Trabajo de Fin de Grado

Módulo de realidad aumentada geolocalizada

Geolocated augmented reality module

Cristo González Rodríguez

La Laguna, 9 de Junio de 2015

D[a]. **Carina Soledad González González**, con N.I.F. 54.064.251-Z profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

D. **Luis Gonzalo Aller Arias**, con N.I.F. 10.081.577-X Director de Proyectos de la empresa Improve Change S.L., como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Módulo de realidad aumentada geolocalizada."

ha sido realizada bajo su dirección por D. **Cristo González Rodríguez**, con N.I.F. 45.899.378-A.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 9 de Junio de 2015.

Agradecimientos

A Carina, por apoyarme.

A Gonzalo, por acogerme.

A Jorge, por valorarme.

A mis profesores, por guiarme.

A mi familia, por entenderme.

Y a mis amigos, por encontrarme.

Simplemente, gracias.

Licencia

Resumen

La finalidad del presente trabajo ha sido la de crear un módulo de realidad aumentada geolocalizada, que pretende integrarse en el proyecto Progrezz, un juego serio que busca la mejora del ámbito social. Esta categoría de juegos recoge aquellas actividades gamificadas cuyo objetivo no es la mera diversión del jugador, y que en este caso se apoya en el auge y las posibilidades tecnológicas que ofrecen los dispositivos móviles.

Se ha decidido la creación del visor de realidad aumentada como una aplicación web, lo que ha requerido previamente, un análisis exhaustivo acerca del empleo de distintas tecnologías y sensores: utilización de la cámara de vídeo, acceso al giroscopio y el GPS, o el dibujado de gráficos 3D. Siempre teniendo en cuenta la máxima de que el módulo fuera accesible al mayor número de personas posible, pudiendo ejecutarse en cualquier dispositivo con acceso a la geolocalización.

El visor aporta la funcionalidad de visualizar e interactuar con objetos ubicados de acuerdo a unas coordenadas geográficas, pudiendo observarse de acuerdo a la posición real y la orientación que posea el dispositivo donde se está ejecutando el visor. Se cuenta con indicadores que muestran al jugador aquellas tecnologías que están disponibles (vídeo, giroscopio y geolocalización), ofreciendo alternativas cuando se puede, para adaptarse a los limitaciones de la tecnología web aún experimental. Así mismo, se planteó un modo estereoscópico, pensado para ser utilizado en gafas de realidad virtual basadas en dispositivos móviles.

Ha constituido una oportunidad de indagar en los entresijos de la tecnología web y la realidad aumentada, poniendo a prueba sus posibilidades y limitaciones. También ha permitido apoyar a un proyecto de la envergadura de Progrezz, tanto por su nobleza como por su repercusión potencial; abriendo la posibilidad de que cumpla la labor para la que fue creado y que cualquier persona pueda aportar en el futuro.

Palabras clave: *Realidad aumentada, geolocalización, juego serio, Progrezz, aplicación web.*

Abstract

The purpose of this work has been to create a geolocated augmented reality module, whose purpose is to be integrated into the Progrezz project, a serious game that intend improve the social environment. This game category includes those activities whose purpose isn't only the fun of the player, and in this case they lean on the rise and the technological possibilities offered by mobile devices.

It has been decided to create the augmented reality viewer as a web application, which previously required an analysis about the use of different technologies and sensors: the video camera, access to the gyroscope and GPS, or drawn of 3D graphics. Keeping in mind that the module can allow access to as many people as possible, and it can run on any device with geolocation access.

The viewfinder provides the functionality of viewing and interacting with placed objects according to geographical coordinates. These objects can be observed according to the player's actual position and the device orientation that is running the application. It has indicators that show the player the technologies that are available (video, gyroscope and geolocation), offering alternatives when possible, to adapt the module to the limitations of the web experimental technology. Also, it has been created a stereoscopic mode, intended to be used in virtual reality glasses based on mobile devices.

It has been provided an opportunity to inquire into the intricacies of web technology and augmented reality, testing its possibilities and limitations. It also has been allowed to support a big project like Progrezz, for its nobility and its potential impact; opening the possibility to fulfill its function, and that anyone can help in the future.

Keywords: *Augmented reality, geolocation, serious game, applied game, Progrezz, web application.*

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

1.1 Descripción

En las próximas páginas se detallarán todas las cuestiones relativas al Trabajo de Fin de Grado que ha consistido en la creación de un módulo de realidad aumentada[1] geolocalizada[2]. En primer lugar, se tratarán una serie de temas y cuestiones con el objetivo de entender el contexto que rodea al presente proyecto y la finalidad del mismo antes de entrar en materia. Se continuará con la descripción de las actividades realizadas en el mismo, junto a las decisiones tomadas a lo largo de todo el desarrollo. Finalizando, como no puede ser de otra manera, con las conclusiones y las referencias bibliográficas utilizadas como apoyo.

1.2 Juegos serios

El tema sobre el que gira el presente Trabajo de Fin de grado no es otro que el de los juegos serios. Consisten en una modalidad de juegos cuya diferencia respecto a las actividades gamificadas clásicas, reside en la finalidad para la que son desarrollados. Dicho de otro modo, el principal objetivo no es la mera diversión del usuario o jugador; pudiendo ser contemplada su utilización en multitud de ámbitos diversos, desde el aprendizaje basado en juegos, hasta la publicitación de una determinada marca o producto.

---

[1] Visión a través de un dispositivo tecnológico, de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales, creando una realidad mixta.

[2] La geolocalización es la técnica de posicionamiento espacial de una entidad en una localización geográfica única, mediante un sistema de coordenadas.

1.3 Progrezz

Es importante entender previamente qué es y en que consiste Progrezz[3], dado que el propósito del proyecto sobre el que trata esta memoria ha sido ampliar y mejorar el videojuego serio que responde a dicho nombre. En líneas generales, se enfoca en el propósito de mejora del ámbito social a todos los niveles.

Progrezz nace como un proyecto de software libre[4] y código abierto[5] (licencia MIT[6]) que busca la creación de un videojuego ideado para smartphones[7] y diseñado como aplicación web[8], y a fecha de la entrega del presente informe, se encuentra en estado de desarrollo constante.

Entrando en materia, se pretende entrelazar la realidad con la propia historia y mecánica del juego, de tal manera que una energía llamada "entropía" simboliza el desequilibrio social, que ha sido generado por todas aquellas acciones negativas que se manifiestan en la sociedad actual. Partiendo de esta realidad, se invita al jugador a encarnar a un miembro de una red clandestina de voluntarios, que buscan detener el colapso de la civilización, y devolver la armonía al mundo.

Haciendo uso de la geolocalización, se proporciona al usuario un escenario que explorar. Donde la posibilidad de encontrar mensajes y otros elementos ubicados en el mapa real, constituye una de las mecánicas básicas, y cuya recolección está ligada al desplazamiento físico del jugador a su lugar concreto. La utilidad de los objetos recolectados: permitir progresar en la

---

[3] Presentación de Progrezz en la 4ª Conferencia Europea de Juegos Serios - http://www.slideshare.net/GoFlipover/progrezz-social-crowdsourcing-crowdfunding-with-serious-games

[4] Software que permite su uso, copia, estudio, modificación y redistribución de forma libre tras su adquisición.

[5] Software 'Open Source', que permite su distribución y desarrollo de forma libre y gratuita, dando acceso al código fuente.

[6] Información sobre licencia MIT: http://es.wikipedia.org/wiki/Licencia_MIT

[7] Tipo de teléfono móvil construido sobre una plataforma informática, con gran capacidad de almacenamiento de datos y realización de actividades.

[8] Herramientas de acceso a un servidor web mediante un navegador.

historia (descubriendo nuevas mecánicas o aumentando el nivel y privilegios del jugador), promocionar eventos, contener textos de jugadores avanzados, etc. Por supuesto es necesario completar una serie de minijuegos a cada paso para la consecución de los distintos objetivos.

Se pretende apoyar la idea de que todos los jugadores puedan entrar en contacto con otros voluntarios y detectar puntos críticos del entorno a través de mensajes y otros recursos geolocalizados. Así mismo, existirá la posibilidad de detectar y potenciar los lugares donde tienen lugar acciones positivas (incluso contribuir económicamente). Es decir, la característica fundamental: los usuarios dispondrán de la capacidad de ayudar a la sociedad mientras juegan y avanzan en Progrezz.

De acuerdo a lo anterior, pretende funcionar como una red social solidaria y plataforma de difusión de acciones y movimientos sociales y de voluntariado. Permitiendo de algún modo visualizar las acciones positivas que suceden en nuestro entorno tanto a nivel local como global. Por supuesto, teniendo presente en cada momento el componente de gamificación que rodea cada acción que realizan los jugadores de Progrezz.

1.4 Aportación

Respecto al tema del presente trabajo y su relación con los juegos serios, como se ha mencionado, la finalidad reside en la ampliación y mejora de Progrezz. Para ello, se han desarrollado actividades orientadas a la creación de un módulo de realidad aumentada geolocalizada.

La idea principal, que será detallada en las secciones posteriores, ha sido la de crear una herramienta que permita visualizar y representar en la interfaz[9] del juego toda clase de elementos y puntos de interés geolocalizados (para apoyar la idea que se ha ilustrado anteriormente de la actividad de detección y recolección de mensajes y otros recursos). Empleando la realidad aumentada, permite representar distintos elementos ficticios el juego en el visor de la realidad del jugador; así como también está disponible la posibilidad de interacción con los mismos.

[9] Medio que permite la comunicación entre un usuario y una máquina.

Al igual que Progrezz, consiste en un proyecto de software libre de código abierto (licencia MIT [A.3. Licencia MIT]), de modo que realmente podrá ser utilizado por cualquier desarrollador que precise un módulo de estas características. También se ha diseñado como aplicación web, tomando decisiones lo más genéricas posibles, teniendo siempre en mente la pretensión de que funcione en el mayor número de dispositivos posible (siendo la única restricción contar con acceso a geolocalización) con el objetivo de que sea accesible para cualquier persona.

Capítulo 2.
Antecedentes y estado actual

2.1 Contexto de los juegos serios

A pesar de que los juegos serios han existido mucho antes de la aparición de los dispositivos electrónicos en el mundo del entretenimiento, sobre todo con propósito educativo, el potencial de la tecnología actual ofrece un abanico extenso de posibilidades en cuanto a la creación de aplicaciones con funciones como acudir a la solidaridad de las personas y potenciar las acciones sociales.

