Universidad como es el caso de la pantalla inicial.

El segundo de los elementos comunes es el botón de menú de la cámara, que se encuentra en la parte inferior izquierda. Desde él, el usuario podrá encender/apagar la luz del dispositivo físico (flash) para mejorar el reconocimiento de las páginas del folleto, hacer un autofocus puntual y/o mantenido en el tiempo para mejorar el reconocimiento de las páginas del folleto.

A continuación se explicará la funcionalidad que ofrece la aplicación en función de la página del folleto que se esté visualizando.

Página 1: Bienvenida y familiarización

Esta página, configurada como un Image Target con un Virtual Button y uso conjunto con un Frame Marker, servirá para dar la bienvenida al alumno a la Universidad. La página del folleto impreso sirve como portada del folleto, y en ella se dan indicaciones de uso conjunto con la aplicación para que el usuario se familiarice con el uso de la Realidad Aumentada.

Cuando el usuario visualice esta página a través del dispositivo móvil, podrá ver inicialmente el escudo de la Universidad en 3D rotando sobre una parte de la página. El usuario deberá situar uno de los Frame Markers disponibles, uno para usuarios de sexo masculino y otro para sexo femenino, bajo el escudo. En ese momento el escudo desaparecerá y se mostrará un avatar 3D en función del sexo elegido y aparecerá un nuevo botón sobre la interfaz para visualizar un vídeo institucional de bienvenida de la Universidad. Si se retira el Frame Marker de la zona donde aparece el escudo (donde se encuentra un Virtual Button), el avatar y el enlace al vídeo desaparecerán y volverá a mostrarse el escudo de la Universidad.

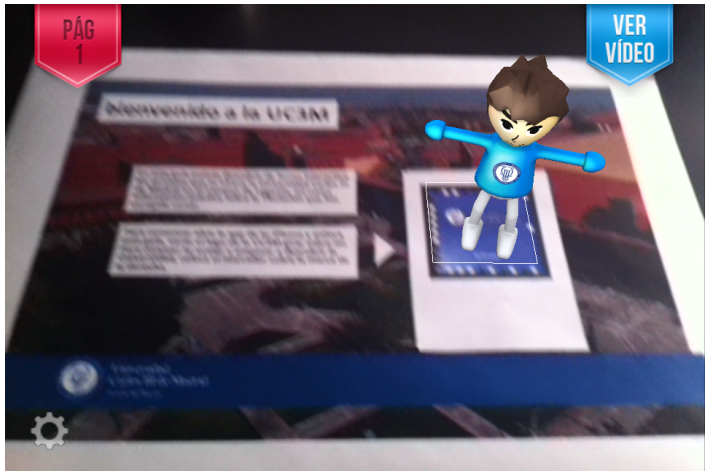


Figura 4.3: Pantalla correspondiente a la página 1 del folleto

Página 2: Edificios del Campus y servicios

Esta página, configurada como un Image Target, servirá para dar a conocer al alumno los edificios del Campus y los servicios que podrá encontrar en cada uno de ellos. En la página del folleto impreso se puede visualizar una vista aérea del Campus de Leganés.

Cuando el usuario visualice esta página a través del dispositivo móvil, podrá ver un globo aerostático que parecerá sobrevolar el Campus; además en la interfaz aparecerá un control de movimiento que le permitirá desplazar el globo aerostático sobre el Campus. Cuando el globo esté situado sobre alguno de los edificios de la Universidad, aparecerá un botón de información que al pulsarlo, se le mostrará al usuario una pantalla donde podrá conocer más información sobre ese edificio, su nombre, su número identificativo en el Campus, una fotografía del edificio y los servicios que presta.

Figura 4.4: Pantalla correspondiente a la página 2 del folleto

Página 3: Aulas

Esta página, configurada como un Image Target, servirá para dar a conocer al alumno la ubicación del aula donde se imparte alguna de sus asignaturas dentro de un edificio y cómo ir hacia ella.

Cuando el usuario visualice esta página a través del dispositivo móvil, podrá ver su avatar en 3D situado sobre la entrada del edificio. El alumno podrá acceder a un menú donde se muestran una serie de asignaturas; cuando se seleccione alguna de ellas, el avatar se desplazará hacia el aula donde se imparte la signatura seleccionada, mostrando al alumno su emplazamiento y cómo acudir hasta él.

Figura 4.5: Pantalla correspondiente a la página 3 del folleto

Páginas 4 y 5: Actividades culturales

Estas páginas, configuradas cada una de ellas como un Image Target que alberga cuatro Frame Markers y cuatro Virtual Buttons, y uso conjunto con un Frame Marker; servirán para dar a conocer al alumno los servicios y actividades de Arte y Cultura (Aula de las Artes) que ofrece la Universidad.

Cuando el usuario visualice alguna de estas dos páginas a través del dispositivo móvil, podrá ver sobre cada uno de los Frame Markers correspondientes a cada una de las actividades ofrecidas un globo con el color correspondiente al logotipo de dicha actividad en el logotipo del Aula de las Artes. Cuando el usuario sitúe el Frame Marker correspondiente a su avatar sobre uno de estos globos, el avatar parecerá explotar el globo y se le mostrará al usuario un cuadro de diálogo para consultar el sitio web de la actividad que haya seleccionado y obtener más información.

