

Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado

Telemetría y telecontrol mediante realidad virtual en robots de
exploración marciana

ESCUELA POLITECNICA

Autor: Daniel Estanguí Román

SUPERIOR

Tutor: David Fernández Barrero

Cotutor: Pablo Muñoz Martínez

2018

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

Telemetría y telecontrol mediante realidad virtual en robots
de exploración marciana

Autor: Daniel Estangüi Román

Tutor: David Fernández Barrero

Cotutor: Pablo Muñoz Martínez

TRIBUNAL

Presidente: María Dolores Rodríguez Moreno

Vocal 1º: Juan Ignacio García Tejedor

Vocal 2º: Concepción Batanero Ochaíta

FECHA: Fecha de depósito

*Dedicado a
mi abuelo Hilario.*

AGRADECIMIENTOS

Este trabajo supone el punto final de mi etapa como estudiante de ingeniería informática en la Universidad de Alcalá, por ello me gustaría agradecer en primer lugar a David Fernández y a Pablo Muñoz por su compromiso y su ayuda para realizar este proyecto. Agradecer también al resto de profesores que durante estos cuatro años, no solo se han dedicado a transmitir sus conocimientos teóricos, sino que también han sido capaces de motivar a sus alumnos, entre los cuales me encuentro, a investigar y esforzarse por aprender más allá de las clases que impartían.

A todos los compañeros que he tenido en la universidad, los que empezaron y han terminado la carrera conmigo, los que la dejaron antes de terminarla, y los que aun siguen en ella, a los cuales le deseo que terminen pronto.

A mis compañeros del trabajo (Fer, Lalo, Lucía, Denys, Laura, Juanjo, Shibei, Alberto, David) por todo lo que me han enseñado en tan poco tiempo y por la gratificante experiencia que ha sido trabajar con ellos. En especial, me gustaría agradecer a mi jefe de equipo por permitirme utilizar el equipo para realizar mi proyecto.

A mis amigos, la PJ, porque en cierta medida ellos también tienen la culpa de mis logros.

Agradecer a mis padres y mi hermano Juanjo todo su apoyo y consejo, por educarme como lo han hecho y porque gracias a ellos he tenido todo lo que he necesitado.

RESUMEN Y PALABRAS CLAVE

Resumen en castellano

El propósito principal de este Trabajo Fin de Grado es realizar una aplicación que permita simular las operaciones de los rovers en la superficie de Marte mediante la utilización de tecnologías de realidad virtual. Para ello, se ha realizado una adaptación de una aplicación que emplea un algoritmo de planificación de rutas, llamado *3D Accurate Navigation Algorithm*, para poder visualizar los resultados en un entorno virtual. Desde este entorno inmersivo se podrá realizar un seguimiento de la misión asignada a un vehículo de exploración espacial y a través de esta simulación determinar su posible éxito.

Resumen en inglés

The main purpose of this Bachelor's Thesis is to develop an immersive application that allows to simulate rover' surface operations in Mars through the use of virtual reality technologies. In order to achieve data visualization in virtual environments, an adaptation to the 3D Accurate Navigation Algorithm application, which uses path planning algorithms, has been made. Finally, from this immersive environment the user can track the mission assigned to a space exploration vehicle, and through this simulation can determine its possible success.

Palabras clave: Rover, Realidad virtual, Realidad extendida, Computación gráfica, Scrum

RESUMEN

El sector de la realidad virtual está creciendo de forma flagrante debido a los avances acontecidos en estos últimos años, permitiendo reducir los costes en la producción de nuevos dispositivos y contenidos, al mismo tiempo que las sensaciones que genera esta nueva tecnología está mejorando. Los estudios estadísticos reflejan que la realidad virtual abarcará un segmento del mercado bastante amplio en los próximos años. Por ello, las principales compañías tecnológicas dedicadas a la creación de *hardware* y *software* están invirtiendo una gran cantidad de recursos para hacer de la realidad virtual una nueva revolución tecnológica.

La realidad virtual no es una tecnología reciente, tiene su origen en el año 1968. En este año Ivan Sutherland, uno de los científicos más influyentes en el ámbito de computación gráfica, construye el primer dispositivo *Head-Mounted Display* de realidad virtual. El principal inconveniente de este prototipo era que necesitaba de un sistema computacional muy aparatoso y de alto coste, que además no garantizaba una experiencia inmersiva suficientemente buena. Aunque esta primera toma de contacto con la realidad virtual sirvió de punto de partida para muchos investigadores que desarrollaron sus propios dispositivos e incluso llegaron a sacarlos al mercado. Sus intentos se vieron frustrados por la falta de tecnología de la época o por su alto coste. No fue hasta el año 2012, en el que un joven emprendedor llamado Palmer Luckey desarrolla Oculus Rift. El prototipo de gafas de realidad virtual de Luckey podía realizarse por un precio más económico garantizando un campo de visión igual o superior al de los dispositivos del momento. Este hecho supuso el renacimiento de la realidad virtual, hizo que muchas empresas empezaran una carrera por desarrollar y comercializar sus propios dispositivos dentro del sector.

Las posibles aplicaciones de la realidad virtual aun están por definir, lo que es seguro es que se puede aplicar a un gran número de sectores como las ciencias, las ingenierías, las artes, el entretenimiento y la educación entre otros. La investigación también se vería potenciada por el uso de las experiencias inmersivas para realizar simulaciones que puedan ser muy costosas en el entorno físico o aquellas que puedan poner en riesgo la seguridad de las personas.

Se ha diseñado e implementado una aplicación de realidad virtual, perteneciente al ámbito de la investigación espacial, para ello se ha utilizando la metodología ágil, Scrum. Esta aplicación ha sido desarrollada con el motor de juegos Unity, uno de los mejores motores para desarrollar aplicaciones interactivas en 2D y 3D. Además se han empleado herramientas *software* para el diseño de los modelos tridimensionales que se han incorporado a la aplicación, como Blender y Autodesk Maya, o para la edición de *scripts*, como MonoDevelop o Visual Studio. El proyecto realizado está disponible para las plataformas de realidad virtual de Oculus, Samsung GearVR y Google Cardboard. La aplicación permite simular las misiones realizadas por un vehículo de exploración espacial no tripulado sobre la superficie de Marte en un entorno inmersivo. Esta aplicación tiene como objetivo ayudar a los investigadores a determinar cuáles son las mejores rutas para la realización de las misiones. Los usuarios pueden visualizar las rutas, calculadas por el algoritmo de optimización de rutas *Accurate Navigation Algorithm*, sobre un terreno real de la superficie de Marte generado a partir de los ficheros DTM generados por el HiRISE, que orbita en torno al planeta rojo.

ÍNDICE GENERAL

RESUMEN Y PALABRAS CLAVE	IV
RESUMEN EXTENDIDO	V
1. INTRODUCCIÓN	1
1.1. Estructura de la memoria	1
1.2. Visión general	2
1.3. Objetivos y campo de aplicación	4
1.4. Enfoque metodológico	5
1.4.1. Metodologías pesadas y metodologías ágiles	5
1.4.2. Metodología Scrum	6
1.4.3. Metodología del proyecto	7
2. ESTADO DEL ARTE	8
2.1. Computación gráfica	8
2.1.1. Historia de la computación gráfica	9
2.2. Continuo de la virtualidad y realidad extendida	15
2.2.1. Realidad aumentada	16
2.2.2. Virtualidad aumentada	18
2.2.3. Realidad virtual	18
2.2.3.1 <i>Head-Mounted Display</i>	19
2.2.3.2 Grados de libertad	20
2.2.3.3 Tracking	21
2.2.3.4 Campo de visión	21
2.2.3.5 Frecuencia de refresco	22
2.2.3.6 Accesorios	23

2.3. Motores gráficos	25
2.3.1. Arquitectura	25
2.3.1.1 Hardware, drivers y sistema operativo	26
2.3.1.2 Librerías	27
2.3.1.3 Capa independiente de la plataforma	27
2.3.1.4 Subsistemas principales	27
2.3.1.5 Gestor de recursos	28
2.3.1.6 Networking.	28
2.3.1.7 Subsistema del juego	28
2.3.1.8 Audio	28
2.3.1.9 Motor de <i>rendering</i>	29
2.3.1.10 Herramientas de desarrollo	31
2.3.1.11 Motor de física	31
2.3.1.12 Interfaces de usuario	32
2.3.1.13 Sistemas específicos de juego	32
2.3.2. Unity	32
3. ANÁLISIS DEL PROBLEMA	34
3.1. Descripción del problema	34
3.2. Historias de usuario	35
3.3. Aplicaciones similares	36
3.3.1. OnSight	36
3.3.2. Access Mars	37
3.3.3. Mars Odyssey	37
3.3.4. Experience Curiosity.	38
3.4. Plataformas utilizadas	39

4. DISEÑO DE LA SOLUCIÓN	40
4.1. Backlog	40
4.1.1. Sprint planning	40
4.2. Organización del equipo.	48
5. IMPLEMENTACIÓN	50
5.1. Etapas de desarrollo	50
5.2. Diseño de escenas	56
5.2.1. Menú aplicación de escritorio.	56
5.2.2. Menú en realidad virtual.	57
5.2.3. Efectos visuales del menú	57
5.2.4. Proceso de incorporación de los modelos en la escena	58
5.2.4.1 Modelo del rover	59
5.2.4.2 Modelos de los mapas	60
5.2.5. Diseño de simulación 3D	61
5.2.6. Diseño de simulación inmersiva	62
5.3. Experiencia de usuario.	64
6. CONCLUSIONES	65
6.1. Objetivos cumplidos	65
6.2. Lineas futuras.	66
6.2.1. Mejoras de rendimiento	66
6.2.2. Mejoras de realismo	67
6.3. Conclusiones del proyecto.	67
6.3.1. Burn down chart	67
6.3.2. Conclusiones personales.	68
BIBLIOGRAFÍA	70

ÍNDICE DE FIGURAS

1.1	<i>Predicción de tamaño de mercado de AR y VR en billones de dólares.</i>	3
1.2	<i>Ciclo de desarrollo de Scrum.</i>	6
2.1	<i>Sword of Damocles.</i>	10
2.2	<i>Captura de pantalla del videojuego Doom, 1993.</i>	12
2.3	<i>Campaña de crowdfunding de Oculus Rift en la plataforma KickStarter.</i>	14
2.4	<i>Representación del continuo de realidad</i>	15
2.5	<i>Prototipo de Magic Leap en 2016.</i>	17
2.6	<i>Magic Leap One HMD.</i>	19
2.7	<i>Sistema CAVE.</i>	20
2.8	<i>Campo de visión de StarVR.</i>	22
2.9	<i>Accesorio Leap Motion para detección de manos.</i>	24
2.10	<i>Estructura de capas que forma un motor de juego.</i>	26
2.11	<i>Visión conceptual de la estructura de un motor de rendering [12].</i>	30
2.12	<i>Plataformas para las cuales se puede desarrollar con Unity.</i>	33
3.1	<i>Captura de Onsight.</i>	36
3.2	<i>Captura de Acces Mars.</i>	37
3.3	<i>Captura de Mars Odyssey.</i>	38
3.4	<i>Captura de Experience Curiosity.</i>	38
5.1	<i>Captura del taskboard del spring 5.</i>	50
5.2	<i>Captura del menú de la aplicación de escritorio.</i>	56
5.3	<i>Captura del menú de la aplicación de Samsung GearVR.</i>	57

5.4	<i>Captura del editor de Unity en el proceso de incorporación de efectos de partículas para el modelo de Marte.</i>	58
5.5	<i>Captura del editor de Unity en el proceso de incorporación de físicas del rover.</i>	59
5.6	<i>Captura de la aplicación Autodesk Maya.</i>	61
5.7	<i>Captura de la escena de simulación 3D en la aplicación de escritorio.</i>	62
5.8	<i>Captura de la escena de simulación de realidad virtual en la aplicación de escritorio utilizando Oculus.</i>	63
5.9	<i>Captura de la escena de simulación en la aplicación de Samsung GearVR.</i>	63
6.1	<i>Gráfica de avance o burn down chart del proyecto.</i>	68

1. INTRODUCCIÓN

Las aplicaciones de Realidad Virtual (VR)¹ o Realidad Aumentada (AR)² están empezando a ser cada vez mas usadas en diferentes procesos industriales, sobretodo en aquellos procesos con un alto coste o aquellos que son lo suficientemente arriesgados para las personas que los realizan. En este sentido, la NASA ha hecho uso de tecnologías de VR para simular operaciones de exploración en la superficie de Marte, permitiendo una perspectiva inmersa dentro del entorno de los robots de exploración. También supone la posibilidad de reconstruir las operaciones que se realizan en un entorno de realidad virtual utilizando información de telemetría proporcionada por un rover³.