Unido al factor de gamificación, permite generar un mayor atractivo, realzando su difusión y utilización. Son aspectos fundamentales, aún más en el caso de Progrezz, tratándose este de un juego que pretende ser una plataforma de difusión de acciones con impacto social positivo, y mantener en contacto a una comunidad que entiende la importancia de la solidaridad.

Aprovechando el auge de los dispositivos móviles y las posibilidades que ofrecen (gracias tanto a sensores y nuevas tecnologías, como por su amplia propagación en la sociedad), se ha decidido hacer uso de recursos tales como la geolocalización, el giroscopio[10], y la cámara de vídeo, para crear un visor de realidad aumentada que pueda apoyar las ideas anteriores.

2.2 Estado de Progrezz

Hasta el momento se han centrado los esfuerzos en generar una aplicación sencilla funcional, con el objetivo de que el juego esté disponible desde sus inicios, y pueda ir creciendo a medida que lo hace su comunidad.

---

[10] Dispositivo mecánico que sirve para medir/manipular la orientación en el espacio, de algún aparato o vehículo.

De acuerdo con esta idea, tanto en el cliente[11] como en el servidor[12], se ha cumplido con todos los requisitos básicos propuestos:

- 'Front-end'[13]:
  - Interfaz sencilla que permite la autentificación del jugador.
  - Mapa para la visualización de los mensajes (divididos en fragmentos).
  - Información acerca del perfil del jugador.
  - Lista de mensajes/fragmentos recolectados por el jugador, clasificados por estado (leídos, bloqueados,...).
  - Minijuego de desbloqueo de mensajes cifrados.
- 'Back-end'[14]:
  - Base de datos funcional con información acerca de los jugadores, mensajes, y fragmentos, entre muchas otras características.
  - Peticiones REST[15] y WebSockets[16] para solicitar y actualizar la información.

Entre los recursos empleados, se encuentra HTML5[17], CSS3[18], JavaScript[19], para la creación de la aplicación web; mientras que se emplea Ruby[20] (con el

---

[11] Aplicación informática o equipo que accede a recursos de otro equipo o servidor.

[12] Aplicación informática capaz de atender peticiones de un cliente y enviar respuestas en consecuencia.

[13] Parte del software que permite la interacción de los usuarios y la recolección y exposición de datos.

[14] Parte del software que procesa la entrada y genera una respuesta para los usuarios.

[15] Mecanismo de solicitud de información a un servidor por parte de un cliente, que emplea el estilo de arquitectura de desarrollo web REST.

[16] Tecnología que proporciona un canal de comunicación simultánea y bidireccional, entre navegadores y servidores web.

[17] Versión 5 del lenguaje de marcado HTML, estándar para la elaboración de páginas web.

[18] Versión 3 del lenguaje CSS, que permite especificar hojas de estilo para definir la presentación visual de documentos escritos en HTML.

framework Sinatra[21]) para el servidor, junto con Neo4j[22] para la creación y gestión de la base de datos.

A fecha de finalización del Trabajo de Fin de Grado sobre el que trata el presente documento, con la incorporación del visor de realidad aumentada, el juego puede ser considerado como 'jugable'. De acuerdo al estado descrito anteriormente, lo único que se precisaba para llegar a esta nueva etapa era una forma de detectar y recolectar los objetos geolocalizados, funcionalidad proporcionada por el visor que se ha desarrollado.

---

[19] Lenguaje de programación interpretado principalmente utilizado en el lado del cliente, para integrar código interpretable por el navegador, en una página web.

[20] Lenguaje de programación interpretado, utilizado de forma común en manejo de servidores y aplicaciones web.

[21] Gema (librería) de Ruby que consiste en una infraestructura para la creación de aplicaciones web.

[22] Motor de bases de datos que almacena información en forma de grafos.

Capítulo 3. Objetivos

3.1 Actividades principales

En líneas generales, para apoyar la idea de integración del mundo real con el proporcionado por Progrezz, se pretendía crear una aplicación para smartphones capaz de acceder a la cámara de vídeo de los dispositivos para funcionar a modo de visor. Así mismo, es necesario realizar el dibujado de distintos objetos en la escena. Por supuesto, dada la característica de geolocalización del visor, es necesario utilizar la señal GPS[23] para gestionar la posición y visualización de cada objeto en pantalla.

Es de vital importancia el proporcionar alternativas cuando surge algún problema de acceso a las distintas tecnologías, puesto que se pretendía que la utilización del módulo sea lo menos restrictiva posible, requiriendo sólo el acceso a las coordenadas geográficas (imprescindible en un módulo que se basa en la geolocalización). Como funcionalidad complementaria de cara al jugador, se debe contar con indicadores que muestren qué tecnologías están disponibles y cuáles no, posibilitando su activación/desactivación (cuando sea posible).

Como no basta con simplemente conocer la posición de los objetos, se debe contar con un sistema de gestión de eventos para permitir la interacción de los jugadores con los mismos, ya sea para recolectarlos, mostrar algún tipo de información, etc...; es decir, se debe permitir, a aquel que emplee el módulo, decidir el resultado de la interacción con cada objeto.

Por último, se ha planteado la posibilidad de emplear el visor en modo "Side By Side", una forma de simular visión estereoscópica (en 3D) gracias a la división vertical de la pantalla en dos partes, de tal manera que cada parte

[23] Sistema de posicionamiento global que permite determinar, con gran precisión, la posición de un objeto en el mundo.

muestra la misma escena, pero desde puntos desplazados unos centímetros entre sí (representando las posiciones de los ojos). De esta manera se consigue simular un efecto de profundidad cuando se utilizan gafas de realidad virtual basadas en dispositivos móviles, donde cada ojo sólo visualiza la parte de la pantalla que le corresponde.

3.2 Plan de trabajo

Según se ha definido anteriormente, las tareas a realizar en el Trabajo de Fin de Grado se desglosarían en una serie de fases:

1. Diseño de estructura y funcionalidades del visor de realidad aumentada. [4.1]
2. Análisis de tecnologías y recursos necesarios para la creación del módulo de realidad aumentada geolocalizada. [4.2]
3. Implementación modular del visor de realidad aumentada. [4.3.1]
4. Indicadores de tecnologías disponibles y gestión de eventos derivados. [4.3.4]
5. Sistema de interacción con los elementos geolocalizados. [4.3.2]
6. Funcionalidad de visión estereoscópica para gafas de realidad virtual. [4.3.5]

Es necesario además, aunque estrictamente no debe ser considerado como parte de la realización del Trabajo de Fin de Grado, mencionar la integración del visor, una vez finalizada la implementación de todas las funcionalidades, en la aplicación de Progrezz. [4.3.6]

También es importante tener en cuenta el testeo y la comprobación de que cada funcionalidad presenta el comportamiento esperado en dispositivos móviles (ya que, como es lógico, se desarrolla en un computador personal); así como la generación de documentación en cada fase, y de (dada la condición de software libre) una guía básica de instalación y uso del módulo en cualquier proyecto. [4.4]

Capítulo 4. Desarrollo

4.1 Diseño

La primera fase del desarrollo del proyecto ha consistido en determinar exactamente con qué funcionalidades debe contar el visor, cómo estructurarlas e interrelacionarlas garantizando su correcto funcionamiento, y los lenguajes de programación utilizados para lograrlo.

4.1.1 Estructura

Respecto al último punto, de acuerdo a las características de Progrezz, es lógico que se desarrolle como aplicación web, y se toma la decisión de utilizar JavaScript. Evidentemente, es necesaria la ejecución y realización de pruebas, por lo que se asociarían los scripts[24] a un documento estructurado con HTML5 y empleando CSS3 para los temas de estilo. Además, para probar el funcionamiento de la aplicación de forma local en dispositivos móviles, se decide crear un servidor muy simple, utilizando Ruby (con la librería[25] Sinatra) para tal labor. La razón de seleccionar estas opciones es básicamente la comodidad; se trata de lenguajes muy extendidos, y dado que Progrezz emplea exactamente los mismos, permitirá evitar posibles conflictos y aportar mayor comodidad.

[24] Fichero de código fuente legible y ejecutable.

[25] Conjunto de implementaciones funcionales, que ofrecen una interfaz bien definida para la funcionalidad que se invoca.

Para llevar el control de versiones y facilitar el desarrollo y gestión de la aplicación, se ha creado un repositorio [A.1. Repositorio de GitHub] empleando GitHub[26], con una estructura de directorios[27] muy sencilla:

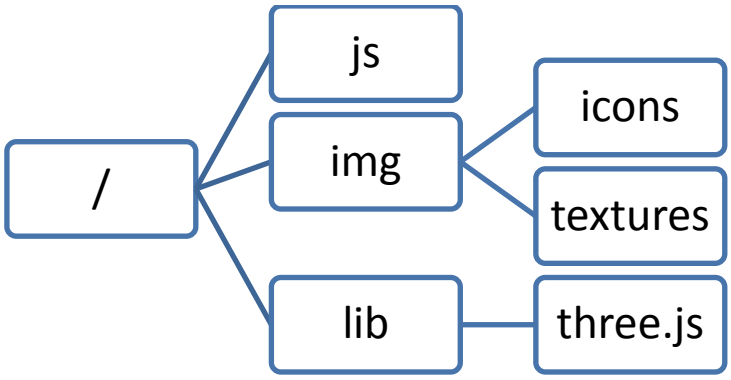

Figura 4.1. Estructura de directorios

- 'js': contenedor de los scripts del proyecto.
- 'img': conjunto de imágenes empleadas, distribuidas por categorías.
- 'lib': ubicación de las dependencias de la aplicación.

Nota: El contenido concreto de los subdirectorios del proyecto se irá detallando en posteriores apartados.

Es importante tener en cuenta que no se emplean hojas de estilo propias del proyecto: la asignación de estilos CSS3 a los elementos del visor se ha realizado directamente desde código JavaScript, para liberar de la inclusión de ficheros innecesarios a aquellos que empleen el módulo.

En lo relativo al directorio raíz del módulo, contiene otros elementos a tener en cuenta:

- 'index.html': Documento empleado para las pruebas, que constituye un ejemplo de empleo del visor.
- 'main.rb': Programa escrito en Ruby que permite el lanzamiento del servidor utilizado en pruebas locales de funcionamiento.

[26] Plataforma de desarrollo que permite el alojamiento de proyectos en repositorios utilizando el control de versiones Git.