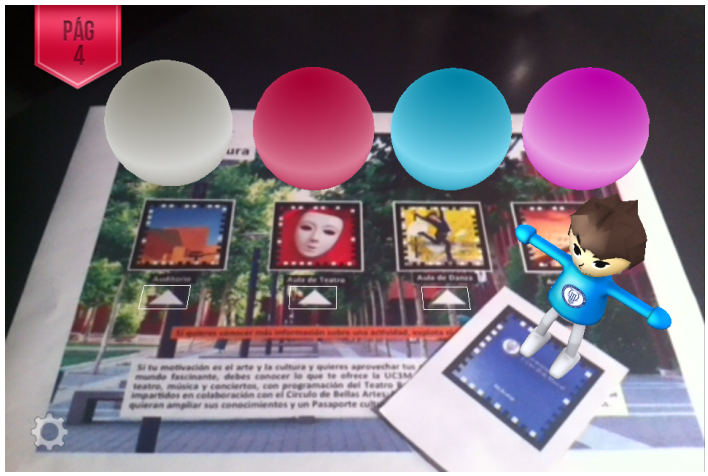


Figura 4.6: Pantalla correspondiente a la página 4 del folleto

Página 6: Actividades deportivas

Esta página, configurada como un Image Target y uso conjunto con un Frame Marker, servirán para dar a conocer al alumno los servicios y actividades de Área de Deporte y Actividad Física de la Universidad

Cuando el usuario visualice esta página a través del dispositivo móvil, el usuario podrá ver superpuesta sobre la página del folleto una Realidad Virtual plana, que será susceptible de mostrar adecuadamente texto e imágenes de forma que se pueda extender la información que presenta impresa el folleto sin necesidad de emplear más papel adicional y todo de forma interactiva. A través de un menú, el alumno podrá navegar a través de las diferentes actividades y cursos deportivos que ofrece la Universidad, y cuando seleccione alguno de ellos podrá obtener más información acerca de dicho curso sobre la Realidad Virtual plana, descripción, horarios, precios, etc. Adicionalmente, si el alumno coloca el Frame Marker de su avatar sobre el folleto, a los pocos segundos se le instará a ver un vídeo promocional de la Universidad en el que se muestra su oferta deportiva.

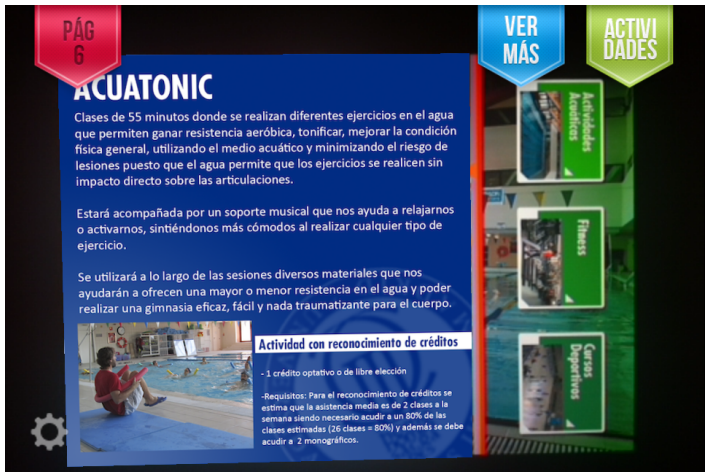


Figura 4.7: Pantalla correspondiente a la página 6 del folleto

5. Futuras líneas para ampliar el Proyecto

Las posibilidades que ofrece la Realidad Aumentada son muy numerosas, como se ha visto en el capítulo Análisis del Estado del Arte; es por ello que la aplicación que se ha realizado en este Proyecto tiene un gran potencial para dotarla de más funcionalidad y de características muy atractivas.

Las posibilidades de ampliación del Proyecto que se van a exponer a continuación se pueden aplicar no sólo sobre una aplicación dedicada como la creada para este Proyecto; sino que se podrían incorporar como funcionalidad extra a las diferentes aplicaciones que la Universidad Carlos III de Madrid tiene publicadas actualmente. Las mejoras y nuevas posibilidades que se podrían implementar se van a describir en diferentes bloques.

5.1. Horarios/Agenda

Partiendo del concepto mostrado en la página del folleto en la que se muestra el plano de planta de un edificio de la Universidad y el avatar se desplaza hasta el aula donde se imparte la asignatura que el alumno selecciona; se podría implementar una agenda de modo que a través de internet, el alumno se pudiera identificar en Campus Global para que la aplicación volcara sobre una agenda el horario de clases del alumno. Si se configuraran los planos ya existentes colgados en las paredes de todos los edificios de la Universidad como Image Targets; el alumno podría consultar sobre ellos; sin necesidad del folleto dónde se impartiría su próxima clase. El menú de selección de asignaturas, se mantendría actualizado en función de las materias matriculadas y del horario que se descargaría una vez identificado el alumno.