La realización de operaciones en la superficie de Marte requiere de una planificación previa para garantizar el éxito de la misión. Dentro de esta planificación tienen un peso importante los algoritmos que permiten obtener una ruta óptima entre los distintos puntos de interés. Para ello se realiza un estudio minucioso del terreno, así como de las condiciones que afectan directamente al rover. La simulación en VR permitirá visualizar los factores que condicionan el éxito de la misión y evaluar los posibles riesgos.

1.1. Estructura de la memoria

En este apartado se presenta la distribución del contenido del trabajo en los distintos capítulos que lo forman.

1. Introducción: Este capítulo es una primera toma de contacto con el proyecto, se muestra el contexto de las tecnologías utilizadas en la actualidad y se introducen los motivos que dotan a este trabajo de valor añadido propio. Se presenta brevemente la metodología empleada en la realización del proyecto y los motivos por los que ha utilizado una metodología ágil en vez de una metodología tradicional.

¹La nomenclatura VR, siglas de la palabra en inglés *Virtual Reality*, hacen referencia a la Realidad Virtual en toda la memoria.

²La nomenclatura AR, siglas de la palabra en inglés *Augmented Reality*, hacen referencia a la Realidad Realidad Aumentada en toda la memoria.

³Un rover es un vehículo de exploración espacial.

2. **Estado del arte:** En el capítulo del estado del arte se definen los conceptos necesarios para entender el proyecto, se explican los antecedentes y los avances que hacen posible a día de hoy la realización de este proyecto. Se presentan las principales tecnologías disponibles en el mercado y las tecnologías que han sido utilizadas.
3. **Análisis del problema:** En este capítulo se analiza el problema que aborda el proyecto, se define un primer enfoque de las funcionalidades de la aplicación y se muestran los resultados del estudio realizado a otras aplicaciones similares.
4. **Diseño de la solución:** Este capítulo explica el proceso para la realización del diseño de la aplicación y la documentación que se ha empleado para su desarrollo.
5. **Implementación:** En el capítulo de implementación se explican los pasos realizados para el desarrollo de la aplicación.
6. **Conclusiones:** En este capítulo se realiza una reflexión sobre los objetivos marcados en el inicio y el resultado obtenido. También se explican las posibles mejoras del proyecto y el futuro del mismo.

1.2. Visión general

Las tecnologías de realidad extendida (XR)⁴, entre las que se encuentra la realidad virtual y la realidad aumentada, representan un sector del mercado en alza. Este crecimiento supone una oportunidad de inversión para todas las empresas que apuesten por tecnologías innovadoras. Grandes empresas como Google, Sony, HTC, Samsung, Facebook, Apple o Microsoft, han empezado a invertir grandes cantidades de dinero en el desarrollo de dispositivos *hardware* y *software* de estas características. A pesar de que han sido muchas las ocasiones en las que la realidad virtual ha intentado llegar a comercializarse, nunca ha llegado a tener el impacto deseado. Los avances tecnológicos de estos últimos años dan motivos suficientes como para ser optimistas y creer que durante los próximos años la XR supondrá una nueva revolución tecnológica a nivel mundial.

El portal de estadísticas Statista⁵ estima que el tamaño de mercado de realidad au-

⁴La nomenclatura XR, siglas de la palabra en inglés *Extended Reality*, hacen referencia a la Realidad Extendida en toda la memoria.

⁵Statista es uno proveedores líderes de datos de mercado e información sobre los consumidores.

mentada y realidad virtual para 2022 alcance la cifra de 209.200 millones de dólares. A continuación se muestran los resultados del estudio estadístico realizado por Statista para los años 2016, 2017, 2018 y 2022 en todo el mundo [1].

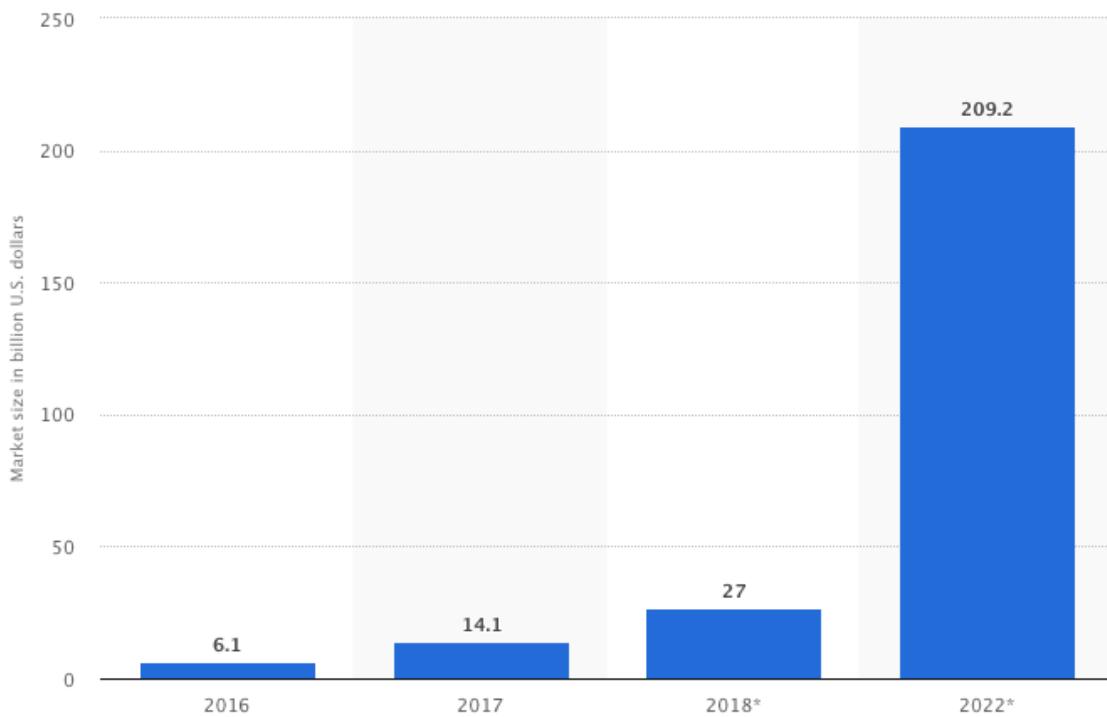


Figura 1.1: *Predicción de tamaño de mercado de AR y VR en billones de dólares.*

Los dispositivos y los contenidos de tecnologías inmersivas son cada vez menos costosos de realizar, por lo que cuentan con precios cada vez más competitivos. Muchos de los teléfonos inteligentes que están disponibles en el mercado son compatibles con estas tecnologías. Por este motivo, se espera que para el año 2018 el número *smartphones* alcance la cifra de 3 millones y que este hecho sirva para potenciar la producción de aplicaciones de VR y AR.

Los estudios estadísticos realizados en la plataforma Orbis Research⁶ estiman que la realidad virtual está a punto de convertirse en la corriente principal y que podrá superar los 40 mil millones de dólares en el mercado para el año 2020. Uno de los puntos claves para desarrollar esta estadística fue que en el año 2016 los dispositivos *hardware* de realidad virtual representaron la máxima participación en el mercado [2].

⁶Orbis Research es una de las bases de datos masiva de investigación de mercado que cuenta con el respaldo de un gran número de investigadores que buscan información actualizada de las últimas tendencias del mercado.

Uno de los factores por los que se espera que las tecnologías de realidad virtual empiecen a crecer de esta manera tan rápida es por la variedad aplicaciones que puede llegar a tener. El sector del entretenimiento ha sido de los primeros en apostar por las tecnologías inmersivas para mejorar las experiencias de los usuarios, pero su aplicación no se limita únicamente a este sector. Prácticamente todas las tareas que se realizan en sectores como el turismo, la educación, la medicina, la arquitectura o la ingeniería se podrán ver beneficiadas por estas tecnologías.

1.3. Objetivos y campo de aplicación

Este trabajo tiene como propósito principal realizar una aplicación que permita simular las operaciones de los rovers en la superficie de Marte. Un aspecto importante de la planificación de las misiones son las rutas que el rover tiene que seguir para alcanzar los puntos de interés científico. Para la planificación de rutas se emplea una combinación de imágenes obtenidas desde la órbita y desde la superficie, que sirven para analizar la topografía y generar un mapa de costes. Aplicando los algoritmos pertinentes a estos mapas de costes es posible obtener soluciones óptimas para dirigir el rover a través de los distintos puntos. El cálculo de rutas se realiza con un algoritmo de planificación llamado *3D Accurate Navigation Algorithm* (3Dana) [3]. Este algoritmo utiliza mapas de costes y/o *Digital Terrains Models* (DTM) obtenidos del *High Resolution Imaging Science Experiment* (HiRISE)⁷.

Se pretende desarrollar una aplicación de bajo presupuesto que proporcione una vista tridimensional en un entorno de realidad virtual que permita explorar la superficie de Marte con la intención de ayudar a los investigadores a decidir qué rutas seguir. Para conseguir tal propósito, serán necesarios una serie de objetivos específicos:

1. Realizar una adaptación de la aplicación encargada de la planificación de rutas a la nueva aplicación encargada de la simulación en realidad virtual.
2. Transformar los mapas en formato DTM en un modelo tridimensional.

⁷HiRISE (*High Resolution Imaging Science Experiment*) es una cámara que orbita entorno al planeta Marte a bordo del satélite *Mars Reconnaissance Orbiter*, lanzado en 2005 [4]

3. Crear un modelo de vehículo de exploración espacial capaz de simular las operaciones que realiza el rover en la superficie de Marte.
4. Diseñar una solución *software* que permita la telemetría, y que esta sea representada dentro del entorno virtual.
5. Proporcionar un entorno amigable con el usuario, desde el cual pueda hacer uso de todas las funcionalidades de la aplicación.
6. Modificar los componentes proporcionados por Google VR SDK para ajustarlos a las necesidades de la aplicación.

1.4. Enfoque metodológico

La metodología se encarga de indicar los métodos y técnicas que hay que utilizar en cada fase del ciclo de vida del proyecto. Es uno de los elementos más relevantes, la diferencia entre escoger una metodología u otra puede definir el éxito o el fracaso. A la hora de escoger una metodología para desarrollar un producto *software* podemos distinguir entre las metodologías tradicionales, también conocidas como metodologías pesadas, y las metodologías ágiles.

1.4.1. Metodologías pesadas y metodologías ágiles

Las metodologías tradicionales se basan en una gestión predictiva, es decir, parten de unos requisitos iniciales. A partir de estos requisitos se realiza un plan utilizando los recursos y el tiempo necesarios. Esta metodología define un conjunto de fases secuenciales, los procesos que van a realizarse en estas fases y cual va a ser su coste.

Las metodologías ágiles surgen como una alternativa a las metodologías tradicionales, las cuales son demasiado burocráticas y rígidas para las actuales características del mercado. El entorno del desarrollo de *software* actual está sujeto a un continuo cambio, por lo que las metodologías tradicionales dejan de tener sentido debido a su falta de adaptación. Las metodologías ágiles no solo se adaptan mejor al entorno, sino que también han sabido paliar algunos de los problemas de las metodologías convencionales como: los excesivos costes que se producen en el cambio de los requisitos funcionales, las modificaciones o la

corrección de errores. Además mejoran la comunicación con el cliente, que prácticamente está incluido dentro del equipo de desarrollo, de esta forma se garantiza que el producto final cumpla con las expectativas y que la calidad de este sea la mejor posible [5].

1.4.2. Metodología Scrum

Este modelo fue definido por Hirotaka Tekeuchi y Ikujiro Nonaka en un artículo en 1986. En este artículo se centraron en analizar las empresas tecnológicas que realizaban productos en menor tiempo y coste, con la mejor calidad. Se comparó la forma de trabajar de estas empresas con la formación melé de los jugadores de Rugby que en inglés se conoce como *scrum*. En 1996, Jeff Sutherland y Ken Schwaber establecieron las reglas para el desarrollo *software* ágil basado en los principios de Scrum.