[27] Contenedor virtual en el que se almacenan agrupaciones de archivos de datos y otros subdirectorios.

- 'README.md' [A.2. Fichero README.md]: Documento que expone brevemente la información sobre el módulo y funciona a modo de guía básica de uso e instalación.
- 'LICENSE' [A.3. Licencia MIT]: Descripción de la licencia MIT.

4.1.2 Funcionalidades

En cuanto a las funcionalidades y su estructura, se decidió que el visor de realidad aumentada estaría formado por una serie de submódulos con finalidades bien diferenciadas, implementando cada uno en un script distinto:

- ARProgrezz.js: Se ocupa únicamente de la creación del espacio de nombres[28] ARProgrezz.
- Utils.js: Funciones y otras utilidades variadas, que emplean el resto de submódulos del proyecto.
- Support.js: Dada la cantidad de tecnologías que utiliza el visor (geolocalización, giroscopio y cámara de vídeo) y los problemas que pueden surgir en el acceso a las mismas, se cuenta con este submódulo de apoyo. Es el encargado de comprobar la disponibilidad de ese acceso, y proporcionar indicadores, tanto de si su utilización es funcional, como del estado de habilitación en cada momento (puesto que se pueden activar y desactivar, en caso de que sea posible, a elección del usuario).
- Video.js: Clase que se ocupa del acceso y gestión del vídeo, para su visualización como fondo de la escena del visor. Así mismo, en caso de no estar disponibles (o desactivados) los datos de la cámara de vídeo, funcionaría de forma idéntica, pero mostrando en dicho situación un fondo alternativo.
- PositionControls.js: Clase que gestiona el desplazamiento del dispositivo en el mundo real. Por un lado monitoriza las coordenadas del dispositivo (mediante la señal GPS), actualizando las posiciones de los objetos en el visor de acuerdo a su distancia y posición en el mundo real respecto al jugador; y por el otro, de acuerdo a la orientación del dispositivo (conocida gracias al giroscopio), se determina en qué dirección y sentido está observando el jugador (y por tanto qué se debe mostrar). También

[28] En programación, contenedor que permite la agrupación de un conjunto de elementos bajo un mismo identificador único.

se ocupa de proporcionar una alternativa en caso de no estar habilitado el acceso al giroscopio, para permitir la visualización de la escena en todas las direcciones (sólo la alternativa al giroscopio, ya que, como se ha mencionado anteriormente, la geolocalización es imprescindible).

- Object.js: Se ocupa de proporcionar los objetos posibles que pueden ser creados y añadidos a la escena de realidad aumentada. Del mismo modo que se encarga de su actualización, y del lanzamiento de eventos.
- Preloader.js: Consiste simplemente en una pantalla de carga, que informa de que el visor se está iniciando.
- Viewer.js: Representa el visor de realidad aumentada en sí, que contiene la escena de realidad aumentada, donde se realiza el dibujado de los objetos virtuales. Emplea todos los demás submódulos, y funciona a modo de controlador, iniciando cada componente e interconectándolos unos con otros.

Nota: Cada script está asociado a su propio espacio de nombres, de la forma 'ARProgrezz.nombre_script'.

Tras la implementación de todas las funcionalidades, los scripts anteriores se han minimizado empleando JSMin[29] (tras chequear problemas con JSLint[30]) y encapsulado en el orden correcto, generando el script 'ARProgrezz.min.js', que se puede encontrar en el directorio 'js' del proyecto.

4.2 Análisis

Posiblemente la fase de análisis puede considerarse como la más costosa y extensa llevada a cabo en el proyecto. Dada la cantidad de recursos requeridos en la creación de los distintos componentes del módulo, fue necesario probar muchas opciones y tomar una serie de decisiones. La finalidad del apartado actual es la de enumerar esas opciones barajadas, explicar aquellas que fueron escogidas, y las razones de tal decisión.

---

[29] Herramienta de filtrado que elimina comentario y espacios innecesarios de un fichero JavaScript, reduciendo su tamaño.

[30] Herramienta que revisa errores, problemas y la calidad del código de un fichero Javascript.

4.2.1 Aplicación

En un principio, el equipo encargado de Progrezz empleaba PhoneGap[31] para crear el cliente de la aplicación como app[32] para Android[33]. El mecanismo que utiliza es la de traducir a una aplicación nativa utilizando la clase WebView proporcionada por dicho sistema operativo, una ventana que cuenta con todas las características de cualquier navegador.

Por esta razón existía la posibilidad tanto de crear el módulo como una librería JavaScript, completamente web, o hacer uso de los plugins[34] de PhoneGap, que funcionan a modo de puente de acceso a recursos como la cámara o el GPS.

Ventajas generales:

- PhoneGap:
  - Acceso a los recursos del dispositivo móvil de forma nativa y eficiente mediante los plugins de PhoneGap.
- Aplicación web:
  - Accesible desde cualquier dispositivo con navegador.

Desventajas generales:

- PhoneGap:
  - Implica la instalación de la aplicación en el dispositivo.
  - Es necesario generar una aplicación para cada sistema operativo en el que se quiera permitir el acceso.
- Aplicación web:
  - El acceso a los recursos del dispositivo desde web, podría no funcionar correctamente o del mismo modo en todos los modelos o navegadores, puesto que aún se carece de estándares.

---

[31] Plataforma para el desarrollo de aplicaciones móviles utilizando herramientas de desarrollo web como JavaScript, HTML5 y CSS3.

[32] Aplicación móvil, una aplicación informática diseñada para ser ejecutada en dispositivos móviles (smartphones, tablets,...).

[33] Sistema operativo diseñado por la empresa Google, para dispositivos móviles con pantalla táctil.

[34] Un complemento que aporta una función nueva y específica a una aplicación principal.

La decisión en este aspecto debía ser tomada, aparte de considerando las ventajas y desventajas comentadas, atendiendo a las oportunidades que ofrece cada opción respecto al acceso y utilización de los distintos recursos, que se detallan a continuación.

4.2.2 Vídeo

En un primer momento, se intentó acceder a la cámara desde web, mediante la API getUserMedia, componente del proyecto WebRTC[35], creado por el W3C[36], que pretende estandarizar la comunicación con la cámara y el micrófono desde el navegador. Esta opción, aunque condicionada a la compatibilidad con el navegador, es válida pues permite acceder a la cámara de los dispositivos en tiempo real.

	Chrome	Android	Firefox	Internet Explorer	Opera	Safari
Escritorio	21	-----	17	No soportado	18	No soportado
Móvil		No soportado	24	No soportado	12	No soportado

Tabla 4.1. Compatibilidad de la API getUserMedia.

Ahora bien, una vez se ha obtenido el flujo de datos de la cámara de vídeo, se plantean opciones para su visualización:

- **Dibujado en Canvas:** Se van obteniendo imágenes a partir de los datos del vídeo y se dibujan en un elemento Canvas de HTML. La ventaja de esta opción es que se admitiría en cualquier navegador que soporte HTML5, y permitiría el tratamiento de los datos del vídeo con una amplia gama de métodos (redimensionado libre, modificación de colores,...); sin embargo, el proceso de dibujado, imagen a imagen, es terriblemente lento en el caso de dispositivos móviles (llegando a presentar un desfase respecto a la realidad de más de un segundo). Teniendo en cuenta la condición de visor, el retardo de actualización

[35] API creada por el W3C, para permitir a las aplicaciones basadas en navegador, el acceso a la cámara, micrófono, llamadas de audio/vídeo, y el uso compartido de archivos P2P.

[36] World Wide Web Consortium, un consorcio internacional que produce recomendaciones para la World Wide Web, y creador de las principales tecnologías web (URL, HTTP, y HTML).

debe ser ínfimo, para que el jugador tenga la sensación de estar observando el mundo real y obtenga una respuesta coherente a sus acciones. Por tanto, esta opción es inviable.

- **Etiqueta Video**: Se trata de una etiqueta que proporciona HTML para la visualización de vídeos en distintos formatos, o flujos de vídeo (como por ejemplo el obtenido de la cámara del dispositivo). Esta opción es considerablemente más rápida (sin sobrepasar el límite de retardo aceptable), pues se evita la obtención de las imágenes una a una a partir del flujo de vídeo, y su posterior dibujado. No obstante, es una opción algo más restrictiva en la anterior, pues no se permite su alteración y el redimensionado sólo es posible respetando la relación de aspecto del vídeo: se ajusta al tamaño de la etiqueta (de forma centrada), de tal manera que ocupa todo el espacio que es capaz de abarcar sin deformarse, y muestra espacios en blanco en las partes sobrantes (si no se ocupara al completo).

La otra posibilidad era acudir a alguno de los plugins de PhoneGap para obtener los datos de la cámara de vídeo:

- **Camera:** Permite acceder únicamente a la cámara de fotos para tomar una captura y transmitirla a la aplicación.
- **Media:** Permite grabar y reproducir únicamente ficheros de audio en el dispositivo.
- **Capture:** Permite la obtención de imágenes, vídeo o audio desde la cámara o micrófono. Sin embargo, emplea las propias aplicaciones que utiliza el dispositivo para realizar capturas de fotos, audio, o vídeo.
- **PhoneRTC:** Plugin que simula WebRTC para aplicaciones basadas en Cordova[37], pero lamentablemente, sólo el componente que permite establecer llamadas de audio y vídeo.

Es decir, no se ha encontrado ningún plugin para PhoneGap que permita acceder de forma libre a la cámara de vídeo de un dispositivo en Android, tal y como se necesita para la creación del visor.

[37] Apache Cordova es una plataforma para la creación de aplicaciones nativas para dispositivos móviles, de la cuál PhoneGap es una distribución.

De modo que como posible alternativa al acceso web, estaría crear, en el desarrollo del trabajo, un plugin para PhoneGap que permita obtener los datos de la cámara de vídeo en Android, de forma nativa (en lenguaje Java), u optar por utilizar otro entorno de desarrollo que si cuente con un plugin válido (como Titanium); opciones cuya complejidad para realizar algo tan básico como mostrar los datos de la cámara de vídeo, provoca que no resulten demasiado atractivas.

También sería necesario tener en cuenta, que aunque se hubiera encontrado algún plugin válido u otra forma de acceso a la cámara, no se garantiza el funcionamiento de la etiqueta Video en PhoneGap (concretamente no se soportaba de forma completa en las WebView de Android hasta la versión 2.3), de cara a la visualización del flujo de datos.