Aprovechando el concepto de Realidad Virtual plana, que se muestra en la aplicación en la página correspondiente a Deportes y Actividad Física, ideal para mostrar texto; se podría aprovechar para que el alumno consultara los horarios de una clase in-situ. Es habitual que el alumno acuda a las aulas informáticas a realizar trabajos y tareas fuera del horario de clase; y también es habitual que en muchas ocasiones éste tenga que abandonar el aula porque en esa misma clase se va a impartir alguna asignatura. Mediante la instalación de algún tipo de placa situada a la entrada del aula, por ejemplo bajo la placa de identificación del aula, se podría configurar como un Image Target sobre el que el alumno podría consultar el horario de dicho aula, que además estaría actualizado en todo momento, pues la textura del plano virtual que contendría el horario, se obtendría a través de internet.

5.2. Material docente

Es muy común que el profesorado emplee diapositivas y presentaciones en la impartición de sus clases y los pongan a disposición de los alumnos. Se podrían configurar algunas de estas diapositivas de modo que el alumno cuando las viera a través de su dispositivo móvil, se le mostraran contenidos extras, que podrían ser interactivos. Se podrían mostrar, por ejemplo, Modelos 3D de artefactos, maquinaria y cualquier elemento cuya visualización como un elemento 3D ofrece muchísima más información que una simple imagen sobre papel. Además estos Modelos se podrían animar para proporcionar todavía más información; incluso del mismo modo que existen *applets* para comprobar y experimentar fenómenos físicos, etc. se podría hacer esto mismo pero haciendo uso de Realidad Aumentada.

También se podrían presentar vídeos y contenido multimedia enlazados a las diferentes diapositivas y, por supuesto, se podría completar la información en soporte escrito mediante el empleo de Realidades Virtuales planas que presentasen texto.

5.3. Información en el Campus (cartelería)

Tradicionalmente, un medio que se viene utilizando como canal para informar al alumno de los eventos culturales, deportivos, actividades docentes, conferencias, etc. son los carteles que se encuentran distribuidos por el Campus. Este soporte puede proporcionar información de forma muy limitada; si misión principal es llamar la atención del alumno para que éste se detenga a leer el contenido del cartel, el cual es habitual que le remita a una página web o algún emplazamiento físico para que obtenga más información.

Haciendo uso de la Realidad Aumentada, se podría configurar la cartelería como diferentes Image Targets, de forma que el usuario obtuviera más información extra sin necesidad de que tuviera que acceder explícitamente a un navegador de internet a buscar información.

Un ejemplo de aplicación directa sobre una infraestructura ya existente en la Universidad sería la cartelería que se encuentra instalada sobre la entrada del Auditorio Padre Soler; en ella suele encontrarse información relativa a la programación de eventos y actos culturales que ahí tienen lugar, pero de forma muy escueta, por lo general, fecha y nombre del evento. Mediante el empleo de Realidad Aumentada se podría ofrecer más información sobre cada uno de los eventos programados en texto sobre una Realidad Virtual plana, mostrar vídeos

relacionados con el evento; incluso se podrían reservar entradas para los mismos a través de la propia aplicación.

Otro ejemplo de aplicación directa sería aprovechar los espacios reservados a cartelería de grandes dimensiones reservados sobre la fachada de una de las esquinas de la Biblioteca. En este lugar es frecuente que se cuelguen carteles ofreciendo información sobre jornadas deportivas, seminarios y cursos que ofrece la Universidad, y del mismo modo que en el ejemplo anterior, a través de la Realidad Aumentada se le podría ofrecer al alumno mayor información además de la que se ofrece en el propio cartel; también se podría gestionar y realizar inscripciones a dichos eventos anunciados a través de la propia aplicación.

Por último, cabe destacar que la misma funcionalidad que ofrecen los monitores que están repartidos por el campus y que ofrecen un servicio de cartelería digital de información sobre la Universidad, se podría conseguir con esta aplicación. Simplemente bastaría con instalar paneles con algún tipo de imagen o texto que explicara el funcionamiento de dicho panel con la aplicación, de esta forma al visualizar el panel con el dispositivo móvil se podría conseguir el mismo resultado que si sobre el panel se encontrara el propio plasma. Esto tiene la gran ventaja, además del nulo consumo eléctrico por parte de la Universidad, que el alumno puede consultar la información en cualquier hora y momento.

5.4. Aplicación Android

Puesto que el SDK Vuforia también está disponible para Android, se podría implementar la aplicación para esta plataforma. De este modo, junto con iOS, estaría cubierta la mayor parte del mercado de Smartphones.

5.5. Ampliación con Unity

Mediante el empleo del motor gráfico Unity se podría dotar a las Realidades Virtuales de animaciones más complejas que mejorarían notablemente la experiencia de usuario, así como incrementar las posibilidades de la aplicación.

5.6. GPS y brújula

Se podría combinar el uso del GPS y brújula disponibles en el dispositivo, para dotar de nuevas funcionalidades a la aplicación: situación de aulas y servicios, itinerarios hacia éstos desde la posición actual del usuario, etc.

6. Presupuesto y planificación

6.1. Presupuesto

A continuación se justifican los costes globales de la realización del presente Trabajo Fin de Grado.