Scrum es una de las mejores prácticas para el desarrollo ágil de proyectos. Para entender el proceso de desarrollo de esta metodología hay que entender dos de los componentes principales que la forman: las reuniones y los roles.

En las reuniones se define la trayectoria y el seguimiento del proyecto, está formado por tres reuniones. En la primera reunión, conocida como *Sprint Planning* se define el documento que refleja los requisitos del sistema o *backlog*, y se planifican también las tareas que se van a realizar el en *sprint* 0. La segunda reunión que se realiza se utiliza de manera iterativa para hacer un seguimiento del *sprint*. La última reunión se realiza cada vez que termina un *sprint* para así poder revisar los resultados obtenidos, ver los fallos para plantear mejor el resto de *sprints* y presentar la versión de prueba al cliente.



Figura 1.2: Ciclo de desarrollo de Scrum.

De manera superficial se van explicar las dos categorías de roles que intervienen en el proyecto. Una de las categorías está formada por las personas que están comprometidas con el proyecto, como el equipo de desarrollo o el Scrum Master, y la otra categoría por las personas que no forman parte del proceso Scrum pero son necesarias para la retroalimentación del proyecto, como los usuarios o el cliente. Más adelante, en la memoria, se explican todos los roles y la forma en la que estos participan en el proyecto.

1.4.3. Metodología del proyecto

Para el desarrollo del proyecto se optado por una metodología ágil como Scrum por los siguientes motivos:

- No es necesario conocer desde el inicio la forma final de la aplicación.
- Gran capacidad de respuesta ante los cambios, los cuales no se entienden como un problema sino como algo necesario para que el producto sea mejor.
- Se realizan pequeñas entregas previas a la entrega final para dar a conocer al cliente el estado del desarrollo del proyecto.
- Favorece la comunicación entre el cliente y los desarrolladores para evitar errores y documentación innecesaria.
- Tiene una gran facilidad para añadir nuevas funcionalidades y mejorar su calidad.

2. ESTADO DEL ARTE

2.1. Computación gráfica

La computación gráfica es un campo de la informática que emplea elementos visuales, generados mediante ordenador, para transmitir información. Crea un medio de comunicación que aprovecha las cualidades de reconocimiento de patrones 2D y 3D de los seres humanos para realizar de manera intuitiva el intercambio de información entre personas y ordenadores. Los gráficos generados mediante ordenador son una herramienta que se emplea diariamente en disciplinas muy diversas como pueden ser: las ciencias, las artes, la ingeniería, los negocios, la industria, la medicina, el entretenimiento, la educación o publicidad [6]. Esto se debe a que prácticamente no hay ninguna tarea en la que representación gráfica no pueda aportar ninguna ventaja.

Uno de los mayores usos de la computación gráfica se encuentra en los procesos de diseño, principalmente en arquitectura e ingeniería, donde los productos se diseñan mediante un ordenador, esta técnica se conoce como CAD⁸. El proceso de diseño se complementa con la manufacturación del producto, donde también la computación gráfica toma parte mediante la fabricación asistida por ordenador o CAM⁹, que hace de puente entre el diseño CAD y las máquinas de producción, con una intervención mínima del operario.

Otra aplicación de los gráficos generados por ordenador es la visualización de datos. Entre los primeros avances por representar datos de manera visual se encuentra la representación de gráficas y diagramas, a día de hoy se pueden generar de manera sencilla mediante una gran variedad de herramientas *software*, por lo que hablar de gráficas y diagramas en visualización de datos sería quedarse corto. El reto de la visualización de datos se encuentra en la representación de una gran cantidad de información de carácter complejo, haciendo que los investigadores y analistas que trabajan con ellos puedan estudiar su comportamiento. Para hacer más intuitiva estas representaciones suelen realizarse gráficas tridimensionales que codifican los valores de los datos utilizando tonos de grises o colores para generar un código cromático.

⁸CAD son las siglas de *Computer Aided Design*, diseño asistido por computadora.

⁹CAM son las siglas de *Computer Aided Manufacturing*.

La computación gráfica influye también en la tarea de modificación o interpretación de imágenes, conocida como procesamiento de imágenes. Aunque parezca que no tiene nada que ver, las técnicas que se emplean para la modificación e interpretación de imágenes son las mismas a las que se emplean en la creación de gráficos de manera computacional. Desde los pequeños retoques de luz y color que se aplican a las fotografías digitales hasta la corrección de color que se aplica en las radiografías con rayos X entrarían dentro de este campo. En el ámbito de la medicina, el uso de estas técnicas está muy generalizado, además de utilizarse en radiografías, pueden utilizarse en múltiples pruebas previas a intervenciones quirúrgicas como, por ejemplo una prueba TAC¹⁰.

Estos son solo unos ejemplos de las posibles aplicaciones que tiene la computación gráfica y la manera en que esta participa en la vida cotidiana de las personas. Donde sin ser conscientes está presente en la mayoría de las tareas que realizan o en los elementos que les rodean.

2.1.1. Historia de la computación gráfica

La historia de la computación gráfica comienza en 1950, cuando el matemático Ben Laposky crea las primeras imágenes gráficas empleando un osciloscopio analógico con finalidades artísticas. Por este experimento, Laposky, es considerado el creador de los primeros gráficos de ordenador y un pionero en el arte electrónico. Un año más tarde, en 1951, se terminó de construir *Whirlwind*. Esta máquina fue creada por el Instituto Tecnológico de Massachusetts (MIT), fue el primer ordenador con pantalla capaz de mostrar datos en tiempo real. El proyecto fue diseñado para el SAGE¹¹ con la intención de poder simular vuelos de entrenamiento para su flota aérea [7].

En 1953, aparece el primer juego para ordenador, llamado OXO, una versión del juego de las tres en raya desarrollado por la Universidad de Cambridge para el ordenador EDSAC. Existe cierta discrepancia en considerar OXO como el primer videojuego para ordenador debido a la falta de animaciones. Por lo que no es hasta 1962 cuando aparece el primer videojuego interactivo de ordenador con animaciones, llamado *Spacewar!*. Este videojuego, desarrollado por varios estudiantes del MIT, simulaba batallas de naves

¹⁰TAC son las siglas de Tomografía Axial Computerizada, es una tecnología sofisticada de rayos X utilizada en medicina para ayudar a detectar enfermedades o estudiar las condiciones del paciente.

¹¹USA Air force's Semi Automatic Ground Environment.

espaciales utilizando un osciloscopio como pantalla. Por otra parte, *Tennis for two* esta reconocido como el primer videojuego de la historia. Este videojuego fue creado por el físico William Higinbotham, en 1958, para ello utilizó un ordenador analógico conectado a un osciloscopio que cumplía la función de monitor.

En 1963, Ivan Sutherland crea *SketchPad*, el primer programa de dibujo para ordenador. Este programa permitía la interacción directa con objetos gráficos, fue pionero en la interacción persona-ordenador, incluía elementos como menús emergentes y utilizaba un lápiz óptico para dibujar sobre la pantalla. La utilización de estos dispositivos para realizar las interacciones con el ordenador era bastante común, hasta que Douglas Engelbart construye el primer ratón en 1967. El ratón demostró ser mas eficiente y preciso que el lápiz óptico o el *joystick*, por lo que no tardó mucho tiempo en sustituir a este tipo de dispositivos [7].

En 1968, Ivan Sutherland crea *The sword of Damocles*¹², considerada como el primer sistema *Head-Mounted Display* (HMD) de realidad virtual. Este sistema consistía en un brazo articulado colgado del techo, con un sistema de visualización compuesto por dos pequeñas pantallas con un soporte para ser ajustado a la cabeza del usuario.



Figura 2.1: *Sword of Damocles*.

¹²Este nombre hace referencia a la espada de Damocles, personaje de la cultura griega, debido a su forma similar a la de una espada y por el modo en que cuelga sobre el usuario.

Henri Gouraud publica en 1971 su técnica de sombreado para gráficos 3D por ordenador. Esta nueva técnica permite suavizar superficies con una carga computacional menor que las técnicas utilizadas hasta la fecha. En 1974, Edwin Catmull desarrolla su propia técnica de sombreado, *Z-buffer*, que a día de hoy se sigue utilizando en algunos componentes gráficos. Un año más tarde, en 1975, Bui Toun Phong mejora la técnica de sombreado creada por Gouraud, la cual se llamará sombreado de Phong. Esta técnica de sombreado calcula la iluminación para cada píxel, esto supone un coste computacional superior al sombreado de Gouraud, pero el resultado tiene una mayor calidad.

Al mismo tiempo que las técnicas de sombreado fueron evolucionando también lo hicieron los comportamientos físicos simulados por ordenador. En 1979, Turner Whitted publica un artículo donde explica la técnica de *Ray Tracing*. Esta técnica simula el comportamiento físico de la luz dentro de un entorno virtual.

En la década de los 70, la computación gráfica llega al cine y empiezan a realizarse los primeros efectos especiales realizados por ordenador. Comenzaron a utilizarse en algunos planos, sobre todo para la representación de pantallas, en películas como *Star wars* (1977). Años más tarde empezaron a utilizarse planos enteros, a partir de la película *Tron* (1982).

En 1982, John Walker funda la empresa Autodesk y lanza su primera versión de AutoCAD. Esta aplicación al igual que el resto de aplicaciones creadas por esta empresa son la herramienta principal de diseño utilizada por ingenieros, arquitectos, mecánicos o modeladores. A día de hoy Autodesk sigue siendo una de las principales empresas encargadas de crear aplicaciones CAD. Este mismo año Voelcker introduce el concepto de voxel, el equivalente al píxel en 3D. La tecnología voxel se integrará años más tarde en la gran mayoría de aplicaciones software médicas.

Apple presenta *Macintosh*, el primer ordenador personal con interfaz gráfica, en Enero de 1984. Este nuevo producto creó un nuevo estándar para el diseño de ordenadores, fue el primero en incorporar una interfaz gráfica de usuario, una pantalla y un ratón.

En 1986, Steve Jobs es despedido de la empresa Apple y compra el departamento de computación gráfica de Lucasfilms, que hasta ahora se encargaba de realizar efectos especiales para cine, creando la empresa independiente Pixar Animation Studios. Ese mismo año Pixar lanza *Luxo Jr.* el primer corto de animación de la empresa Pixar Animation Studios.

En 1989, aparece Parasolid, un motor o algoritmo de modelado geométrico 3D de sólidos y superficies de forma libre orientado a objetos desarrollado por ShapeData y hoy mantenido por Siemens PLM Software, su especificación es abierta y puede ser usada por programas basados en CAD 3D.

En 1992, Silicon Graphics desarrolla el API gráfico OpenGL, el cual se usa en la mayoría de aplicaciones tridimensionales. Dos años más tarde, Microsoft desarrolla, DirectX, su propia API gráfica para competir con OpenGL.

En 1993, se publica *Doom*, el primer videojuego que implementaba un motor gráfico. En abril de este mismo año se funda Nvidia, una de las compañías de hardware gráfico más importante hasta la fecha. Este año también fue un año importante para la computación gráfica dentro del mundo del cine, gracias al estreno de *Jurassic Park*, que fue la primera película con efectos CGI¹³ de gran presupuesto. Dos años más tarde, Pixar presenta *Toy story*, la primera película de animación completamente CGI. En 1995, Sony también lanzó en Europa y Estados Unidos su primera generación de consolas PlayStation.



Figura 2.2: Captura de pantalla del videojuego *Doom*, 1993.

En 2001, se estrena *Final Fantasy: The Spirits Within*, que fue la primera película de animación por ordenador que creó humanos fotorrealistas. Esta película también fue pionera en la simulación de fluidos de manera virtual. Este mismo año Microsoft lanza su primera generación de videoconsolas Xbox para empezar a competir con las segunda

¹³CGI son las siglas de *Computer-generated imagery*, en castellano: imágenes generadas por ordenador.

generación de consolas de Sony, lanzadas un año antes. Desde este año las nuevas tecnologías en procesadores gráficos permiten la utilización de *shaders*. Un año más tarde, la empresa americana Epic Games publica la segunda versión de su motor de juego Unreal Engine y Microsoft publica la versión 9.0 de DirectX.