4.2.3 Gráficos

Era necesario algún modo de dibujar sobre el vídeo los objetos de realidad aumentada; para lo que se barajaron las siguientes librerías que permitían la utilización de gráficos 3D en un entorno web:

- **Phoria.js**: Consiste en una librería de JavaScript para la creación de gráficos simples en 3D, mediante renderizado[38] sobre un elemento Canvas[39] (2D).
  La dificultad de utilización, podría residir en que se encuentra en una fase aún temprana de desarrollo, y únicamente cuenta con una serie de ejemplos a modo de documentación. Tras revisar dichos ejemplos, se concluyó que es una librería muy ligera, que posibilita el dibujado de objetos sencillos, y presenta un mecanismo para descubrir (aunque en principio sólo para eventos de ratón) cuándo los usuarios seleccionan algún objeto en el visor. Para ello se emplea un mecanismo que detecta las intersecciones entre los objetos y un vector cuya dirección depende de la posición de las pulsaciones en pantalla.

[38] Proceso de generación una imagen o vídeo a partir de un modelo en 3D.

[39] Elemento HTML incorporado en HTML5 que permite la generación de gráficos estáticos y animaciones.

Sería necesario acudir a eventos externos, de HTML, para la gestión de eventos de toque en dispositivos móviles, y en principio, si la adaptación para dispositivos móviles funciona, serviría para lo que se necesita.

- **Three.js**: Librería escrita en JavaScript para la creación y visualización de gráficos en 3D. También es muy ligera, y cuenta con gran cantidad de opciones y efectos, más allá del dibujado de objetos simples (como la carga de modelos 3D); así como ejemplos de su utilización, y un componente importante de documentación.

  La diferencia respecto a la anterior reside en que está pensada para renderizar mediante WebGL[40], raramente soportado en navegadores de dispositivos móviles (y aunque lo hagan, suele tratarse de tecnología experimental que aparece desactivada por defecto, y cuya activación no constituye una acción trivial para un usuario estándar). No obstante, también cuenta con una herramienta de renderizado en Canvas (más lenta), como alternativa.

  Respecto a las interacciones con objetos, la librería proporciona la clase 'Raycaster', que permite detectar de forma sencilla los objetos seleccionados, de forma similar al caso anterior, a partir de las coordenadas de la pantalla sobre las que el usuario realiza una pulsación.

Ambas opciones serían ajenas a otras decisiones tomadas en la fase de análisis, pues independientemente de la forma de obtener el vídeo, se podría realizar el dibujado superponiendo la escena. Y en cuanto a la decisión de crear una aplicación totalmente web, o acudir a Phonegap, aunque la WebView que utiliza este último para traducir a Android nativo no soporta WebGL, existen plugins como WebGLGap[41] que permiten su empleo; y de todas formas en ambos casos se admitiría el dibujado en Canvas.

---

[40] Especificación estándar que se está desarrollando para la visualización de gráficos 3D en navegadores web.

[41] Proyecto de creación de un plugin para PhoneGap que proporciona un subconjunto de la interfaz de programación de WebGL para dispositivos móviles.

4.2.4 Geolocalización

A la hora de acceder al sensor GPS de un dispositivo móvil, la opción más extendida consiste en acudir a la API Geolocation creada por el W3C. Proporciona una serie de funciones para obtener, tras petición de permiso previa (pues se debe respetar la privacidad del usuario), las coordenadas geográficas (latitud y longitud) del dispositivo.

	Chrome	Android	Firefox	Internet Explorer	Opera	Safari
Escritorio	5	-----	3.5	9	10.6 Eliminado en 15 Reintroducido en 16	5
Móvil	Desconocido	Desconocido	4	Desconocido		Desconocido

Tabla 4.2. Compatibilidad de la API Geolocation.

Si se utilizara PhoneGap tampoco habría ningún problema en relación a la geolocalización, pues existe el plugin Geolocation, que está basado en la API anterior, y proporciona exactamente las mismas utilidades para el acceso al sensor GPS. Una ventaja podría ser que sólo se pide permiso de acceso al GPS cuando se instala la aplicación, al contrario que en el caso anterior, en el que se solicita cada vez que se ejecuta una función con necesidad de acceso al GPS.

4.2.5 Giroscopio

El otro sensor al que se desea acceder, para conocer la orientación del dispositivo, es el giroscopio. Para tal fin, W3C proporciona el DeviceOrientationEvent, que permite a los desarrolladores web, desde el navegador, detectar cuando cambia la orientación, obteniendo información acerca de su posición: los ángulos de rotación (alpha, beta y gamma), en cada uno de los tres ejes de coordenadas (x, y, z). No es ni mucho menos un estándar, se trata aún de tecnología experimental sujeta a cambios, por lo que no se garantiza que funcione en cualquier navegador o dispositivo.

	Chrome	Android	Firefox	Internet Explorer	Opera	Safari
Escritorio	7	-----	3.6	Desconocido	Desconocido	Desconocido
Móvil		3		No soportado	No soportado	4.2

Tabla 4.3. Compatibilidad del DeviceOrientationEvent.

Respecto a PhoneGap, en principio no se ha encontrado ningún plugin válido, que permita obtener los ángulos de rotación respecto a los tres ejes de coordenadas. Sí que se permite el acceso a otros sensores, como la brújula (proporciona la dirección a la que apunta el dispositivo: la rotación en un solo eje) o el acelerómetro (mide la aceleración de los movimientos en los tres ejes), pero nada en cuanto al giroscopio.

4.2.6 Conclusiones

Después de considerar todas las posibilidades anteriores, se optó por realizar el módulo completamente web, pues PhoneGap realmente no presenta ninguna ventaja destacable. Se planteó su empleo la idea de poder acceder de forma nativa a los sensores, pero al carecer de plugins para el caso del giroscopio o el vídeo, pasa a ser inviable.

De modo que, la única posibilidad aparte de la aplicación web, sería realizar todo de forma nativa para Android; lo que se desviaría de los objetivos fundamentales que se pretenden alcanzar en el presente trabajo. Para empezar, se tendría que ignorar el objetivo de que pudiera utilizarse en cualquier dispositivo con geolocalización (pues habría que crear versiones para todos los sistemas, y no solo para Android). De hecho ya existe un proyecto de este tipo, Wikitude, un entorno de desarrollo de realidad aumentada y reconocimiento de imágenes muy completo implementado de forma nativa para Android, iOS y BlackBerry10, que incluso cuenta con un plugin para su utilización en PhoneGap.

Creando una aplicación web, por el contrario, contaría con todas las piezas necesarias para construir el módulo de realidad aumentada geolocalizada; incluso permitiría probar la capacidad real de la tecnología web existente hasta el momento.

Resumen de Tecnologías				Navegador	PhoneGap
Vídeo	**Acceso**			✓	×
	Visualización	**Canvas**		×	×
		Etiqueta Video		✓	✓
Gráficos	**Phoria.js**	**Renderizado**	**Canvas**	✓	
			WebGL	×	
		Eventos		✓	

		Documentación		✗	
	Three.js	**Renderizado**	**Canvas**	✓	
			WebGL	✓	
		Eventos		✓	
		Documentación		✓	
GPS				✓	✓
Giroscopio				✓	✗

Tabla 4.4. Resumen del análisis de tecnologías.

Partiendo de lo anterior, las opciones definitivas serían las siguientes:

- **Vídeo:** Acceso mediante la API getUserMedia, y visualización mediante la etiqueta vídeo, que aunque impida el redimensionado libre, funciona de forma fluida.
- **Gráficos:** Librería Three.js, puesto que ofrece las mismas posibilidades que Phoria.js y más, como el renderizado en WebGL y múltiples utilidades que cuenta con ejemplos y documentación, lo que favorece la ampliación futura del visor.
- **GPS:** API Geolocation para navegador.
- **Giroscopio:** DeviceOrientationEvent de HTML.

4.3 Implementación

En el presente apartado se describirán las tareas realizadas, problemas encontrados, soluciones y alternativas planteadas, y decisiones tomadas; con el objetivo de elaborar el código del módulo de realidad aumentada.

4.3.1 Visor

Se podría considerar al visor como el núcleo del módulo, pues aparte de ser el que realiza la representación visual del módulo, también es el encargado de gestionar y coordinar el resto de recursos para que funcionen e interactúen de la forma adecuada.

El vídeo constituye la base del visor, y se obtiene de la forma indicada en la fase de análisis, mediante la API getUserMedia. Lo que permite es realizar una petición de acceso a la cámara de vídeo de un dispositivo, a lo que se

responde con un flujo de datos. Ubicando dicho flujo en una etiqueta Video de HTML, se puede visualizar lo que capta la cámara de vídeo en tiempo real.

El vídeo se amplía todo lo posible, y se centra (tanto verticalmente como horizontalmente), dado que la etiqueta Video no permite su expansión, tal y como se mencionó anteriormente. Esto implica que por los laterales, o en la parte superior e inferior, quedarán espacios en blanco, dependiendo del limitante (alto o ancho respectivamente). Sin embargo, a la hora del dibujado de los objetos, sobre el que se hablará a continuación, sólo se contemplará el área del visor en el que se visualiza el vídeo.

Un detalle que hubo que tener en cuenta, fue las distintas formas de acceder a la cámara de vídeo, ya que en caso de que los dispositivos cuenten con más de una cámara, como muchos de los smartphones que se encuentran actualmente en el mercado (normalmente frontal y trasera), se debe proceder de una forma u otra dependiendo del navegador. En el caso de Firefox, es el usuario el que decide qué cámara compartir (por lo que se solicita el vídeo con las opciones por defecto), pero por ejemplo, en Chrome, se comparte por defecto la cámara delantera, lo que implica que haya que revisar los dispositivos disponibles desde código, y seleccionar el que corresponde a la cámara trasera. A efectos prácticos, se detecta previamente el navegador en el que se está ejecutando el visor y se procede de una forma u otra dependiendo del mismo. En principio sólo se contempla la alternativa de Chrome: en el resto de los casos, se actúa con el comportamiento por defecto.

Respecto a la escena de realidad aumentada en 3D, funciona como un lienzo transparente (para que se vea el vídeo de fondo) sobre el que se dibujan los objetos geolocalizados; haciendo uso de una serie de elementos de la librería Three.js:

- **Scene:** Representa una escena 3D a la que se le pueden añadir otros elementos.
- **Camera:** Representa la visión del jugador, lo que se visualiza en la escena es aquello que enfoca una cámara.
- **Renderer:** Renderizador, es el que se encarga del dibujado, a partir de una escena y una cámara. El tipo de renderizador dependerá de si está disponible WebGL; si lo está, se empleará el WebGLRenderer, y en caso contrario el CanvasRenderer.