La realización del Proyecto comenzó en el mes de Diciembre de 2011, concluyéndose en Agosto de 2012; por lo que su desarrollo se ha prolongado durante 9 meses. Este proceso de desarrollo se ha compatibilizado durante los 6 primeros meses con los estudios, prácticas y tareas del curso académico; y los últimos 2 meses de desarrollo con un trabajo a media jornada.

Como se expone en el punto 1.4 de la presente memoria, el desarrollo del Proyecto está dividido en diversas fases. A continuación se muestra la distribución de horas empleadas en cada una de dichas fases:

- **FASE 1**: Introducción a la programación en Objective-C: Se llevó a cabo durante los meses de Diciembre y Enero empleando unas 90 horas.
- **FASE 2**: Despliegue del entorno de desarrollo: Se realizó en el mes de Febrero empleándose unas 22 horas.
- **FASE 3**: Introducción a la programación para iOS: Esa fase tuvo lugar durante los meses de Febrero, Marzo y Abril, empleando unas 163 horas.
- **FASE 4**: Introducción al SDK "Vuforia" de Qualcomm: Se realizó entre los meses de Abril y Mayo empleando unas 43 horas.
- **FASE 5**: Introducción al modelado en 3D: Tuvo lugar en el mes de Mayo empleando unas 38 horas.
- **FASE 6**: Desarrollo de la aplicación final: Es la fase central del Proyecto y tuvo lugar desde Mayo hasta la finalización del Proyecto en Agosto, empleándose unas 423 horas en su desarrollo.
- **FASE 7**: Redacción de la Memoria: Se realizó paralelamente al desarrollo de la aplicación empleando unas 158 horas.

Los costes imputables aplicados a gastos de material y personal se pueden deducir de las Tablas 6.1 y 6.2.

Fase 1	Introducción a la programación en Objective-C	**90 horas**
Fase 2	Despliegue del entorno de desarrollo	**22 horas**
Fase 3	Introducción a la programación para iOS	**163 horas**
Fase 4	Introducción al SDK "Vuforia" de Qualcomm	**43 horas**

Fase 5	Introducción al modelado en 3D	**38 horas**
Fase 6	Desarrollo de la aplicación final	**423 horas**
Fase 7	Redacción de la Memoria	**158 horas**
TOTAL		**937 horas**

Tabla 6.1: Fases del Proyecto y horas invertidas

En la Tabla 6.1 se muestran las fases del Proyecto y el tiempo aproximado que se ha invertido en cada una de ellas. De ella se concluye que para la consecución del Proyecto han sido necesarias unas 937 horas. Estableciendo unas tarifas de 20 €/hora, el coste de personal asciende a 18.740 €.

Concepto	Precio	Amortización	Importe
Ordenador portátil Apple MacBook Pro	2.279,00 €	1/4	569,75 €
Teléfono móvil Apple iPhone 4, 32GB	645,00 €	1/4	161,25 €
Impresora Photosmart 5510 e-All-in-One de HP con AirPrint	49,95 €	1/4	12,48 €
Xcode 4.3.2	0,00 €	-	0,00 €
Cuenta Apple Developer Program	79,00 €	1/4	19,75 €
Vuforia SDK 1.5.8 de Qualcomm	0,00 €	-	0,00 €
Cheetah3D 6.0	80,00 €	1/4	20,00 €
Blender 2.63a	0,00 €	-	0,00 €
OpenGL Export Addon	0,00 €	-	0,00 €
Adobe Photoshop CS6	942,82 €	1/4	235,71 €
Documentación	-	-	103,47 €
TOTAL			**1.122,41 €**

Tabla 6.2: Costes de material

En la Tabla 6.2 se desglosan los costes de material. Se ha tenido en cuenta un coeficiente de amortización de 1/4 según la vida media de cada elemento que así lo requiera. Así la partida de costes de material asciende a 1.122,41 €. De esto, se obtiene el presupuesto total del Proyecto que se presenta en la Tabla 6.3.

Concepto	Importe
Costes de personal	18.740 €
Costes de material	1.122,41 €
Costes indirectos (20%)	3.972,48 €
Base imponible	23.834,89 €
I.V.A. (21%)	5.005,33 €
TOTAL	**28.840,22 €**

Tabla 6.3: Presupuesto

El coste total del Proyecto asciende a **veintiocho mil ochocientos cuarenta euros con veintidós céntimos**.

6.2. Planificación

En la Tabla 6.4 se muestra la Estructura de Descomposición de Trabajo (EDT) en la que se ve con más detalle cuál es la descomposición en tareas de cada una de las fases generales de la elaboración del Proyecto que se citan en el presupuesto, así como las horas planificadas a emplear en cada una de ellas. Se omite la forma de representación gráfica de la EDT debido a la gran extensión que requiere.