En 2004, se publica la segunda versión de OpenGL con la novedad, de que esta versión es compatible con GLSL (*OpenGL Shadign Lenguage*). La utilización de *shaders* no se popularizó hasta que no aparecieron las siguientes generaciones de consolas. En el caso de Microsoft fue la Xbox 360, mientras que Sony lanzó su PlayStation 3, ambas utilizando un procesador de IBM. Nintendo también lanza Wii, esta consola no destaca por ser una consola con buenos acabados gráficos, pero si por las formas de interacción entre la consola y el jugador. El controlador de esta consola es capaz de capturar los movimientos que realiza el jugador e interpretarlos para realizar las interacciones dentro del videojuego.

En 2005, la compañía Unity Technologies lanza Unity, un motor de videojuegos con soporte para diferentes tipos de plataformas. Unity tiene una amplia compatibilidad con diversas aplicaciones de modelado, texturizado o animación, también destaca por la facilidad de este software para añadir herramientas de desarrollo externas. Por estos motivos es muy utilizado por los pequeños desarrolladores, no tanto por los grandes a pesar de su continuo perfeccionamiento del motor, y a día de hoy cuenta con una comunidad muy amplia en internet.

Nvidia publica CUDA en 2007, un lenguaje de programación para sacar el máximo partido a su tarjetas gráficas. Este lenguaje era exclusivo de las tarjetas gráficas de Nvidia por lo que un año más tarde Intel, Apple, AMD, IBM y Nvidia se juntan para crear un estándar libre para el procesamiento en GPU, dando como resultado OpenCL.

En 2010, debido al éxito que obtuvo el lanzamiento de la videoconsola Wii, Sony lanzó su propia versión con un controlador parecido. Aunque fue Microsoft la compañía que inicio la revolución en el mercado de los videojuegos y del reconocimiento de imágenes con su Kinect. Este controlador permite a los usuarios interactuar con la consola sin utilizar el contacto físico, reconoce gestos, comandos de voz, objetos e imágenes. Tres años más tarde, en 2013, tanto Sony como Microsoft lanzan las nuevas generaciones de consolas, PlayStation 4 y Xbox One respectivamente.

En el año 2012 aparece un de los acontecimientos más relevantes de la realidad virtual y por consiguiente de la computación gráfica. El emprendedor Palmer Luckey comienza una campaña de *crowdfunding* para la financiación de las gafas de realidad virtual Oculus Rift. Estas gafas han supuesto el renacimiento de la realidad virtual abriendo las puertas de esta tecnología, tanto para los videojuegos como para el público empresarial. La razón del éxito de este proyecto se encuentra en que Oculus Rift garantiza un campo de visión mucho mayor por un precio inferior al de cualquier dispositivo de realidad virtual del momento. Tanto fue el éxito de esta campaña de financiación que llegó a recaudar cerca de dos millones y medio de dólares cuando su objetivo inicial era recaudar doscientos cincuenta mil dólares.

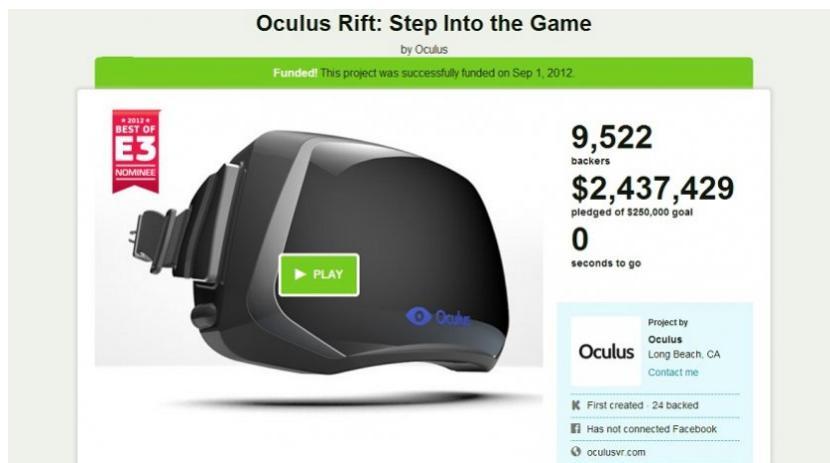


Figura 2.3: *Campaña de crowdfunding de Oculus Rift en la plataforma KickStarter.*

En 2014, Facebook compra Oculus VR por dos mil millones de dólares. Este mismo año, Google comienza Project Tango uno de los primeros desarrollos de realidad aumentada pensados para el público.

En 2016, nace de la mano de HTC y Valve las gafas de realidad virtual HTC Vive. Sony no quiso quedarse atrás en el mercado de la realidad virtual por lo que en octubre de este mismo año presentó PlayStation VR.

En 2017, Apple lanza Arkit, una herramienta para crear aplicaciones de realidad aumentada en dispositivos Apple. Este lanzamiento tuvo su impacto en su competidor directo, Google y su Project Tango, el cual se termina cerrando para centrarse en el desarrollo de ARCore.

En 2018, sale al mercado la segunda generación de dispositivos Oculus, Oculus Go,

con un resultado gráfico similar a la primera generación pero con grandes avances. Este nuevo dispositivo de realidad virtual, a pesar de tener solo tres grados de libertad, funciona de manera completamente independiente (sin necesidad de un ordenador) y su precio es mucho más asequible. Google lanza ARCore con unas características similares a ARKit: reconocimiento de imágenes, planos horizontales y verticales con la diferencia de que ARCore es compatible tanto con los dispositivos con sistema operativo iOS y Android (con versiones superiores a la 7.0).

2.2. Continuo de la virtualidad y realidad extendida

El término continuo de virtualidad fue acuñado por Paul Milgram y Fumio Kishino en el año 1994. En él se sitúa en un extremo la realidad física, y en el otro, la realidad virtual, de manera que el espacio comprendido entre ambos está formado por los entornos donde se mezclan lo real y lo virtual indistintamente para formar lo que se conoce como realidad mixta. Según el grado de virtualidad presente en estos entornos podemos distinguir entre realidad aumentada, virtualidad aumentada o realidad virtual, siendo este último, el único entorno puramente virtual donde no participa la realidad física.

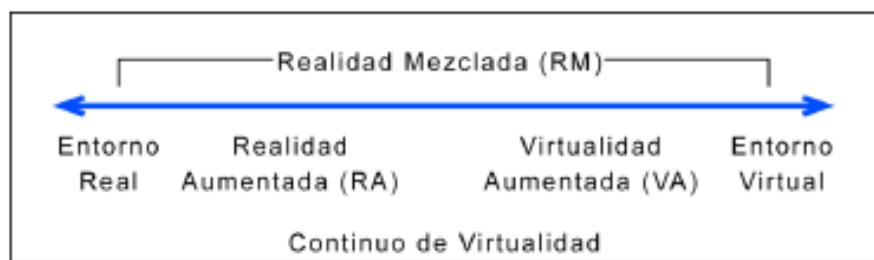


Figura 2.4: *Representación del continuo de realidad*

Debido a los avances tecnológicos de estos últimos años, como impresoras 3D, *chatbots*¹⁴, inteligencias artificiales o sensores inteligentes, los elementos reales están empezando a estar cada vez más ligados al mundo digital. La realidad extendida, o XR, es un concepto que describe todo el espectro de la realidad, desde la realidad virtual a la realidad física. La llegada de la XR supondrá la unión entre el mundo real y el mundo digital, donde lo virtual amplia la realidad y lo real persiste en un espacio virtual [8].

¹⁴Un *chatbot* es un programa informático con el que es posible mantener una conversación, se le puede pedir algún tipo de información o que realice una determinada acción.

2.2.1. Realidad aumentada

La realidad aumentada es una tecnología que permite visualizar la realidad a través de un dispositivo añadiendo información virtual a la información física existente. Este término aparece por primera vez en 1992, cuando el científico Tom Caudell estaba desarrollando un software para mejorar la actividad de los operarios de Boeing¹⁵. Caudell pensó que los operarios perdían mucho tiempo interpretando las instrucciones y que una pantalla que las proyectase mejoraría el ensamblaje de las piezas, para así minimizar los tiempos de producción [9].

Aunque Caudell fuera el primero en emplear este término, mucho antes aparecieron varios proyectos que empleaban elementos característicos de esta tecnología. En el año 1957, Morton Heilig empieza a construir Sensorama. Esta máquina proyectaba imágenes grabadas de las calles de Brooklyn en tres dimensiones dentro de una cabina que producía estímulos sensoriales a quien la utilizaba, es decir, ampliaba la información que recibía el usuario del entorno real. Estos inventos no tuvieron éxito debido a la falta de tecnología de la época, hasta que en el año 2016 la RA empezó a popularizarse con el lanzamiento del videojuego Pokemon Go. Este videojuego acercó la realidad virtual a más de sesenta y cinco millones de jugadores, que utilizaban únicamente sus dispositivos móviles para poder jugar. Los últimos modelos de *smartphones* disponibles en el mercado ya están preparados para soportar esta tecnología, por lo que muchas compañías están empezando a desarrollar aplicaciones RA, por ejemplo, Ikea. La empresa de muebles sueca ha desarrollado la aplicación *Ikea Place* en la cual se puede ver todo su catálogo de muebles y utilizar la RA para ver cómo quedarían en nuestras casas [10].

La RA también está creando tendencia dentro del mundo del *marketing*, ya que cada vez son más las empresas que están empleando esta tecnología para realizar sus campañas publicitarias. Este es el caso de Zara, que ha dejado de lado los maniquíes tradicionales para mostrar sus productos al consumidor utilizando modelos profesionales en una pasarela improvisada dentro de la tienda, por medio un dispositivo móvil.

Además de los *smartphones*, existen distintos tipos de dispositivos que han sido diseñados exclusivamente para esta tecnología, como por ejemplo las gafas de realidad au-

¹⁵Boeing es una empresa multinacional estadounidense que diseña, fabrica y vende aviones, helicópteros, misiles y satélites.

mentada. Estas gafas proyectan sobre las lentes los elementos virtuales, de tal forma que desde el punto de vista de la persona que las lleva, imágenes digitales se funden con el entorno físico. Dentro de esta categoría se encuentran las HoloLens de Microsoft o las Google Glass, su principal inconveniente es su precio, debido a que las tecnologías que utilizan, a día de hoy son muy costosas. Uno de los proyectos que puede revolucionar el sector de las gafas de realidad aumentada son las gafas de realidad aumentada de Magic Leap. Se espera que estas gafas salgan al mercado en los próximos años. Hasta ahora cuenta con un gran número de patrocinadores, se estima que la empresa recaudó dos mil millones de dólares a lo largo de sus cuatro años de desarrollo. Hasta ahora este desarrollo se ha realizando de manera oculta, las únicas personas que han podido probar los prototipos han sido algunos patrocinadores y los desarrolladores. Magic Leap promete conseguir una mejor resolución utilizando una nueva tecnología de campos de luz que hasta ahora no se ha podido llevar a cabo.

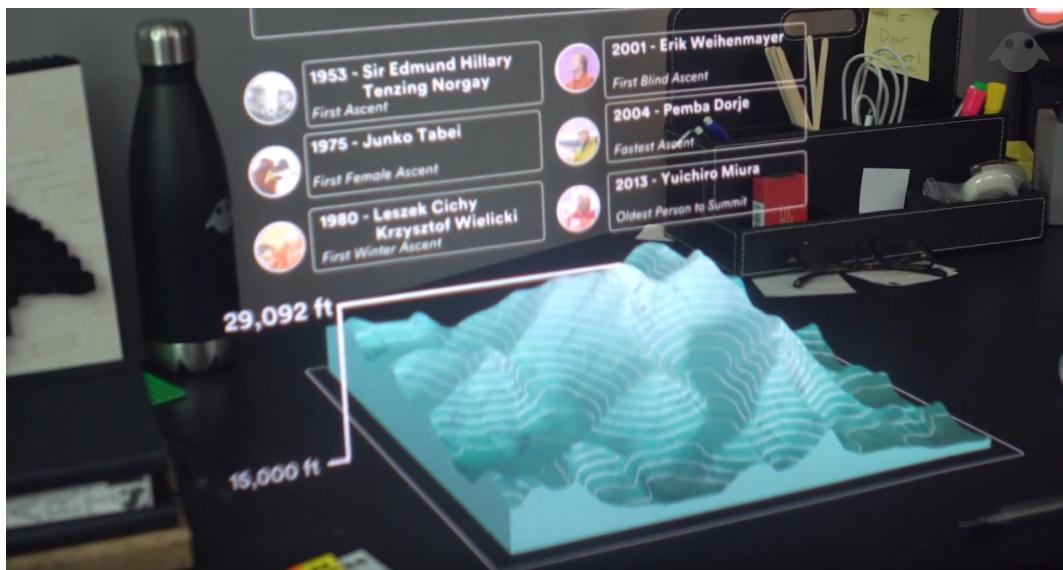


Figura 2.5: Prototipo de Magic Leap en 2016.