Una vez creados los distintos elementos, se inicia un proceso de animación de la escena. Consiste en un bucle, en el que en cada iteración, se actualizan todos los elementos de la escena (la cámara que representa al jugador, las posiciones de los objetos,...), y se dibuja en pantalla.

La idea, como ya se ha mencionado, es que sólo sea imprescindible la geolocalización para emplear el módulo, por lo que para el caso en que se produjera algún error al obtener los datos del vídeo, o se denegara el permiso de acceso, es necesario contar con una alternativa. Se decidió utilizar una imagen, que simulara el panorama de una ciudad, una escena alrededor de la cámara que respondiera con un efecto visual realista a los cambios en la orientación de la cámara.

Para ello se emplea una imagen equirectangular de una ciudad [Figura 4.2], un tipo especial de imagen que, aplicada de la forma adecuada, es capaz de simular un entorno que envuelve al jugador.

Figura 4.2. Imagen equirrectangular de una ciudad.

Su utilización es bastante sencilla: simplemente se crea una esfera haciendo uso de las herramientas de la librería Three.js, se le aplica la imagen como textura, y se sitúa de tal manera que la cámara que representa la visión del jugador quede en su interior (ubicando el centro de la esfera en el origen de coordenadas).

Nota: Se tomó como referencia el ejemplo Equirectangular Panorama de la librería Three.js, en el que se utiliza una imagen equirectangular para simular un panorama en el interior de una casa.

También es importante mencionar, que se ha implementado la detección de ciertos eventos (redimensionado de la ventana, o cambio de orientación), para modificar los tamaños del vídeo, la escena y otros elementos, de acuerdo a los cambios. Es decir, ante cualquier cambio en las dimensiones, relación de aspecto u orientación, el visor se adaptará automáticamente a las nuevas características de su contenedor.

Otro aspecto a tener en cuenta, es que lo recomendable es utilizar el navegador a pantalla completa cuando se ejecute el visor. La forma de realizarlo desde código, es emplear la API Fullscreen, tecnología experimental aún en desarrollo.

	Chrome	Android	Firefox	Internet Explorer	Opera	Safari
Escritorio	20	-----	10	11	12.10	5.1
Móvil	28	Desconocido		Desconocido	Desconocido	Desconocido

Tabla 4.5. Compatibilidad de la API Fullscreen.

El problema en caso de que esté disponible en el navegador, reside en las restricciones de utilización del mismo: sólo se puede utilizar un evento lanzado por el usuario, de ejecución corta. Es decir, no se puede utilizar para poner el navegador a pantalla completa nada más iniciar el visor. Se ha creado una función que realiza la petición de pantalla completa, y se ejecuta cuando tiene lugar un evento de pulsación en pantalla (o de ratón), para que se establezca la primera vez que el usuario pulse la pantalla.

Es recomendable además, por comodidad, utilizar el navegador con la pantalla apaisada cuando se ejecute el visor. Para ello se dispone de la funcionalidad Screen.lockOrientation, también tecnología experimental, que se ocupa de bloquear la orientación de la pantalla de una determinada forma. Se ha implementado de tal manera que, si está disponible, se utilice para bloquear la orientación en modo apaisado. De todas formas, como se ha dicho, en caso de que no se pudiera bloquear, el visor se ajustaría automáticamente a cualquier cambio de orientación.

	Chrome	Android	Firefox	Internet Explorer	Opera	Safari
Escritorio	38	-----	Sí	11	No soportado	No soportado
Móvil		No soportado		No soportado		

Tabla 4.6. Compatibilidad de la funcionalidad Screen.lockOrientation.

Para terminar, como se ha empezado a ver, es necesario realizar una de ajustes y solicitar una serie de permisos, de forma previa a la utilización del visor, que suele llevar unos cuantos segundos. Para hacer entender al usuario que se está iniciando el visor, se ha creado una pantalla de carga con una imagen giratoria en su centro; mostrándola al empezar la inicialización y eliminándola a su término.

4.3.2 Creación e interacción con objetos

El visor proporciona la función 'addObject' para la creación y adición de los objetos, a la que se le deben indicar una serie de opciones:

- **Coordenadas (latitud y longitud**): Coordenadas geográficas del objeto, empleadas para determinar su ubicación en la escena del visor.
- **Tipo de objeto:** En principio se permite sólo incluir un objeto básico, que consiste en un tetraedro giratorio, al que se le ha aplicado una textura, utilizando las herramientas que proporciona la librería Three.js. La opción está pensada para la futura ampliación, permitiendo añadir toda clase de objetos.
- **Función de selección:** Se trata de aquel código que debe ejecutarse en caso de que el objeto sea seleccionado mediante un evento de pulsación en pantalla (sobre el que se hablará a continuación), de cara a que sea aquel que utilice el módulo, el que decida que debe ocurrir cuando seleccionen un objeto (ya sea mostrar información o redireccionar a otra página).
- **Recolectable:** Opción que indica si el objeto es recolectable (sólo se puede seleccionar una vez, desapareciendo tras su selección), o puede ser seleccionado todas las veces que se desee (con la consiguiente llamada a la función anterior).

Una vez se añaden los objetos a la escena, podrán visualizarse, si la cámara está apuntando hacia ellos, y si se está lo suficientemente cerca. Para determinar qué se entiende como 'cerca', es necesario especificar, en el momento de creación del visor, un rango de visión: la distancia máxima (en metros), a la que se pueden ver los objetos. Este valor es el que se utiliza para establecer el tamaño al que se dibujan los objetos. La asignación de tamaño se realiza de tal modo, que al rango máximo, el objeto se vea siempre al mismo

tamaño, independientemente de cual sea el valor del rango, y se irá viendo más grande, de forma uniforme (crecerá más rápido cuanto menor sea el rango máximo), a medida que el jugador salve la distancia que los separa. Si el objeto se encuentra más lejos del rango máximo, simplemente no se verá.

Respecto a la gestión de eventos, es necesario detectar los eventos de ratón y de toque que se realizan sobre el navegador. A partir de estos eventos se obtienen las coordenadas en pantalla sobre las que ha pulsado el jugador. Ahora bien, con dichas coordenadas se puede detectar si se ha seleccionado algún objeto, realizando una serie de pasos:

1. Las coordenadas de la pulsación están entre 0 y el ancho del contenedor, y entre 0 y el alto del contendor. Hay que transformarlas al intervalo (-1,1) en ambos ejes.
2. Con las coordenadas, es posible 'lanzar un rayo' (raycast) en la dirección determinada por el vector con esas coordenadas, mediante la clase 'Raycaster' de la librería Three.js, tal y como queda ilustrado en el dibujo [Figura 4.3].
3. Se comprueban aquellos objetos que han sido 'alcanzados por el rayo', y si hubiera alguno, se escogería el primero con el que se ha chocado, descartando el resto. Tiene lugar una pequeño parpadeo en color rojo a modo de retroalimentación, para indicar que ha sido seleccionado, y en caso de ser un objeto recolectable, se verá reducido su tamaño hasta desaparecer mediante una sencilla animación.
4. Se ejecuta la función de selección de dicho objeto. En caso de no haberse especificado ninguna función, se proporciona una por defecto, que indica que el objeto ha sido capturado y se muestran sus coordenadas geográficas.

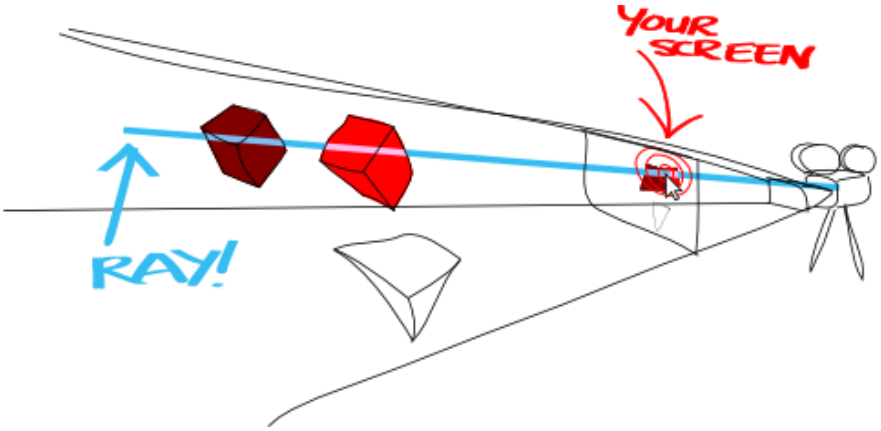

Figura 4.3. Representación del funcionamiento del 'Raycaster' de Three.js.

Nota: Se ha utilizado como referencia el ejemplo de Three.js de 'Canvas Interactive Particles', en el que se muestra cómo detectar los objetos seleccionados a partir de la pulsación en pantalla.

4.3.3 Sensores

La idea de utilización del giroscopio reside en monitorizar la orientación del dispositivo, para que los cambios en la misma se vean reflejados en la escena virtual.

Se utiliza el DeviceOrientationEvent como se había acordado anteriormente, un evento que emplea el giroscopio para detectar las variaciones de orientación de un dispositivo móvil. Una vez detectado un cambio de orientación, se lanza automáticamente un evento de tipo 'deviceorientation'. Lo que se hace es crear una escucha que detecta su lanzamiento, una función que se ejecuta cada vez que tiene lugar el evento, y que en este caso se ocupa únicamente de obtener, en cada ejecución, la información de la orientación del dispositivo en ese instante.

Esta información se proporciona mediante tres valores: 'alpha', 'beta' y 'gamma', que indican en grados los ángulos de rotación respecto a los ejes de coordenadas 'z', 'x' e 'y', respectivamente, de acuerdo a la posición inicial del dispositivo; tal y como se muestra en la figura [Figura 4.4].