TAREA	DESCRIPCIÓN	HORAS
1	**Introducción a la programación en Objective-C (100h)**	**100**
1.1	Búsqueda de recursos y documentación sobre Objective-C	15
1.2	Lectura de libros y documentación para aprender a programar en Objective-C	85
2	**Despliegue del entorno de desarrollo (20h)**	**20**
2.1	Xcode	4
2.1.1	Descarga	1,5
2.1.2	Instalación y configuración	2,5
2.2	Apple Developer Program	6
2.2.1	Registro	0,5
2.2.2	Descarga de certificados, etc.	2
2.2.3	Configuración terminal (iPhone) sobre el que se realizarán las pruebas de desarrollo	3,5
2.3	SDK Realidad Aumentada "Vuforia"	4
2.3.1	Registro en Qualcomm Developer	0,5
2.3.2	Descarga	1
2.3.3	Instalación y configuración	2,5
2.4	Pruebas para comprobar el correcto funcionamiento del entorno de desarrollo completo	6
3	**Introducción a la programación para iOS**	**150**
3.1	Búsqueda de recursos y documentación sobre iOS	6
3.2	Lectura de libros y documentación para conocer la arquitectura de iOS y aprender a programar aplicaciones para la plataforma	110
3.3	Programación de aplicaciones sencillas para iOS	30
3.4	Pruebas de funcionamiento de aplicaciones de ejemplo desarrolladas sobre simulador y dispositivo físico	4
4	**Introducción al SDK de Realidad Aumentada "Vuforia" de Qualcomm**	**50**
4.1	Documentación sobre el SDK	10
4.1.1	Estudio de la arquitectura de Vuforia	5
4.1.2	Búsqueda de recursos y documentación sobre C++	1
4.1.3	Lectura y estudio de documentación obtenida	4
4.2	Pruebas para conocer las funcionalidades ofrecidas por Vuforia	3
4.2.1	Descarga de aplicaciones sencillas de ejemplo proporcionadas por Qualcomm	0,5

4.2.2	Prueba en dispositivo físico de aplicaciones de ejemplo proporcionadas por Qualcomm	2,5
4.3	Ingeniería Inversa	37
4.3.1	Lectura y compresión del código fuente de las aplicaciones de ejemplo proporcionadas por Qualcomm	5
4.3.2	Modificación del código fuente de las aplicaciones de ejemplo proporcionadas por Qualcomm para comprender su funcionamiento y conseguir nuevas funcionalidades	32
5	**Introducción al modelado en 3D**	**30**
5.1	Búsqueda de recursos y documentación sobre OpenGL	3
5.2	Despliegue del entorno de creación de Modelos 3D	1
5.2.1	Cheetah 3D	0,3
5.2.2	Blender	0,3
5.2.3	Adobe Photoshop CS6	0,4
5.3	Búsqueda de recursos y documentación para aprender a manejar el entorno de creación de Modelos 3D	7
5.4	Elaboración de un procedimiento básico de importación/adaptación de Modelos 3D a aplicaciones Vuforia	19
6	**Desarrollo de la aplicación final**	**400**
6.1	Elaboración del folleto informativo que se usará junto a la aplicación final	10
6.1.1	Diseño y maquetación del folleto	9,5
6.1.2	Creación del dataset correspondiente al folleto mediante el Target Management System para incorporarlo a la aplicación final	0,5
6.2	Programación de la aplicación final	350
6.3	Desarrollo de la interfaz de usuario	25
6.3.1	Diseño de los elementos de interfaz de usuario en Adobe Photoshop CS6	8
6.3.2	Incorporación a la aplicación final (programación adicional necesaria)	17
6.4	Pruebas y evaluación de la aplicación final sobre dispositivo físico	15
7	**Elaboración y redacción de la Memoria**	**150**
7.1	Estructuración de contenidos	4
7.2	Búsqueda de recursos y documentación	25
7.3	Redacción de la Memoria	121

Tabla 6.4: EDT

La planificación del Proyecto se ha realizado en horas en previsión de que el tiempo al día dedicado a su realización no sería regular a lo largo del proceso de desarrollo debido a las circunstancias personales.

Una vez vista la descomposición del Proyecto en tareas, en el ANEXO I se muestra el diagrama PERT donde se ven las dependencias y la secuenciación de tareas, así como las tareas críticas que establecen el camino crítico que determina la duración mínima para realizar el Proyecto. Esta duración mínima queda establecida en 847 horas.

A la vista de estos datos, la realización del Proyecto ha tenido un desfase respecto a la planificación inicial que ha hecho que su finalización se haya demorado en unas 90 horas.

7. Conclusiones

7.1. Consecución de objetivos

Cuando se inició el desarrollo de este Proyecto se estableció un objetivo principal y una serie de objetivos parciales cuya consecución determinaría el resultado del Proyecto.

- Se ha aprendido a programar en Objective-C, conociendo su sintaxis y características; lo que ha permitido desarrollar y establecer la lógica de la aplicación desarrollada.
- Se ha estudiado y comprendido cómo es la arquitectura y funcionamiento de la plataforma iOS de Apple, que ha permitido que la aplicación desarrollada sea apta para ejecutarse sobre un iPhone.
- Se ha logrado manejar con soltura el entorno de desarrollo Xcode, lo que ha permitido realizar todo el trabajo de programación y realizar las pruebas de la aplicación a desarrollar sobre el dispositivo físico, detección de errores en código, establecer la interfaz de usuario, etc.
- Se han estudiado los entornos de desarrollo de Realidad Aumentada disponibles en iOS, consiguiendo así tener una visión global de las características y funcionalidades que proporcionan.
- De estos entornos, se ha estudiado y comprendido el funcionamiento de uno de ellos en particular, Vuforia. Se ha aprendido cómo es su funcionamiento y arquitectura.
- Se ha conseguido integrar satisfactoriamente las funcionalidades de Realidad Aumentada proporcionadas por Vuforia en la aplicación final.
- Se ha aprendido a modelar Objetos 3D, además de adquirir conocimientos sobre OpenGL ES, necesarios para crear y representar las Realidades Virtuales que aparecen en la aplicación. Esto ha permitido elaborar un procedimiento básico de importación/adaptación de Objetos 3D inexistente hasta ese momento.