Otro proyecto interesante dentro de la RA es el denominado *Deep Frame*, de la empresa danesa Real Fiction. Este proyecto introduce un nuevo concepto, que se acerca más a los hologramas de películas de ciencia ficción que a los dispositivos de realidad aumentada. Consiste en una especie de ventana fija en la cual se proyectan los elementos virtuales. Aun no está abierto al gran público y su objetivo se centra en el *retail* y en los museos o instalaciones con fines educativos.

2.2.2. Virtualidad aumentada

La virtualidad aumentada (VA), es quizás de estas tres formas de representación de la virtualidad la más aceptada dentro de la sociedad. Este término no es nuevo y consiste en aumentar el entorno virtual utilizando estímulos procedentes del mundo tangible como las manos, el rostro u objetos. Dentro de la línea de continuidad de la virtualidad, la VA es la que más se acerca a la realidad virtual, ya que la parte digital tiene un peso determinante, mientras que el mundo físico está presente como un refuerzo.

Hay un amplio abanico de posibilidades dentro de la VA, por ejemplo, la captura de una imagen del mundo físico con una cámara y esta se incorpora al mundo virtual. Este es el caso del croma o el juego *EyeToy* de PlayStation. Otro planteamiento de la VA diferente al anterior es añadir elementos físicos para mejorar la consistencia de la experiencia en el entorno virtual. Los diversos elementos que pueden participar en la experiencia son muy variados, desde paredes, emisores de vapor, ventiladores, sillones motorizados hasta atracciones de parques temáticos. Este es el caso de la atracción TNT Tren de la mina del Parque de Atracciones de Madrid, mientras dura el trayecto los pasajeros utilizan gafas de realidad virtual haciendo que los estímulos externos producidos por el movimiento de los vagones sobre las vías complementen la experiencia virtual.

2.2.3. Realidad virtual

La realidad virtual se distingue del resto por ser capaz de sumergir por completo al usuario en un mundo virtual. Esta representación del entorno o imágenes de objetos está generada por un sistema informático permitiendo una experiencia sensorial puramente artificial. La experiencia puede desarrollarse utilizando para ello una representación de escenas que simulen la realidad o por el contrario puede simular entornos fantásticos.

Las aplicaciones que puede tener la VR se pueden aplicar a distintos sectores, como puede ser el arte, la educación, el entretenimiento, la medicina o la arquitectura entre otros. Permite simular situaciones peligrosas para la integridad de una persona minimizando los riesgos y el coste, por eso también se emplea en sectores de investigación o militares. A continuación se explican unos fundamentos esenciales para poder entender el funcionamiento de la realidad virtual.

2.2.3.1. Head-Mounted Display

Un *Head-Mounted Display*, conocido habitualmente por sus siglas HMD, es un dispositivo más común utilizado para experiencias de realidad virtual. Está formado por una o dos pantallas de alta resolución que se sitúan delante de los ojos sobre las que se proyecta el entorno virtual utilizando una técnica que se conoce como estereoscopía¹⁶. Esta técnica recibe un entorno tridimensional, lo interpreta y genera dos imágenes bidimensionales para cada ojo que dan una sensación de profundidad si se sitúan de forma correcta. Los HMD también suelen tener unos auriculares incorporados para mejorar la experiencia virtual. Si bien se ha mencionado anteriormente que los HMD son el elemento más común para experiencias de realidad virtual, también se puede considerar HMD unas gafas de realidad aumentada o mixta ya que las tecnologías que utilizan son muy similares.



Figura 2.6: *Magic Leap One HMD*.

Además de los HMD existen otros dispositivos que merecen ser mencionados como las Cardboard de Google o las Samsung GearVR. Estas formas de visualización de realidad virtual necesitan de una plataforma en la cual se pueda colocar un teléfono móvil a modo de casco de realidad virtual. Esta plataforma puede ser tan sencilla, que se puede

¹⁶La estereoscopía fue desarrollada por Sir Charles Wheatstone a modo de experimento para probar su teoría de la visión estereoscópica en 1840.

realizar en casa, utilizando cartón y dos lentes a modo de lupas, como las Cardboard de Google.

Otra forma de entender la realidad virtual, aunque más costosa y solo únicamente empleada en entornos de investigación, sería la tecnología *Cave Automatic Virtual Environmental* (CAVE) [11]. Este sistema fue diseñado por científicos de la Universidad de Illinois en 1991. Se trata de una sala en forma de cubo donde se proyectan sobre las distintas paredes, techo y suelo imágenes tridimensionales. Además, el usuario tiene que llevar unas gafas 3D para ver los gráficos tridimensionales.



Figura 2.7: Sistema CAVE.

2.2.3.2. Grados de libertad

Un término muy importante dentro de la realidad virtual son los grados de libertad. Los grados de libertad hacen referencia al movimiento en un espacio tridimensional, es decir, la capacidad de translación en el sentido de los ejes perpendiculares, combinados o no, con la rotación sobre estos mismos ejes. Se puede distinguir dos tipos de dispositivos de realidad virtual, aquellos que tienen tres grados de libertad o los que tienen seis grados de libertad. Los dispositivos que tienen tres grados de libertad hacen referencia a la rotación dentro del entorno, entre estos dispositivos se encuentra los dispositivos móviles con giroscopio¹⁷ acoplados a una Cardboard para simular unas gafas de realidad virtual. También tienen tres grados de libertad la segunda generación de Oculus standa-

¹⁷Aparato formado por un disco pesado que gira a gran velocidad sobre un eje, para que cualquier alteración en la inclinación de este provoque un movimiento de precisión que lo contrarreste.

*lone*¹⁸, Oculus Go, que tiene tres grados de libertad tanto en las gafas como en el mando. Los dispositivos con seis grados de libertad permiten, además de la rotación, la translación dentro del entorno, estos dispositivos suelen utilizar sensores infrarrojos para capturar el movimiento.

2.2.3.3. Tracking

El concepto de *tracking*, su traducción en castellano sería rastrear, hace referencia a la captura de los movimientos del usuario en el entorno físico. Para la captura de movimientos utilizan sensores que delimitan la zona de acción de usuario, esta forma de *tracking* se conoce como *outside-in*. Entre estos dispositivos se encuentran las gafas de realidad virtual HTC Vive que emplean de 1 a 3 sensores de infrarrojos para determinar la posición del usuario. Otra forma menos común de conseguir este posicionamiento en el espacio es mediante cámaras o sensores incorporados al casco de realidad virtual. Este *tracking* es más complejo y utiliza técnicas comunes en el mundo la robótica como algoritmos SLAM, que se emplean para construir un mapa tridimensional de un entorno desconocido.

2.2.3.4. Campo de visión

El campo de visión, también conocido como *Field Of View* (FOV), es el ángulo delimitado por dos líneas imaginarias que definen la parcela de realidad captada por los ojos. En caso de las realidad extendidas se puede definir como el ángulo que se puede percibir en un entorno sintético generado por un dispositivo de visualización.

El FOV del ojo humano es aproximadamente 200 grados en horizontal y 180 grados en vertical. Las gafas de realidad virtual actualmente tienen un campo de visión que oscila entre los 90 y los 120 grados de FOV. El efecto que produce ver la realidad virtual con un FOV más pequeño que el natural del ojo se conoce como *efecto buzo*. Recibe este nombre por la similitud con la sensación de llevar unas gafas que limitan la visión a los lados de la cabeza. Por este motivo se están empezando a diseñar nuevas gafas de realidad virtual con un FOV cercano a los 200 grados en horizontal.

¹⁸Este término inglés se emplea para matizar que este dispositivo funciona independiente de un ordenador.



Figura 2.8: *Campo de visión de StarVR*.

Aunque la amplitud del ángulo de visión del ser humano sea tan amplia como se ha mencionado anteriormente, el ojo humano no es capaz de observar con nitidez las porciones de visión en las que no se está focalizando. Este hecho se está utilizando para desarrollar nuevos dispositivos de realidad virtual que únicamente sean capaces de renderizar la zona de visión en la que está centrando su atención el usuario. Consiguiendo este efecto se podrían optimizar la construcción del entorno digital ahorrando recursos de procesamiento. Esta técnica se conoce como *foveated rendering*, el único problema es que aun no hay una tecnología eficiente capaz de hacer un *tracking* del movimiento del ojo suficientemente bueno, debido a que este movimiento es uno de los más rápidos que se dan en el cuerpo del ser humano.

2.2.3.5. Frecuencia de refresco

La frecuencia de refresco es el número de imágenes que se visualizan por unidad de tiempo. Podemos distinguir entre dos frecuencias de refresco: *vertical frequency*, es decir, la frecuencia del dispositivo, y la frecuencia de la aplicación o *frame rate*.

La primera hace referencia al número de imágenes que es capaz de mostrar un dispositivo en un segundo, esta frecuencia se mide en Hz. Mientras que el *frame rate* es el número de fotogramas que es capaz de calcular en un segundo, esta medida se realiza utilizando FPS (*frames per second*).

En el caso de la realidad virtual este punto es bastante importante debido a que una tasa baja de FPS, dará la sensación al usuario de que percibe imágenes con saltos, en ocasiones esta sensación puede llegar incluso a causar mareos. Lo habitual es que la frecuencia vertical de barrido del dispositivo duplique la tasa mínima establecida en 30 imágenes por segundo. En el caso de los FPS, para tener una sensación fluida dentro del entorno virtual la aplicación debe rondar entre los 90 y 100 FPS.

2.2.3.6. Accesorios

Dentro del mundo de la realidad virtual encontramos una amplia variedad de accesorios. Todos ellos pensados de manera muy original para acercarse cada vez a una experiencia tan realista, dentro de un entorno virtual, que incluso reemplace el mundo físico.

Uno de los puntos débiles de los entornos virtuales es engañar el sentido del tacto del usuario. Dentro de este mundo digital se puede interactuar con diversos objetos pero no se puede experimentar ni la rugosidad o el peso que este produce al estar en la mano. En este aspecto se centran los guantes hápticos como Glove One. Existen también trajes hápticos, dando un paso más allá que los guantes, que estimulan el cuerpo de tal manera que el usuario note su presencia dentro del mundo virtual. Para producir estímulos para el sentido del olfato existe un prototipo de máscara que se acopla a los dispositivos de realidad virtual convencionales llamado FeelReal. Estas máscaras son capaces de simular olores, expulsar aire frío o caliente para aumentar la experiencia virtual.

La forma en la que las personas se mueven dentro del entorno virtual es muy diferente a cómo se mueven en el mundo físico. En el mundo virtual el usuario puede hacer uso del teletransporte o utilizar un *joystick* para desplazarse. Estas son dos de las formas más comunes de movimiento que se utilizan en las aplicaciones de realidad virtual. Esto es así, porque el usuario en el mundo físico se ve muy limitado en las distancias que puede recorrer. Para solucionar este inconveniente hay una gran cantidad de prototipos en desarrollo que permiten, andar, correr, saltar o incluso sentarse, utilizando una plataforma

compatible con los principales dispositivos de realidad virtual.

Uno de los accesorios que más se está utilizando para combinar con dispositivos de realidad virtual son los sensores de *Leap Motion*. Estos sensores permiten hacer un *tracking* de las manos del usuario, detectan cada uno de los dedos y su posición en tiempo real. La empresa que los desarrolla, Leap motion, ha lanzado herramientas para que los desarrolladores puedan incorporar esta tecnología en sus aplicaciones.



Figura 2.9: Accesorio *Leap Motion* para detección de manos.

En la conferencia Siggraph¹⁹ de 2018, la empresa Neurable presentó uno de sus proyectos más ambiciosos. Consiste en un dispositivo que se acopla como un casco sobre las gafas de realidad virtual que actúa a modo de interfaz entre el cerebro y los objetos interactivos del entorno virtual. Este proyecto, aun en desarrollo, emplea una tecnología que permite capturar los impulsos del cerebro, interpretarlos y generar información de entrada para el ordenador.

¹⁹Siggraph es un grupo de interés en la infografía y la computación gráfica que organiza conferencias desde 1974 principalmente en EEUU.