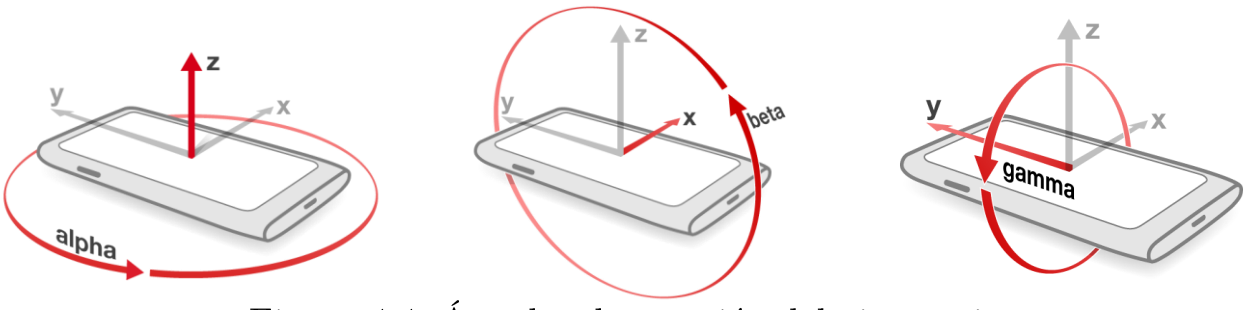

Figura 4.4. Ángulos de rotación del giroscopio.

Estas tres coordenadas angulares conforman un conjunto denominado ángulos de Euler, que sirven para especificar la orientación de un sistema de referencia de ejes ortogonales, respecto a otro sistema de referencia fijo. En realidad, lo que se utiliza son ángulos Tait-Bryan, que representan exactamente lo mismo, pero estos asignan 0º a un objeto en horizontal (del mismo modo que Three.js) en lugar de los 90º que tendría en ángulos de Euler.

También se debe tener en cuenta que la rotación de la cámara dependerá de la orientación de la pantalla del dispositivo, por lo que también es necesario detectar el modo: 'landscape' (apaisada) o 'portrait' (en vertical). Para ello se emplea otra escucha, en este caso para el evento OrientationChange, encargado de obtener el modo de orientación.

Ahora bien, esta información es utilizada, de forma periódica, en el momento que se actualiza y renderiza la escena 3D. La orientación de la cámara que representa al jugador se modifica de acuerdo a esos tres ángulos. La opción escogida en este caso ha sido la de utilizar cuaterniones[42], para lo que se requiere la realización de una serie de operaciones:

1. En primer lugar, modificar la disposición 'ZXY' de los ángulos proporcionada, por la convención 'YXZ'.
2. A continuación, se convierten los ángulos Tait-Bryan a una representación de un cuaternión unitario y se aplica a la cámara para modificar su orientación.

[42] Sistema numérico que extiende los números complejos, proporcionando los cuaterniones unitarios, una notación matemática para representar la orientación y rotación de objetos en tres dimensiones.

3. Se corrigen los valores del cuaternión para que se oriente hacia la parte trasera del dispositivo, en lugar de a la parte delantera.
4. Se corrigen los valores del cuaternión en función de la orientación actual de la pantalla.

Nota: Se ha utilizado como referencia el ejemplo 'Device Orientation Controls' de Three.js, en el que se realizan estas transformaciones, empleando como herramienta, las clases que proporciona la librería para la utilización de ángulos de Euler, vectores y cuaterniones, facilitando en gran medida las operaciones.

Esto tiene como finalidad que la cámara, que representa los ojos del jugador, rote sobre sí misma de forma idéntica a cómo lo ha hecho el dispositivo, y la escena que se visualice guarde coherencia con el movimiento.

Un problema encontrado es que se produce una vibración en pantalla, producto de la actualización de orientación. Esto se debe a que el giroscopio es muy sensible, reaccionando al mínimo movimiento, y los ángulos que proporciona tienen gran cantidad de decimales de precisión. Por esta razón, con el objetivo de ganar estabilidad, se ha reducido el número de decimales de precisión (mediante un redondeo) en la actualización del dispositivo, de manera que si la variación ha sido muy pequeña, en vez de rotarse la cámara una cantidad simbólica que en conjunto produce ese efecto de vibración, la cámara seguirá en la misma posición.

Por otra parte, como se trata aún de tecnología experimental, no se garantiza el correcto funcionamiento, por lo que en este caso es necesario proporcionar una alternativa cuando no se permita el acceso o no funcione correctamente. Se ha acudido a crear eventos de toque y de ratón, cuyo desplazamiento en pantalla se traduce en un desplazamiento en cada uno de los tres ejes de coordenadas. Utilizando la función 'lookAt' que proporciona la librería Three.js, se puede orientar la cámara hacia ese punto al que se ha desplazado la escena.

El otro sensor del que se hace usos es el GPS. La API Geolocation tratada en la fase de análisis proporciona una serie de funciones para acceder a las coordenadas geográficas de un dispositivo. La empleada en este caso es 'watchPosition'. Esta función lo que hace es acceder a lo largo del tiempo a la información del sensor para obtener la latitud y longitud del dispositivo,

ángulos que indican una posición sobre la superficie de la tierra. La latitud es un ángulo entre -90º y 90º, siendo el valor positivo en el hemisferio norte, y negativo en el sur; mientras que la longitud es un valor entre -180º y 180º, con el meridiano de Greenwich marcando los 0º, con valores negativos hacia su izquierda, y positivos hacia su derecha, tal y como se muestra en el dibujo [Figura 4.5].

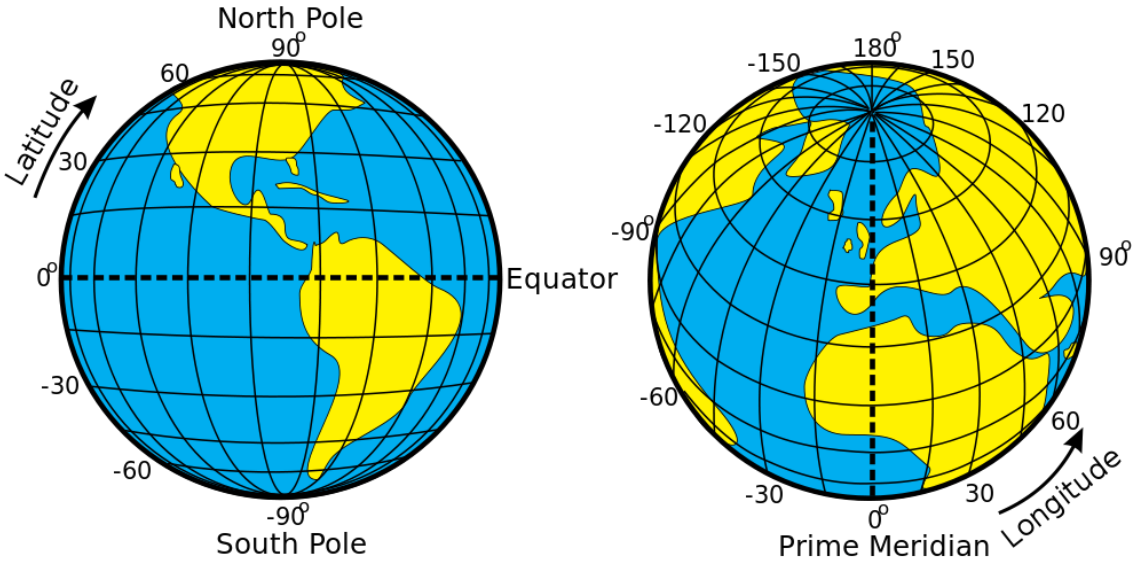

Figura 4.5. Representación de la latitud y la longitud.

Al igual que en el caso anterior, se ejecuta una función que almacena estos valores, y de forma periódica se emplean para actualizar la escena. Para ello se realiza una translación, buscando un equivalente a la latitud y longitud en la disposición de los ejes de coordenadas que representan el espacio. Fijándose en el sistema de coordenadas de Three.js [Figura 4.6], se puede observar que el eje Y es la altura, que para este caso no interesa en absoluto, pues la latitud y longitud solo indican la posición sobre la superficie, y no su elevación. En cuanto al resto de ejes, para guardar coherencia con lo que simboliza la cada una, se ha determinado que el eje X represente la longitud (ambos tienen sentido positivo hacia la derecha y negativo hacia la izquierda), y que el eje Z represente la latitud, pero invertida, pues el eje Z hacia abajo es positivo y hacia arriba es negativo, al contrario que la latitud.

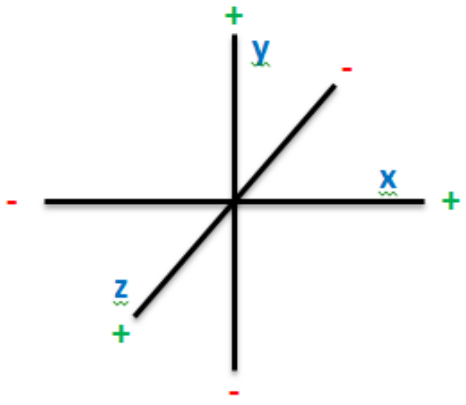

Figura 4.6. Representación de los ejes en Three.js.

Ahora bien, para actualizar la escena, se emplea la fórmula que calcula la distancia entre dos puntos dados por su latitud y longitud [B.1. Cálculo de distancias entre dos puntos (latitud, longitud).]. Para cada objeto, se estima la distancia en metros respecto a las coordenadas del jugador; por un lado, se calcula la distancia entre ambas latitudes (usando la longitud del jugador para los dos puntos) para asignar la posición en el eje Z, y por el otro, la distancia entre ambas longitudes (usando la latitud del jugador para los dos puntos) para asignar la posición en el eje X, en sentido inverso. De esta forma, se ubican los objetos alrededor del usuario, dependiendo de las coordenadas de cada uno, en función de las coordenadas del jugador, y adaptándose a los cambios de las mismas.

En un principio lo que se hacía era mover al jugador, es decir, se desplazaba la cámara en los ejes Z y X, dependiendo de la distancia en metros entre la latitud y longitud de un momento dado, respecto a una posición inicial del dispositivo que se obtenía al iniciar el visor. No obstante esto traía problemas, pues surgían conflictos con otras operaciones (como la de rotación) que se realizaba respecto al origen, y también implicaba desplazar el panorama (alternativa al vídeo); por lo que se optó, en lugar de desplazar al jugador, desplazar el resto de objetos en función de las nuevas coordenadas del dispositivo del jugador; o dicho de otro modo, cuando el jugador se mueve, no es él el que se desplaza, sino es el mundo el que se mueve a su alrededor en sentido opuesto.