Gracias a la consecución de todos y cada uno de los objetivos parciales establecidos, se ha conseguido alcanzar el objetivo principal de este Proyecto.

Se ha logrado implementar con éxito una aplicación atractiva y novedosa para dispositivos iOS, con una interfaz de usuario sencilla pero agradable; que mediante el empleo de Realidad Aumentada y junto con un folleto informativo impreso, sirve de ayuda a los estudiantes.

7.2. Reflexiones personales

Para la realización de este Proyecto ha sido necesario el estudio de varias tecnologías y plataformas de las cuales no se tenía absolutamente ninguna noción, lo cual ha supuesto un gran esfuerzo personal para adquirir los conocimientos necesarios para dominar cada una de ellas individualmente y para hacerlas funcionar en común.

En las primeras etapas del Proyecto, correspondientes a la documentación sobre las tecnologías y herramientas implicadas en la consecución del mismo; fue bastante complicado realizar estimaciones de cuánto tiempo sería necesario invertir para realizar el Proyecto ya que no se conocía la dificultad real de las herramientas involucradas. Por ello, y aunque se iban estableciendo plazos conservadores para ir avanzando en el desarrollo del mismo, se tuvieron que ir aplazándose en varias ocasiones debido a las dificultades surgidas durante el aprendizaje, lo que repercutía en la demora total del Proyecto.

Además, la falta de documentación existente sobre el SDK Vuforia de Qualcomm ha hecho necesario el establecer contacto con otros desarrolladores de la plataforma a través del foro proporcionado por Qualcomm, lo que ha supuesto una experiencia muy enriquecedora y útil. Sumado al hecho de tener que leer la documentación en inglés y la necesidad de expresarse en este idioma para plantear y resolver dudas a la comunidad de desarrolladores en el foro, ha supuesto una oportunidad para practicar con esa lengua. A pesar de ello, esta falta de documentación ha sido la parte más negativa del SDK; lo cual establece una gran barrera de entrada y dificulta mucho la programación.

Como conclusión final, decir que a sabiendas de la gran dificultad que entrañaba la realización de este Proyecto, desde el inicio se planteó como un reto personal motivado por las ganas de aprender a programar para la plataforma iOS. Aunque en ciertos momentos del desarrollo del Proyecto, en los que las dificultades que se iban presentando impedían avanzar en la implementación de la aplicación se pensó en abandonar; finalmente se logró completar con éxito alcanzando todos los objetivos marcados al inicio del mismo. El hecho de tener en el teléfono móvil una aplicación creada por uno mismo y verla ejecutarse después de todo el tiempo y esfuerzo empleado; produce una grata satisfacción y hace tener la sensación de que haya merecido la pena tanta dedicación y esfuerzo.

Bibliografía

(Bennett, Fisher, & Brad, 2010) Bennett, G., Fisher, M., & Brad, L. *Objective-C for Absolute Begginers.* Apress. 2010

(Ray, 2012) Ray, J. *Teach Yourself iOS 5 Application Development*. Sams. 2012

(Rideout, 2010) Rideout, P. *Developing Graphical Applications with OpenGL ES, iPhone 3D Programming*. O'Reilly. 2010

(Milgram & Kishino, 1994) Milgram, P., & Kishino, A. *Taxonomy of Mixed Reality Visual Displays.* IEICE Transactions on Information and Systems. 1994

(Azuma, 1997) Azuma, R. *A Survey of Augmented Reality. Presence Teleoperators and Virtual Environments*. 1997