2.3. Motores gráficos

El término motor gráfico o motor de videojuegos empezó a utilizarse en los años 90 con el lanzamiento del videojuego *Doom*. Este juego se hizo muy popular por la facilidad que tenían los jugadores para modificar su código creando nuevas dinámicas de juego, nuevos mapas o incluso nuevas expansiones que mejoraban el juego original. Este motor gráfico se podía reutilizar para crear nuevos videojuegos de disparos en primera persona o *first person shooter* reduciendo enormemente los costes y los tiempos de desarrollo.

Los motores gráficos son soluciones *software* que facilitan el desarrollo de mecánicas de juego o aplicaciones que tengan un comportamiento similar. Es decir, un motor de videojuegos garantiza una infraestructura que permite reutilizarse para distintos videojuegos del mismo o de distinto género, además de que la implementación de este tiene que poder realizarse para diferentes plataformas.

2.3.1. Arquitectura

La construcción de un motor gráfico no es una tarea sencilla, por ello su estructura está basada en el paradigma de programación orientado a objetos para simplificar su diseño. Esto quiere decir que el motor de juegos no es un único elemento que se encarga de realizar todo el proceso, está formado por un conjunto de subsistemas cada uno con sus tareas específicas. Los motores gráficos utilizan un modelo de arquitectura estructural en capas. Este tipo de arquitecturas tienen la peculiaridad de que los elementos de las capas de nivel superior dependen de las anteriores, pero no al contrario. En los siguientes apartados se explicarán cada una de las capas que forman un motor de juego [12].

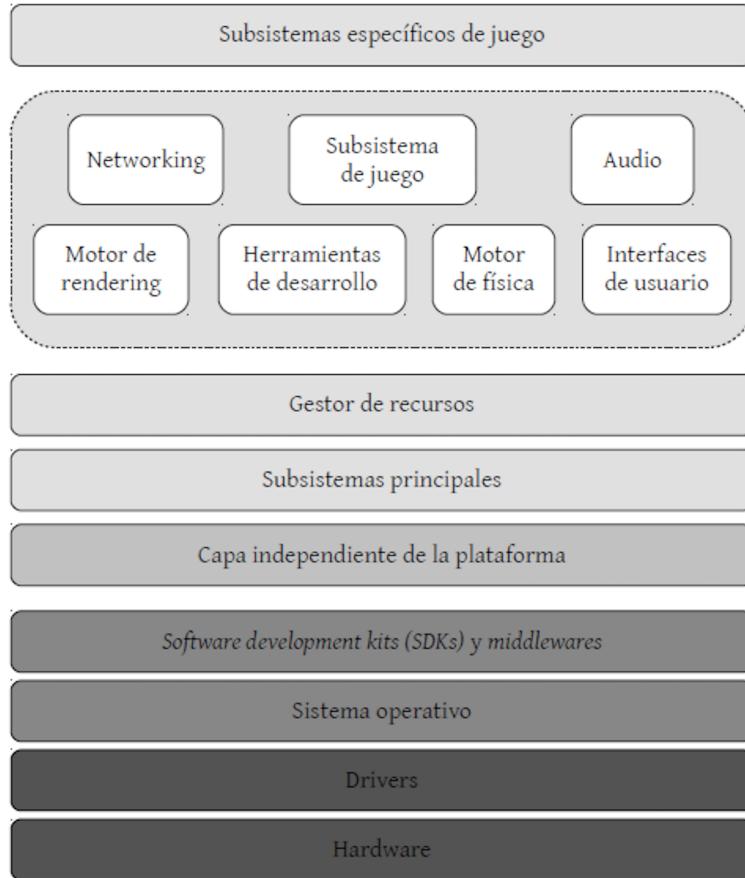


Figura 2.10: *Estructura de capas que forma un motor de juego.*

2.3.1.1. Hardware, drivers y sistema operativo

Estás tres capas no forman parte del motor gráfico como tal, pero influyen en él. En el caso del *hardware*, está vinculado a la plataforma en la que se ejecutará el motor de juegos, es muy común que los desarrolladores modifiquen el motor de juegos en base a la plataforma para mejorar su rendimiento [12].

La capa de *drivers* está formada por todos los componentes *software* que manejan el intercambio de información con distintos dispositivos. En el caso de los motores de juegos los dos dispositivos más influyentes son las tarjetas de vídeo y las de sonido.

Por último, la capa del sistema operativo. Esta capa es la que se encarga de gestionar los procesos dentro de los elementos *hardware* asociados a la plataforma del motor de juegos. Las consolas de última generación cuentan con un sistema operativo que se encarga de gestionar los procesos y los recursos, de tal manera que incluso es posible detener

la ejecución de un videojuego.

2.3.1.2. Librerías

Las aplicaciones informáticas suelen apoyarse en bibliotecas existentes para agilizar el proceso de desarrollo. Para ello se pueden llegar a utilizar SDKs (*Software Development Kit*) o APIs (Application Programming Interface) para proporcionar una determinada funcionalidad. Los desarrolladores utilizan habitualmente librerías externas modificadas por ellos mismo para adaptarlas a sus necesidades.

Es muy común utilizar APIs gráficas para tratar problemas comunes en todos los videojuegos como el renderizado de objetos tridimensionales. Las APIs gráficas más utilizadas son DirectX de Microsoft y Vulkan, mantenida por Khronos Group.

Los motores de videojuegos suelen utilizar motores de físicas para gestionar el comportamientos de los objetos en el mundo virtual, así como la detección y el tratamiento de las colisiones que se producen. Hay una gran variedad de motores de físicas, cada uno con sus propias peculiaridades, podemos destacar PhysX de la compañía Nvidia o CaronteFX de la compañía Nextlimit.

2.3.1.3. Capa independiente de la plataforma

Aunque se utilicen librerías de bajo nivel estandarizadas es necesario utilizar una capa que cumpla la función de envolver los módulos de las capas superiores para que su desarrollo se haga de forma independiente a las capas inferiores. Esta capa es importante, ya que es muy habitual en mundo de los videojuegos, que el mismo videojuego se lance para múltiples plataformas, por lo que es necesario mantener de forma independiente la plataforma y el desarrollo del videojuego.

2.3.1.4. Subsistemas principales

En esta capa se encuentran todas las librerías que dan soporte al motor de juego independientemente del tipo de videojuego que se vaya a desarrollar. La capa de subsistemas principales incluye desde las librerías matemáticas, que proporcionan al desarrollador los diferentes tipos de datos y las operaciones que definirán el entorno digital y su comporta-

miento, hasta librerías de gestión de memoria y depuración de código.

2.3.1.5. Gestor de recursos

Esta capa se emplea para unificar todos las entidades *software* o recursos que forman el motor de juegos. Se encarga de su creación y su gestión dentro de la aplicación, recursos como la creación o el intercambio entre escenas, las colisiones, las cargas de texturas y materiales o la representación de los objetos dentro del mundo tridimensional.

2.3.1.6. Networking.

El módulo de *networking* ha adquirido una gran importancia estos últimos años, debido a que muchos juegos comerciales incluyen modos de juego multijugador en línea. Este componente se encarga de la comunicación por red, típicamente esta información se transmite utilizando *sockets* donde se envía únicamente la información relevante para evitar sobrecargar la aplicación con salidas y entradas.

2.3.1.7. Subsistema del juego

Esta capa separa la lógica de la aplicación formada por sistemas de *scripting* junto con los lenguajes de alto nivel. La lógica de la aplicación gestiona los elementos que forman parte del entorno de la aplicación: cámaras, luces, cuerpos rígidos dinámicos, elementos geométricos, etc. El sistema de *scripting* se utiliza también para mejorar el proceso de desarrollo y evitar tener que compilar el motor en cada una de las comprobaciones.

Otro de los elementos que forman parte de esta capa es la gestión de los eventos. Es muy común en este tipo de aplicaciones o videojuegos la utilización de eventos, ésta permite la comunicación entre los diversos elementos presentes en la escena.

2.3.1.8. Audio

El apartado gráfico parece tener más peso que el audio en un videojuego, aun así es un elemento muy a tener en cuenta. Este componente se encarga de dar presencia a los elementos con sonoridad dentro del entorno, de tal manera que el usuario sea capaz de

identificar que el sonido procede de un elemento en concreto. Habitualmente este componente no se desarrolla, sino que se emplean componentes de audio modificados de otras compañías.

2.3.1.9. Motor de *rendering*

El motor de *rendering*, como indica su propio nombre, se encarga del renderizado. El renderizado es el proceso por el cual los elementos situados dentro de la escena tridimensional pasan a ser proyectados en una pantalla bidimensional. Este componente se encarga de gestionar tareas complejas con una alta carga computacional como: cuáles objetos serán visibles, qué materiales y texturas tienen que ser aplicados o de qué manera afectarán las luces del entorno a dichos objetos.

Existen dos formas distintas de renderizado:

- **Renderizado en tiempo real**

Su objetivo principal es renderizar con la mejor calidad posible contando con las limitaciones de mantener una tasa de *frames* por segundo aceptables. Este tipo de renderizado es el que se utiliza en videojuegos o aplicaciones con gráficos interactivos.

- **Pre-Renderizado**

El pre-renderizado se utiliza en situaciones donde el tiempo no es tan relevante, por lo que el único objetivo es garantizar el mejor resultado visual posible. Este renderizado se utiliza en animaciones, cinemáticas o efectos especiales. Es típico utilizar ordenadores muy potentes diseñados exclusivamente para este fin y dejarlos funcionando durante grandes periodos de tiempo para lograr el mejor *render* posible.

El motor de *rendering* es uno de los componentes más significativos dentro de un motor gráfico y además cuenta con el hándicap de que tiene que ser actualizado de manera constante debido a los avances tecnológicos en el sector gráfico. Del mismo modo que la construcción de un motor de juego es una tarea complicada, construir un motor de

rendering también lo es. Por ello su arquitectura sigue el mismo modelo estructural por capas.

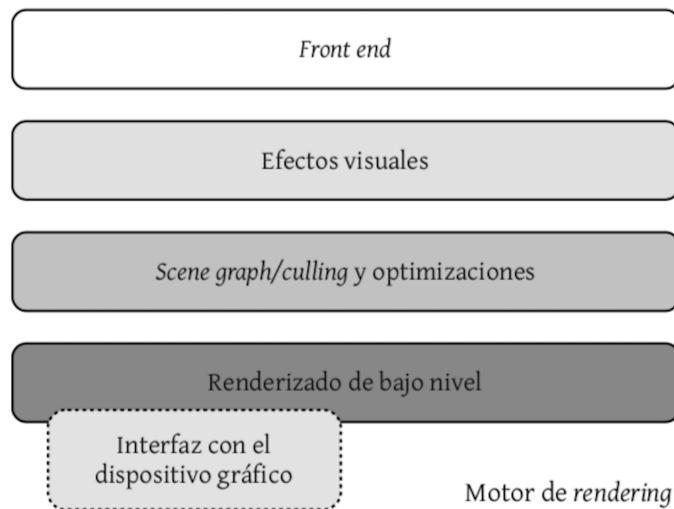


Figura 2.11: Visión conceptual de la estructura de un motor de rendering [12].

La capa de renderizado de bajo nivel tiene como principal objetivo renderizar las distintas primitivas geométricas de la manera más rápida posible sin tener en cuenta las posibles optimizaciones que se puedan realizar, pero si que tiene en cuenta la posición de la cámara. En esta capa también se gestionan las interacciones con las APIs gráficas (módulo de interfaz con el dispositivo gráfico), así como la gestión de la cámara y los diferentes modos de proyección. Por otra parte, se gestionan también el estado del *hardware* y los *shaders*. Los *shaders* se encargan de guardar la información asociada de cómo afecta la luz a los distintos materiales de las primitivas geométricas de la escena.

La capa superior al renderizado de bajo nivel se conoce como *scene graph/culling* y optimizaciones. En esta capa se seleccionan únicamente los elementos que van a ser renderizados, con la finalidad de mejorar el rendimiento del motor de *rendereing*. Esta capa también integra elementos de *culling*, es decir, los elementos encargados de determinar que objetos están siendo solapados por otros, evitando que los objetos no visibles desde la perspectiva de la cámara sean renderizados.

En la capa de efectos visuales se gestionan todo tipo de efectos gráficos como los efectos de partículas (agua, humo, polvo, fuego, etc), los mapeados de los modelos o las sombras dinámicas.