4.3.4 Indicadores

Como se ha mencionado, en muchos casos se trabajó con tecnología experimental, de ahí que se creara un sistema encargado de comprobar el gestionar el acceso a las distintas tecnologías. Cuando se inicializa el visor, este se ejecuta en primer lugar para conocer con qué tecnologías se cuenta:

1. En primer lugar se comprueba el acceso al GPS, solicitando permiso al usuario para acceder a sus coordenadas geográficas. Si no se consiguiera permiso, no estuviera disponible, o se superara el tiempo máximo para la operación, se alertaría al usuario de que no es posible utilizar el visor sin geolocalización, y se detendría la carga.
2. A continuación, se intenta acceder al giroscopio. Tras realizar pruebas, se ha comprobado que en algunos dispositivos, aunque si se permita el acceso al giroscopio, no se detectan de forma correcta los cambios de orientación (indicando siempre el valor cero para los tres ángulos). En caso de que ocurriera esto, o no estuviera disponible, se acudiría a la alternativa de los eventos de pulsación en pantalla. Si eso ocurre, no es necesario comprobar el acceso al vídeo (acudiendo directamente al panorama alternativo), ya que no tiene mucho sentido que existan eventos para desplazarse por la escena, con un vídeo que se desplaza de forma independiente (ya que usa la cámara y no depende del giroscopio).
3. Por último, se comprueba el acceso al vídeo. En caso de que no existiera u ocurriera algún tipo de error, se utilizaría la alternativa del panorama que simula la escena de una ciudad.

Estos indicadores permiten luego inicializar el visor de realidad aumentada de una forma u otra, dependiendo de las disponibilidades. Además, funcionan a modo de botones, que se pueden ver en la esquina inferior izquierda de la captura de pantalla [Figura 4.7], permitiendo activar/desactivar las distintas tecnologías (si fuera posible, o alertando al jugador si no lo fuera), y mostrando al usuario aquello que está disponible en cada momento.

Figura 4.7. Captura de pantalla del visor de realidad aumentada.

4.3.5 Modo estereoscópico

Para la implementación del modo estereoscópico se emplea la herramienta StereoEffect de Three.js, tomando como referencia el ejemplo de mismo nombre que ofrece la librería. Lo que permite esta herramienta es dividir la pantalla en dos partes, de tal modo que cada lado de la escena se renderiza con una cámara distinta, desplazadas entre sí unos centímetros en horizontal (como si de los dos ojos humanos se tratara) [Figura 4.8]. Al igual que para las tecnologías disponibles, también se emplea un indicador que actúa como botón, permitiendo su activación y desactivación.

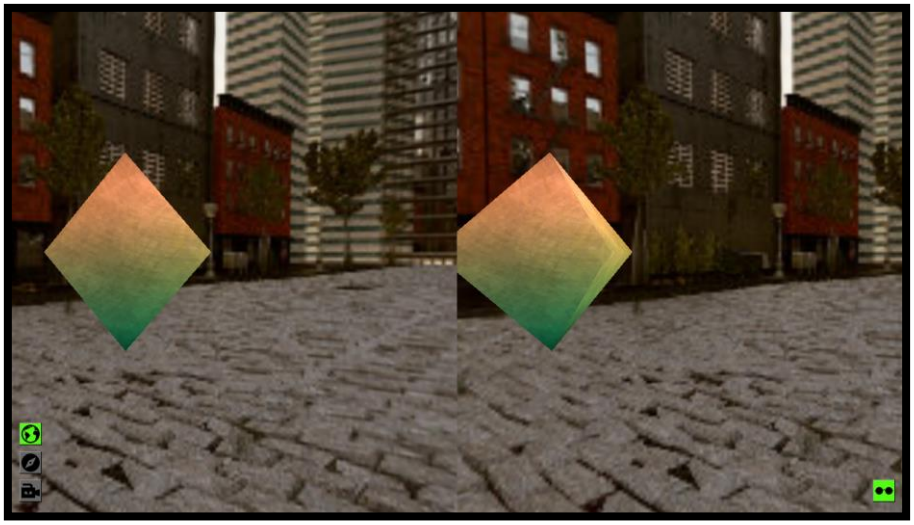

Figura 4.8. Captura de pantalla del visor de realidad aumentada en modo estereoscópico.

Está pensado para ser utilizado con unas gafas de realidad virtual basada en dispositivos móviles, en la que se coloca un Smartphone a modo de lentes, de tal modo que cada ojo sólo ve una sección de la pantalla, creando un efecto 3D mediante una ilusión de profundidad.

Lo anterior significa que la gestión de eventos se debe realizar de otro modo, pues ya no se puede seleccionar el objeto pulsando simplemente sobre una de sus copias (lo que podría implicar tener que quitarse las gafas). En este modo, el 'rayo' se lanza hacia el centro, independientemente de la posición de la pantalla sobre la que se pulse, es decir, recolectando el objeto sólo si el jugador lo ve en el centro.

Sin embargo, existen varios problemas en la utilización de este modo en dispositivos móviles. El primero: el borde central no está bien definido porque la sección derecha se sobrepone sobre la izquierda en varias partes. Se asume que deriva del renderizado en Canvas que se utiliza en dispositivos sin soporte de WebGL; la parte derecha es la última en renderizarse (por lo tanto se coloca encima), y el dibujado se realiza dividiendo los elementos en polígonos, y desde que una parte de un polígono (tanto de un objeto como del panorama) sea visible en la parte derecha, se dibujará completo, incluso si parte del mismo cayera sobre la parte izquierda, superponiéndose.

El otro error es debido a que el vídeo obtenido de la cámara no se puede expandir, por tanto, al duplicarlo, se duplican también las bandas blancas (y se centra automáticamente), y dado que la herramienta StereoEffect dibuja la escena dividiendo a partir del centro de la pantalla, si las bandas blancas están en los laterales, el fondo del espacio central sería un espacio en blanco, lo que complica considerablemente superponer la escena de realidad aumentada sobre el vídeo. Aunque este error no se considera tan grave, dado que, aunque se ha implementado, se recomienda utilizar el modo estereoscópico sin el vídeo, pues hasta que no se extienda la comercialización de los móviles con dos cámaras traseras, el fondo que se verá a un lado y a otro, será el mismo.

Estos errores, provocados por causas externas, de las tecnologías utilizadas, de momento no han podido solucionarse. Las opciones consistirían en buscar alguna alternativa a Three.js o a la etiqueta Video, o esperar simplemente a

que la tecnología avance; pero de momento la utilización del modo estereoscópico en dispositivos móviles no es del todo factible.

4.3.6 Integración en Progrezz

Aunque este apartado no debe ser considerado a la hora de valorar el trabajo realizado (dado que el proyecto se ocupa de la creación del módulo y no de su utilización en aplicaciones concretas), se menciona únicamente para mostrar un ejemplo de uso, y dejar constancia del funcionamiento del visor y de su correcta adaptación a las necesidades de la aplicación de Progrezz.

Siendo breve, comentar que se emplea el visor añadiendo una serie de objetos recolectables cercanos al jugador, que representan fragmentos de mensajes; de tal modo que al ser seleccionados, se envía una petición a un servidor con la información del objeto y del usuario, confirmando su recolección y permitiendo al jugador ganar experiencia.

4.4 Documentación y testeo

Respecto al tema de documentación, únicamente se cuenta con una guía de uso e instalación, con información básica acerca del proyecto [A.2. Fichero README.md], incluyendo dependencias y nombrando a los autores de los recursos empleados (iconos y texturas). No obstante, se ha puesto especial cuidado en crear una estructura del proyecto clara y simple, y en generar código autodocumentado, de fácil entendimiento, al que se le han añadido comentarios detallados para indicar la finalidad que posee cada segmento de los scripts.

En cuanto a las pruebas, dado el carácter delicado de muchos de los recursos utilizados, ha sido necesario probar el módulo en distintos navegadores de escritorio y de dispositivos móviles. En principio, las funcionalidades implementadas, a excepción de los errores mencionados que pueden tener lugar, tienen el comportamiento esperado en los navegadores más utilizados: Firefox y Chrome, tanto en las versiones de escritorio como en dispositivos móviles.

Capítulo 5.
Conclusiones y líneas futuras

Las expectativas futuras acerca del presente proyecto giran en torno a dos líneas principales:

- Como el trabajo se realizó asumiendo que el módulo se integraría en el proyecto Progrezz, cuya máxima es la del crecimiento constante, el visor de realidad aumentada sufrirá modificaciones, ampliaciones y correcciones a lo largo del tiempo. Algunas ya mencionadas, como hacer crecer la lista de tipos de objetos que se pueden visualizar, o plantear alternativas funcionales para la visión estereoscópica. Y por supuesto, se espera compartir autoría con cualquier persona que decida sumarse a la causa.
- El segundo aspecto es el de la creación de documentación más detallada acerca del funcionamiento y la implementación de los scripts que constituyen el módulo; en principio, acudiendo a la posibilidad de creación de una wiki[43] que ofrece GitHub. Este planteamiento deriva del anterior, pues si se tiene como objetivo que la comunidad aporte a un proyecto, es casi imprescindible que cuente con documentación clara y suficiente que permita ilustrar acerca del funcionamiento del mismo sin necesidad de acudir al código.

No puede concluir el presente documento sin que se mencione lo interesante que se consideran a juicio personal los temas tratados. Por una parte, el trabajo realizado ha dado la posibilidad de sumergirse en los entramados de la tecnología web y la realidad aumentada, dirigida hacia dispositivos móviles. Son dos campos de aparición relativamente reciente aún por explorar, cuya propensión al cambio derivada de su creciente auge y el desarrollo constante al que se encuentran sometidos, podría incluso hacerlos pecar de inestabilidad.

[43] Sitio web cuyas páginas pueden ser editadas desde el navegador, creando, modificando o eliminando contenidos.

De ahí que sea importante la aparición de proyectos que traten sobre ellos, sin más objetivo que el de investigar cuáles son las limitaciones y posibilidades que ofrecen; de la mejor manera posible: poniéndolas a prueba.

No obstante, el aspecto que lo hace más interesante es sin duda que se ha creado con el claro objetivo de que sea utilizado en Progrezz, un proyecto de repercusión real. Una oportunidad de múltiples caras, ofreciendo la posibilidad de observar la utilidad del esfuerzo personal, demostrar las capacidades adquiridas a lo largo de años aprendizaje, y al mismo tiempo ayudando a una causa de abnegación incuestionable.