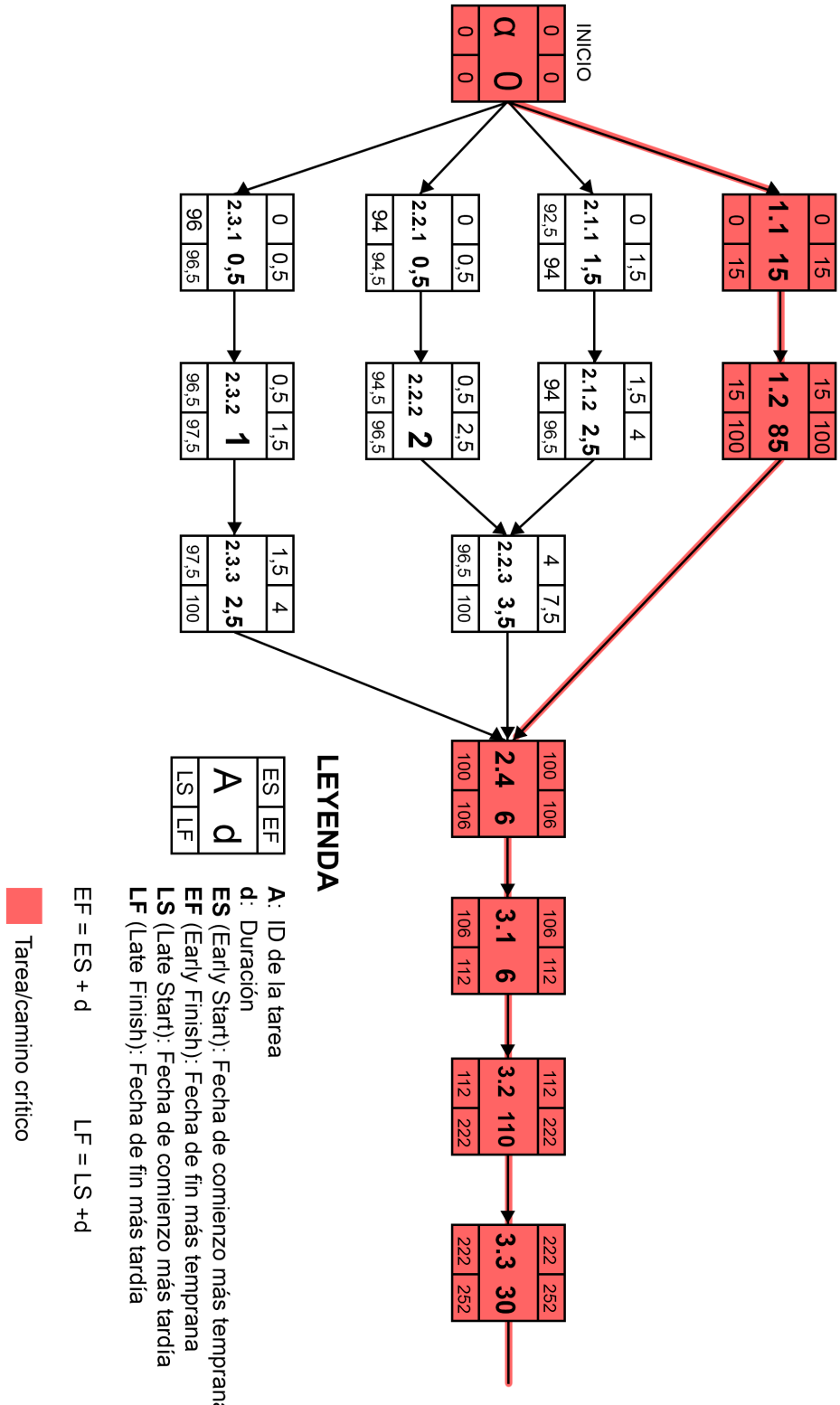
(All Things D., 2012) All Things D. (04 de 04 de 2012). Recuperado el 08 de 08 de 2012, de AllThingsD.: *http://allthingsd.com/20120404/google-unveils-project-glass-wearable-augmented-reality-glasses/*

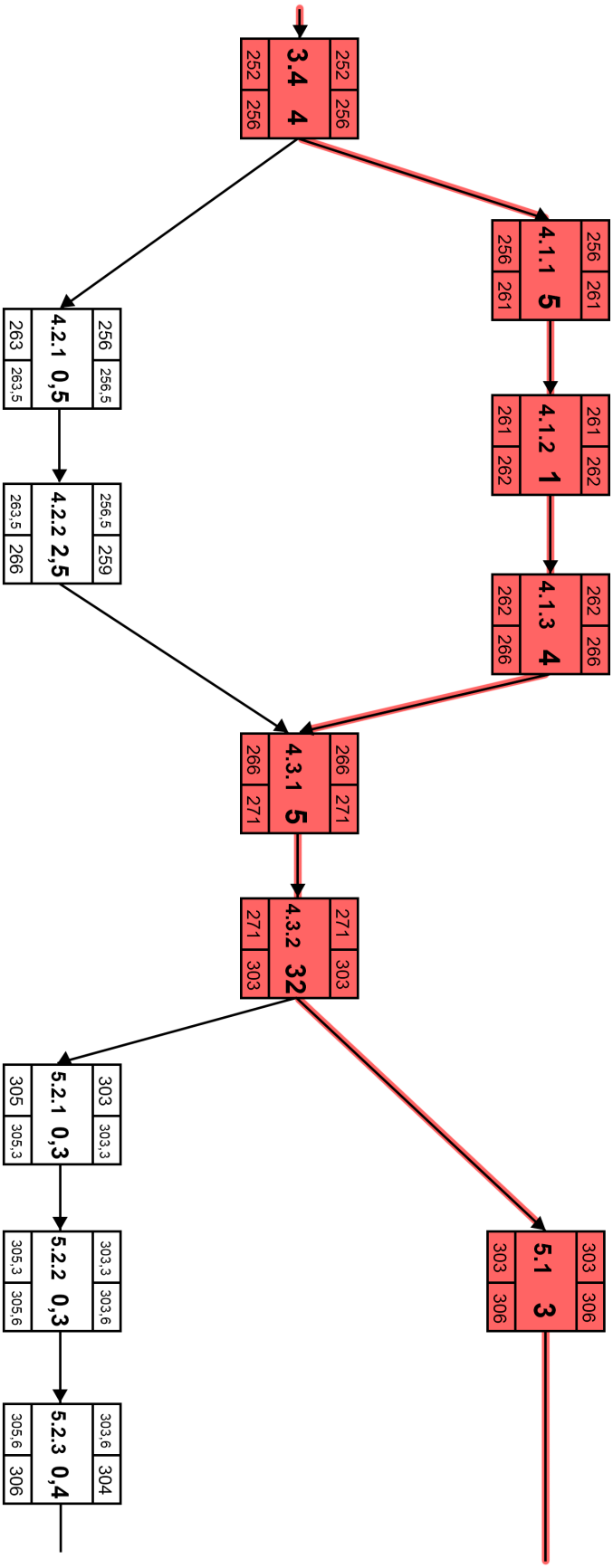
(Feiner, 2011) Feiner, S. *Augmented reality: a long way off?* Recuperado el 09 de 08 de 2012, Pocket-lint. 03 de 03 de 2011