La capa de *front end* es la que se encarga de gestionar los elementos 2D, como un menú o cualquier tipo de interfaz gráfica, que se sobreponen a los objetos 3D.

2.3.1.10. Herramientas de desarrollo

Las herramientas de desarrollo son todo el conjunto de herramientas que se utilizan para depurar y optimizar el motor de juego y así tener el mejor rendimiento posible. Los desarrolladores suelen utilizar sus propias herramientas de desarrollo para cada aplicación gráfica, aunque también existen herramientas que son facilitadas por el propio motor gráfico. Un ejemplo de este tipo de herramientas puede ser un mecanismo capaz de medir el tiempo que tarda en ejecutarse un fragmento de código, una representación visual del rendimiento en ejecución o herramientas para medir los recursos *hardware* que se están utilizando.

2.3.1.11. Motor de física

Este es el componente que se encarga de atribuir leyes físicas a los objetos virtuales, de forma que estos se vean afectados por las fuerzas que actúan sobre ellos y las colisiones con otros objetos.

Los objetos dentro de la escena que son visibles habitualmente cuentan con una malla de vértices, también conocida como *mesh*. Esta malla por si sola no posee físicas y si un objeto con físicas entra en contacto con ella, este lo atravesaría como si no estuviese. Para que el objeto colisione necesita un *collider*, una malla de puntos que envuelve el *mesh* del objeto y hace que ocupe un volumen dentro de la escena. Hay muchas técnicas para generar un *collider*, pero para simplificar, se podría decir que cuanto más parecida sea la malla *mesh* a la malla del *collider*, el volumen que ocupa el objeto será más realista, por el contrario, cuanto menos vértices tenga el *collider* mejor será el rendimiento de la aplicación.

Ahora bien, si este nuevo objeto con componentes *mesh* y *collider* entrase en contacto con otro objeto con físicas se produciría una colisión, pero el objeto inicial permanecería inamovible. Para que este objeto pueda recibir fuerzas externas, como la gravedad, necesita un componente que lo convierta en un cuerpo rígido.

El motor de físicas también se encarga de simular sistemas complejos de alta carga computacional como fluidos, humo, fuego o rayos. Este tipo de sistemas suelen simularse de manera externa y luego son incluidos a modo de animación pre-calculada en la aplicación.

2.3.1.12. Interfaces de usuario

Este componente se encarga de gestionar la conexión entre los elementos a bajo nivel utilizados para el acceso al *hardware* y los controladores de alto nivel. Para realizar esta tarea utiliza un sistema de eventos que reciben los parámetros de entrada o interacciones del usuario y muestra las salidas, que son los resultados de la interacción con la aplicación.

2.3.1.13. Sistemas específicos de juego

La capa de sistemas específicos de juegos es la capa que separa el motor gráfico de la aplicación gráfica, en esta capa se implementan todo lo relacionado con las características propias de la aplicación.

2.3.2. Unity

Unity es un motor de juego desarrollado por Unity Technologies, utilizado para crear videojuegos y contenidos interactivos en 2D y 3D. Este motor se ha vuelto muy popular dentro de la industria por su relación calidad-precio, cuenta con una versión gratuita y una versión profesional, por 125\$ al mes, que permite desarrollar con Unity para la gran mayoría de plataformas [13]. Otra de características por las que Unity se ha vuelto tan popular es por su rápida curva de aprendizaje, existe una amplia variedad de documentación y tutoriales con todo lo que este motor de juego puede ofrecer. La misma plataforma oficial de Unity facilita una serie de tutoriales para conocer los fundamentos básicos del programa. Dentro del sector de las empresas que no se dedican al desarrollo de videojuegos, también es una opción muy recurrente utilizar Unity para la creación de contenidos interactivos. Su principal ventaja frente a otros motores de juego es su versatilidad, la cual permite que estos contenidos lleguen a un público más amplio. Unity tiene soporte para desarrollar aplicaciones para más de 25 plataformas entre móvil, escritorio, consola, TV,

VR, AR y web.

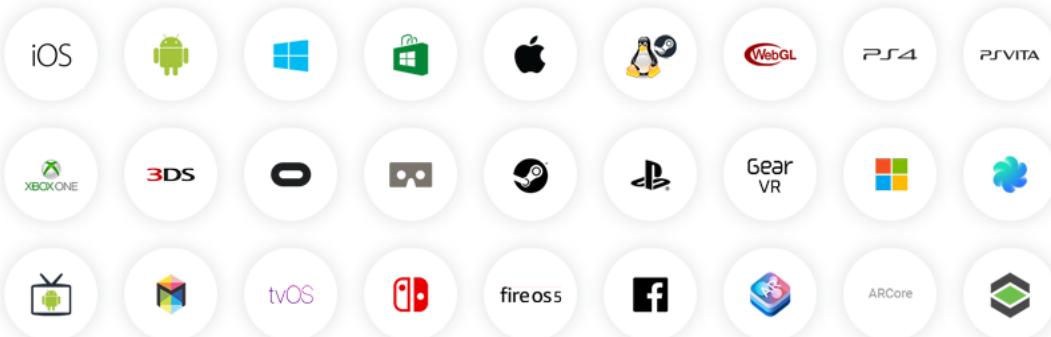


Figura 2.12: *Plataformas para las cuales se puede desarrollar con Unity.*

El motor esta implementado en C++, aunque los lenguajes de programación que permite compilar son C# y UnityScript, un lenguaje específico para este motor basado en JavaScript. Unity no permite la modificación del código fuente para adaptarlo al proyecto que se este desarrollando.

El editor de Unity es el entorno de desarrollo, disponible par Mac y para Windows, desde el cual tenemos acceso a todas las herramientas que nos ofrece Unity. Estas herramientas se utilizan para el diseño de los entornos tridimensionales, realizar las animaciones o implementar la lógica del juego. Unity da soporte tanto aplicaciones en 3D como aplicaciones 2D, por lo que muchas de estas herramientas se pueden utilizar en ambas modalidades. Una de las herramientas más interesantes que ofrece el editor de Unity, relacionado con el seguimientos de rutas, es un sistema de navegación, este puede ser aplicado para crear NPCs²⁰ que se muevan por el entorno de forma inteligente.

En cuanto a las físicas, toda las funcionalidades del motor de Unity están proporcionadas por los motores de físicas de Box2D y Nvidia PhysX. Estos motores garantizan una experiencia de juego realista y de alto rendimiento.

Unity ha incluido en sus últimas versiones una nueva multiplataforma de realidad extendida nativa que tiene soporte para Google Daydream, Oculus, ARCore y Magic Leap, entre otros.

²⁰Un NPC es personaje no jugador controlado por el director de juego.

3. ANÁLISIS DEL PROBLEMA

En este capítulo se va a explicar el proceso seguido para realizar el análisis de la situación previa al desarrollo de la aplicación, así como la toma de requisitos. También se presentarán los recursos elegidos para la realización del proyecto.

3.1. Descripción del problema

El departamento de automática de la Universidad de Alcalá Henares está desarrollando un proyecto de investigación sobre exploración espacial, en concreto sobre la exploración de la superficie de Marte. El equipo del proyecto ha diseñado un algoritmo de planificación de rutas llamado *3D Accurate Navigation Algorithm* (3Dana) que utiliza mapas de costes y/o *Digital Terrains Models* (DTM) obtenidos del *High Resolution Imaging Science Experiment* (HiRISE).

La aplicación a desarrollar en este trabajo de fin de grado tiene como propósito principal estudiar las nuevas tecnologías disponibles de realidad virtual y realizar una aplicación que permita la simulación de las operaciones realizadas en la superficie de Marte. Un aspecto importante de la planificación de las misiones son las rutas que el rover tiene que seguir para alcanzar los puntos de interés científico. Para la planificación de rutas se emplea una combinación de imágenes obtenidas desde la órbita y desde la superficie, que permiten analizar la topografía y generar un mapa de costes. Aplicando los algoritmos pertinentes a estos mapas de costes es posible obtener soluciones óptimas para dirigir el rover a través de los distintos puntos. En concreto, el algoritmo utilizado para programar estas rutas será el algoritmo 3Dana mencionado anteriormente.

Se pretende desarrollar una aplicación de bajo presupuesto que proporcione una vista tridimensional en un entorno de realidad virtual que permita visualizar los datos de telemetría e incorpore las funciones de telecontrol para explorar la superficie de Marte con la intención de ayudar a los investigadores a decidir qué rutas seguir. Además de realizar la simulación en un dispositivo de realidad virtual, se realizará una aplicación para ordenador que permita la misma simulación en el entorno tridimensional.

3.2. Historias de usuario

En la metodología Scrum, las características del proyecto son tomadas desde la perspectiva del usuario final, estas características se conocen como historias de usuario. El *backlog* está formado por todas las historias de usuario.

El *backlog* se realizó en varias reuniones entre los tutores y el autor del proyecto. La lista de historias de usuarios ha sido modificada conforme avanzaba el proyecto para ajustarse más a la solución del problema, la lista definitiva se expone a continuación.

1. La primera escena que ve el usuario es un menú principal donde puede acceder a las funcionalidades básicas de la aplicación.
2. El usuario tendrá la posibilidad de crear nuevas rutas a través de la aplicación de ordenador y poder cancelar su creación.
3. El usuario podrá elegir la ruta que desee utilizar para la realización de la simulación de la misión, entre una variedad de mapas con varias rutas. Este requisito es independiente de la plataforma que este utilizando el usuario.
4. El usuario podrá elegir el mapa que desee utilizar para la simulación de la misión.
5. El usuario siempre tendrá la opción de simular la misión con realidad virtual.
6. El usuario podrá realizar la simulación en 3D sin necesidad de dispositivos de realidad virtual en la aplicación de escritorio.
7. El usuario debe poder salir de la misión en cualquier momento.
8. El usuario podrá caminar libremente por la superficie de Marte.
9. El usuario realizará la simulación posicionado en la parte superior de rover.
10. La simulación contará con un modelo de baja resolución que emule los movimientos del rover de sobre la superficie de Marte.
11. El usuario podrá elegir mapa, misión y realizar la simulación desde la plataforma Samsung GearVR.

3.3. Aplicaciones similares

En este apartado se van a mostrar algunos de los proyectos con características similares a la aplicación desarrollada. Estos proyectos han servido de fuente de inspiración para el desarrollo de este trabajo. En especial, cómo se ha gestionado la experiencia del usuario, los controles e interacciones que son a día de hoy uno de los problemas de la VR.

3.3.1. OnSight

OnSight es una aplicación desarrollada por la NASA en colaboración con Microsoft que permite a los científicos trabajar sobre la superficie de Marte desde sus oficinas utilizando HoloLens, las gafas de realidad mixta de Microsoft. Esta aplicación sirve de medio para los científicos e investigadores del *Jet Propulsion Laboratory* (JPL)²¹ de Los Ángeles para planificar las operaciones del rover Curiosity [14]. OnSight permite a científicos de todo el mundo encontrarse en un entorno virtual de la superficie marciana, compartir datos reales sobre el rover permitiendo una mejor planificación de las misiones de exploración.

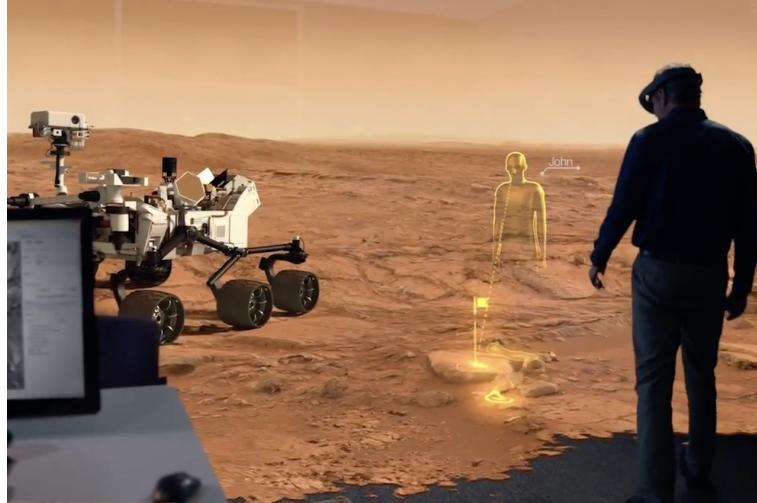


Figura 3.1: Captura de Onsight.