Se confía fervientemente en que el presente Trabajo de Fin de Grado haya logrado arrojar algo de luz sobre el tema que ocupa, y sobre todo, lejos de desear reconocimiento por haber realizado una aportación al proyecto Progrezz, tener la oportunidad de participar creando una herramienta para el mismo que permitirá apreciar la utilidad real del trabajo realizado, o haber favorecido de algún modo a que llegue a apoyar a la más noble de las causas: ayudar a las personas, son sin duda razones por las que estar agradecido.

Capítulo 6.
Summary and conclusions

6.1 Summary

Serious games are a type of games that not only aim the player's fun, but can be used for many different goals, such as education or health.

The main objective of this work was to create a geolocated augmented reality module, whit aims to integrate into the Progrezz project, a serious game that intends to improve the social environment, and approaches the attraction of games and the possibilities of mobile devices, to work as a platform for diffusion of social actions and volunteering. It's a free and open source project, and, to be playable, it needed a way to visualize and collect geolocated objects on any device with access to geolocation.

It was decided to create a viewer that could perform that functionality. It was previously necessary to make an analysis phase, in which it was decided to create a web application, studysing the access to different technologies, and making decisions: getUserMedia for the video camera, Geolocation API for the GPS, DeviceOrientationEvent for the gyroscope, and the library Three.js for drawing 3D graphics. Always trying to achieve that the module was available for more people as possible, providing alternatives to the used resources, because in many cases the technology is experimental, and its correct performance is still not guaranteed.

The objectives have been fulfilling. The viewfinder allows viewing and interacting with objects, according to geographical coordinates. The position of these objects will depend on the actual position of the player, and how the device orientation is. Indicators are also provided to inform the user what technologies are available, allowing to toggle them; and other features such as stereoscopic vision, designed to be used with virtual reality goggles based on mobile devices.

6.2 Conclusions

This document cannot conclude without mentioning how interesting the topics are, as a personal opinion. Doing these tasks has allowed to dive into the intricacies of web technology and augmented reality, focused on mobile devices. Two relatively recent fields, unexplored and emerging, whose propensity to change derived from its growing importance and constant development to which they are subject could make them unstable. For this reason, it's important to spend time to investigate what the limitations and possibilities exist, in the best manner: by testing them.

However, the aspect that makes it more interesting, is undoubtedly that has been created with the clear objective to be incorporated Progrezz, a real impact project. An opportunity multifaceted, offering the possibility to observe the usefulness of personal effort, demonstrating the skill acquired through years of learning, while helping a cause of unquestionable abnegation.

It is hoped that the work done have shed some light on the topic. No acknowledgment is expected for the contribution made to Progrezz. Instead, have the opportunity to participate, developing a tool that will show the real values of the work done, or have favored a project that supports the noblest cause: helping people; are certainly reasons to be truly grateful.

Capítulo 7. Presupuesto

La finalidad de esta sección es la de presentar una estimación acerca del coste económico que supondría la realización del presente proyecto, en caso de que hubiera sido llevado a cabo como actividad remunerada por un graduado en ingeniería informática.

7.1 Coste del proyecto

La realización del trabajo se ha dividido en una serie de fases bien diferencias, que se han completado en conjunto, en un plazo de tres meses. Cada una está asociada al tiempo (en horas) requerido para completarlas y a un coste económico, considerando que la remuneración asciende a 15 €/h.

Fases	Duración (h)	Coste (€)
Análisis	90 h	1440 €
Diseño	30 h	450 €
Implementación	150 h	2250 €
Total	**270 h**	**4140 €**

Tabla 7.1. Estimación de tiempo y coste por fase.

También es importante, dadas las características del proyecto, considerar una serie de materiales necesarios para el desarrollo del mismo, recogidos a continuación.

Materiales	Coste (€)
Smartphone de gama media	219 €
Gafas de Realidad Virtual para Móviles	39 €
Total	**258 €**

Tabla 7.2. Estimación del coste de materiales.

Apéndice A.
Enlaces del proyecto

A.1. Repositorio de GitHub

Como se mencionó en el cuerpo del trabajo [4.1], se hizo uso de un repositorio de GitHub para realizar el control de versiones, gestionar y facilitar el desarrollo, y, dado que se trata de software libre de código abierto, para ofrecer la posibilidad de que cualquiera pueda adquirir y/o colaborar en el módulo:

https://github.com/teamprogrezz/progrezz-ar

A.2. Fichero README.md

Se recomienda encarecidamente leer el documento 'README.md' (al menos los apartados relativos a su utilización, instalación, y dependencias), en caso de que se precise su utilización. En él también se especifica la autoría de los iconos e imágenes utilizadas en el proyecto:

https://github.com/teamprogrezz/progrezz-ar/blob/master/README.md

A.3. Licencia MIT

Acceso a la descripción de la Licencia MIT incluida en el proyecto:

https://github.com/teamprogrezz/progrezz-ar/blob/master/LICENSE

Apéndice B. Fórmulas

B.1. Cálculo de distancias entre dos puntos (latitud, longitud).

Notación:

- φ: latitud.
- λ: longitud.
- R: radio de la Tierra (6371 km).

Haversine formula:

- $a = \sin^2(\Delta\varphi/2) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2(\Delta\lambda/2)$
- $c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$
- $d = c \cdot R$ (km)

Para el cálculo entre dos latitudes o dos longitudes, se emplea la misma fórmula, pero siendo ambas longitudes o ambas latitudes idénticas, respectivamente.

Bibliografía

[1] **Bidelman, Eric. 2013.** *Capturing Audio & Video in HTML5.* [En línea] 29 de Octubre de 2013. [Citado el: 5 de Junio de 2015.] http://www.html5rocks.com/es/tutorials/getusermedia/intro/.

[2] **Crockford, Douglas.** *The JavaScript Minifier.* [En línea] [Citado el: 1 de Junio de 2015.] http://www.crockford.com/javascript/jsmin.html.

[3] —. *The JavaScript Code Quality Tool.* [En línea] [Citado el: 1 de Junio de 2015.] http://www.jslint.com/.

[4] **Echegaray, Bruno García. 2014.** *Artículo de introducción a PhoneGap.* [En línea] 18 de Octubre de 2014. [Citado el: 3 de Junio de 2015.] http://blog.garciaechegaray.com/2014/10/18/phonegap-workshop.html.

[5] **Gubkin, Alon.** *PhoneRTC Documentation.* [En línea] [Citado el: 5 de Junio de 2015.] https://github.com/alongubkin/phonertc/wiki.

[6] **LePage, Pete. 2011.** *Using Device Orientation.* [En línea] 29 de Abril de 2011. [Citado el: 6 de Junio de 2015.] http://www.html5rocks.com/en/tutorials/device/orientation/.

[7] **Mozilla Developer Network.** *Navigator.getUserMedia API.* [En línea] [Citado el: 4 de Junio de 2015.] https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia.

[8] —. *Using geolocation.* [En línea] [Citado el: 6 de Junio de 2015.] https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/Using_geolocation.

[9] —. *DeviceOrientationEvent.* [En línea] [Citado el: 6 de Junio de 2015.] https://developer.mozilla.org/en-US/docs/Web/API/DeviceOrientationEvent.

[10] —. *OrientationChangeEvent.* [En línea] [Citado el: 7 de Junio de 2015.] https://developer.mozilla.org/en-US/docs/Web/Events/orientationchange.

[11] —. *Using fullscreen mode.* [En línea] [Citado el: 7 de Junio de 2015.] https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Using_full_screen_mode.

[12] —. *Screen.lockOrientation API.* [En línea] [Citado el: 7 de Junio de 2015.] https://developer.mozilla.org/en-US/docs/Web/API/Screen/lockOrientation.

[13] **Penadés, Soledad.** *Object picking.* [En línea] [Citado el: 7 de Junio de 2015.] http://soledadpenades.com/articles/three-js-tutorials/object-picking/.

[14] **PhoneGap.** *PhoneGap Documentation - Camera plugin.* [En línea] [Citado el: 5 de Junio de 2015.] http://docs.phonegap.com/en/edge/cordova_media_media.md.html#Media.

[15] —. *PhoneGap Documentation - Media plugin.* [En línea] [Citado el: 5 de Junio de 2015.] http://docs.phonegap.com/en/edge/cordova_media_media.md.html#Media.

[16] —. *PhoneGap Documentation - Capture plugin.* [En línea] [Citado el: 5 de Junio de 2015.] http://docs.phonegap.com/en/edge/cordova_media_capture_capture.md.html#Capture.

[17] —. *PhoneGap Documentation - Geolocation plugin.* [En línea] [Citado el: 6 de Junio de 2015.] http://docs.phonegap.com/en/edge/cordova_geolocation_geolocation.md.html.

[18] **Roast, Kevin.** *Librería phoria.js.* [En línea] [Citado el: 3 de Junio de 2015.] http://www.kevs3d.co.uk/dev/phoria/.

[19] —. *Ejemplo de eventos sobre objetos en phoria.js.* [En línea] [Citado el: 3 de Junio de 2015.] http://www.kevs3d.co.uk/dev/phoria/test0p.html.

[20] **Three.js.** *Examples - Canvas Interactive Particles.* [En línea] [Citado el: 7 de Junio de 2015.] http://threejs.org/examples/#canvas_interactive_particles.

[21] —. *Examples - Device Orientation Controls.* [En línea] [Citado el: 7 de Junio de 2015.] http://threejs.org/examples/#misc_controls_deviceorientation.

[22] —. *Examples - Equirectangular Panorama.* [En línea] [Citado el: 7 de Junio de 2015.] http://threejs.org/examples/#webgl_panorama_equirectangular.

[23] **Tibbett, Rich. 2014.** *Practical Application and Usage of the W3C Device Orientation API.* [En línea] 26 de Marzo de 2014. [Citado el: 7 de Junio de 2015.] https://dev.opera.com/articles/w3c-device-orientation-usage/.

[24] **W3C.** *DeviceOrientation Event Specification.* [En línea] [Citado el: 6 de Junio de 2015.] http://w3c.github.io/deviceorientation/spec-source-orientation.html.

[25] **W3Schools.** *HTML5 Geolocation.* [En línea] [Citado el: 6 de Junio de 2015.] http://www.w3schools.com/html/html5_geolocation.asp.

[26] **Wikipedia.** *Juego Serio.* [En línea] [Citado el: 27 de Mayo de 2015.] http://es.wikipedia.org/wiki/Juego_serio.

[27] —. *Ángulos de Euler.* [En línea] [Citado el: 7 de Junio de 2015.] http://es.wikipedia.org/wiki/%C3%81ngulos_de_Euler.