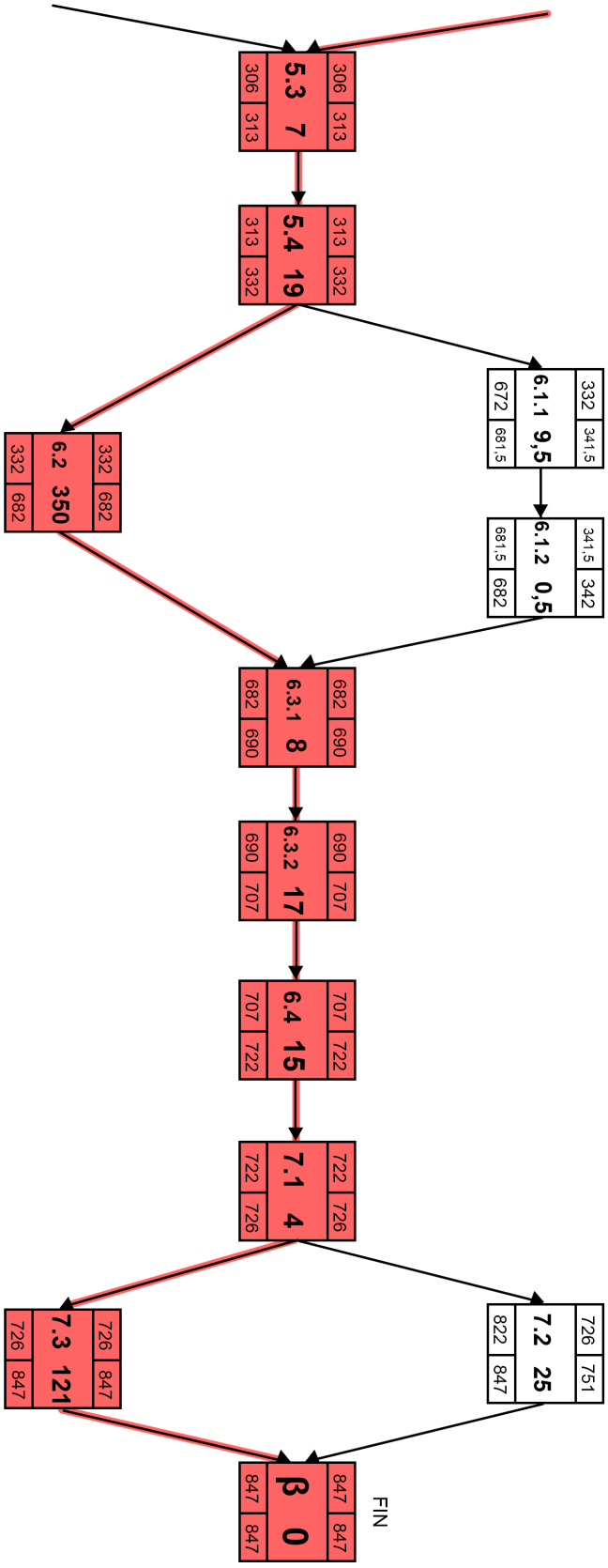
(Raskar, Greg, & Henry, 1998) Raskar, R., Greg, W., & Henry, F. *Spatially Augmented Reality. First International Workshop on Augmented Reality.* 1998

(PC Magazine, 2011) PC Magazine. *Smartphone definition from PC Magazine Encyclopedia. PC Magazine*. 15 de 12 de 2011

Anexo I: Diagrama PERT




3.4 4
4.1.1 5
4.2.1 0,5
4.1.2 1
4.2.2 2,5
4.1.3 4
4.3.1 5
4.3.2 32
5.2.1 0,3
5.1 3
5.2.2 0,3
5.2.3 0,4


5.3 7
306 313
306 313
5.4 19
313 332
313 332
6.1.1 9,5
672 681,5
332 341,5
6.2 350
332 682
332 682
6.1.2 0,5
681,5 682
341,5 342
6.3.1 8
682 690
682 690
6.3.2 17
690 707
690 707
6.4 15
707 722
707 722
7.1 4
722 726
722 726
7.3 121
726 847
726 847
7.2 25
822 847
726 751
β 0
847 847
847 847
FIN