²¹El *Jet Propulsion Laboratory* es un centro de la NASA dedicado a la construcción y a la planificación de los vehículos y satélites no tripulados.

3.3.2. Access Mars

Access Mars es una aplicación creada por la NASA en colaboración con Google disponible para todos los dispositivos de escritorio, móviles y de realidad virtual. Esta aplicación es una adaptación del *software* OnSight del JPL y permite a los usuarios realizar una visita guiada del camino que siguió el rover Curiosity, o por el contrario, se puede pasear libremente por la superficie de Marte, sobre sus dunas y sus valles previamente explorados por el rover.

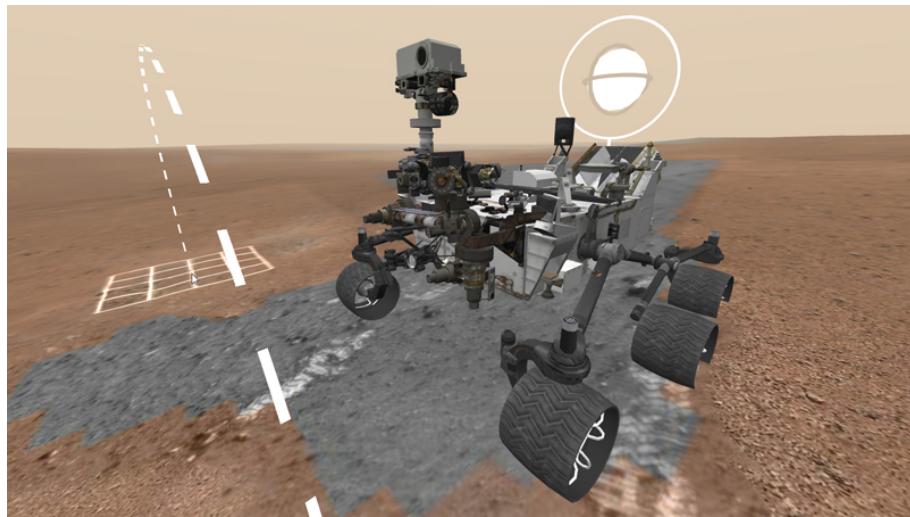


Figura 3.2: Captura de Acces Mars.

3.3.3. Mars Odyssey

Mars Odyssey es un videojuego de realidad virtual que permite al jugador experimentar cómo sería una misión tripulada en la superficie del planeta rojo [15]. La aplicación lleva al jugador a los lugares de aterrizaje de los rovers y las distintas sondas lanzadas por la NASA para realizar una determinada misión. Los modelos tridimensionales que se pueden ver en la aplicación guardan un parecido muy realista con los utilizados por la NASA. Este juego está disponible únicamente para plataformas HTC Vive y Oculus.



Figura 3.3: *Captura de Mars Odyssey.*

3.3.4. Experience Curiosity

Experience Curiosity es una aplicación de navegador WebGL donde el usuario puede aprender sobre el camino que recorrió en rover Curiosity en el cráter Gale en la región Pahrump Hills. Esta aplicación esta disponible de forma gratuita para los navegadores Google Chrome, Mozilla Firefox, Safari y Microsoft Internet Explorer [16].



Figura 3.4: *Captura de Experience Curiosity.*

3.4. Plataformas utilizadas

Para el desarrollo de la aplicación se ha utilizado el motor de juegos Unity por estar completamente integrado con las nuevas tecnologías de realidad virtual y por su compatibilidad con múltiples plataformas. Otro de los aspectos por los que Unity ha sido elegido como plataforma de desarrollo es por su rápido aprendizaje, ya que permite que personas con conocimientos de programación aprender de forma sencilla el funcionamiento del programa. Además, Unity cuenta con una amplia documentación y una comunidad muy extendida para poder resolver los posibles problemas que aparezcan en el desarrollo.

Como plataforma principal del desarrollo se ha utilizado Cardboard. Los motivos principales de esta elección son su independencia de *hardware* y su coste. Para desarrollar con Unity se ha utilizado la SDK de Google VR gratuita que hay para esta plataforma.

Para realizar una mejor experiencia inmersiva se ha decidido realizar la aplicación para Samsung Gear VR y Oculus. Estas plataformas tienen una fácil integración con Unity y emplean el mismo paquete de herramientas de desarrollo. De esta forma queda reflejada la capacidad de Unity para el desarrollo de aplicaciones en distintas plataformas. Además se obtiene una mejor calidad visual y experiencia de usuario. En el caso del HDM de Samsung Gear VR se tendrán los mismos grados de libertad y el mismo *field of view* que utilizando Cardboard, pero con la ventaja de tener una mejor resolución y la posibilidad de utilizar un mando. Por el contrario Oculus ofrece seis grados de libertad, un FOV superior y una mejor resolución que las plataformas mencionadas anteriormente.

4. DISEÑO DE LA SOLUCIÓN

En este capítulo se explica el proceso de diseño realizado, así como la documentación utilizada para su posterior implementación. Aplicando la metodología ágil Scrum, en esta fase de diseño se realiza el *backlog*, también se analizaran las historias de usuario detalladamente y se establece la estructura del proyecto.

4.1. Backlog

En el *backlog*, como se ha mencionado con anterioridad, se almacenan todas las funcionalidades y requisitos priorizando aquellas que tengan una prioridad más alta. Esta lista de requisitos está sujeta a cambios, pudiendo añadir nuevas funcionalidades en cualquier momento o prescindir de aquellas que se crean no convenientes de implementar.

Las historias de usuario presentes en el *backlog* estarán formadas por un identificador único para cada historia de usuario, un título, una breve descripción, una estimación de tiempo y las dependencias que pueda tener sobre otra historia. A continuación se expone la documentación realizada en la reunión de *sprint planning*, que refleja las historias de usuario de forma detallada y las tareas a realizar en cada una de ellas.

4.1.1. Sprint planning

1. La primera escena que ve el usuario es un menú principal donde puede acceder a las funcionalidades básicas de la aplicación.

La primera escena disponible para el usuario es la del menú principal, en ella podrá seleccionar el mapa y la misión, crear nuevas rutas o empezar la simulación. Se realizará una primera versión a modo de boceto para luego ir añadiendo las diferentes funcionalidades.

- **Diseñar menú:** Se creará la escena principal con el menú y se empezará a plantear su forma con una primera versión del mismo. Esta versión irá incorporando nuevas funcionalidades conforme avance el proyecto.

- **Editar skybox:** Añadir un nuevo *skybox* que sitúe al usuario en el espacio.
- **Modelo tridimensional de Marte:** Crear un modelo tridimensional de Marte a modo de decoración de la escena.

Coste	4h
Dependencias	

Cuadro 4.1: Coste y dependencias de la historia de usuario 1.

2. El usuario tendrá la posibilidad de crear nuevas rutas a través de la aplicación de ordenador y poder cancelar su creación.

El usuario podrá añadir una ruta al mapa que este seleccionado pulsando un botón desde el menú principal. Cuando pulse este botón se creará una pestaña desde la cual podrá indicar la posición de inicio y la posición destino de la misión que desee realizar. Cuando se expanda la pestaña aparecerá un botón en la parte superior para que cuando sea pulsado por el usuario se cancele la creación de la ruta y pueda volver al menú principal.

- **Crear menú de creación de nueva ruta:** Crear una pestaña que aparezca sobre el menú principal cuando se pulse el botón de creación de nueva ruta. En esta pestaña aparecerán cuatro campos de texto para indicar las coordenadas (x, y) de las posiciones de inicio y de destino. También aparecerá en esta interfaz un botón que inicie el proceso de creación de ruta.
- **Crear botón de cancelación de ruta:** Crear un botón que se encargue de cerrar la pestaña del menú de creación de rutas y restablezca la información del mapa y de la misión que estaban antes de abrir dicho menú.
- **Dibujar la nueva ruta sobre el mapa:** Crear un método que se encargue de dibujar las posiciones de inicio y destino sobre la imagen del mapa 2D.
- **Ejecutar aplicación de cálculo de rutas:** Crear los métodos necesarios para realizar las llamadas al proceso de cálculo de rutas en un segundo plano. Se espera que los métodos que se realicen tomen como parámetros las informaciones relativas al mapa y las posiciones de inicio y final de la ruta, y se genere un archivo de texto con los resultados de la aplicación de cálculo de rutas.

Coste	12h
Dependencias	Historia de usuario 1

Cuadro 4.2: Coste y dependencias de la historia de usuario 2.

3. El usuario podrá elegir la ruta que desee utilizar para la realización de la simulación de la misión, entre una variedad de mapas con varias rutas. Este requisito es independiente de la plataforma que este utilizando el usuario.

El usuario podrá seleccionar una misión desde el menú principal y ver la información de esta. Podrá ver el número de puntos por los que tiene que pasar, la distancia recorrida y el recorrido sobre un mapa del terreno en dos dimensiones. La aplicación tendrá cargada unas misiones de prueba para poder probar la aplicación sin necesidad de crear una nueva ruta.

- **Lectura de ficheros de salida de la aplicación de cálculo de rutas:** Se crearán los métodos necesarios para realizar la lectura de ficheros de texto con la información de las rutas. Esta información se transformará en un objeto ‘Mission’ dentro de la aplicación.
- **Lectura de ficheros de salida de la aplicación de cálculo de rutas para Android:** Se crearán los métodos necesarios para realizar la lectura de ficheros de texto con la información de las rutas en Android. Esta información se transformará en un objeto ‘Mission’ dentro de la aplicación.
- **Dibujar ruta en la imagen del mapa:** Cuando se selecciona el mapa se carga una imagen 2D del terreno, sobre esta imagen del terreno se dibujará la ruta de la misión indicando el punto de inicio, el punto final y los puntos intermedios.
- **Seleccionar misión:** Se realizarán los botones para cambiar entre las misiones. En el caso de la aplicación de escritorio se implementará un menú desplegable, donde aparezcan todas las misiones, y para la aplicación de Cardboard se diseñaran dos botones(‘misión siguiente’ y ‘misión anterior’) que interactúen con los controles del dispositivo.

- **Mostrar información de la misión:** En el menú principal se mostrará una pequeña información de la ruta de la misión.
- **Añadir rutas:** Se calcularán varias rutas precargadas para poder probar la aplicación sin necesidad de crear una ruta nueva. También se realizarán pruebas para comprobar que el proceso de creación haya sido correcto.

Coste	22h
Dependencias	Historia de usuario 1

Cuadro 4.3: Coste y dependencias de la historia de usuario 3.

4. El usuario podrá elegir el mapa que desee utilizar para la simulación de la misión.

En esta historia de usuario se crearán los *prefabs* de los mapas, así como los modelos tridimensionales de los mismos y se añadirán a la aplicación. Se crearán un total de 8 mapas.

- **Crear modelos tridimensionales de los mapas:** Se realizará la transformación de los ficheros DTM de los mapas para obtener un modelo tridimensional que sea compatible con Unity. Antes de importar estos modelos se pasarán por el programa Maya3D para reducir el número de polígonos y mejorar el rendimiento de la aplicación.
- **Crear los *prefabs* de los mapas en Unity:** Crear los objetos de los mapas y asignar la información de cada uno de ellos.
- **Mostrar la información de los mapas:** En el menú principal se mostrará la información del mapa seleccionado (Tamaño, nombre, imagen y nombre acortado).
- **Seleccionar mapa:** Se realizarán los botones para cambiar entre los mapas. En el caso de la aplicación de escritorio se implementará un menú desplegable, donde aparecen todos los mapas. Para la aplicación de Cardboard se diseñarán dos botones ('mapa siguiente' y 'mapa anterior') que interactúen con los controles del dispositivo.

Coste	12h
Dependencias	Historia de usuario 1

Cuadro 4.4: Coste y dependencias de la historia de usuario 4.

5. El usuario siempre tendrá la opción de simular la misión con realidad virtual.

Independientemente de la plataforma que utilice el usuario en el menú principal aparecerá un botón que le permita acceder a escena de simulación de realidad virtual. En esta historia de usuario también se gestionará el paso de información entre la escena principal y la escena de simulación.

- **Crear una gestor de cambio de escena:** Crear un *script* que se encargue de cambiar de una escena a otra.
- **Transición a realidad virtual:** En la aplicación de escritorio hay que realizar una transición para activar y desactivar la realidad virtual para realizar la simulación con Oculus.
- **Intercambiador de información entre escenas:** Se creará un elemento que permita intercambiar información de la escena del menú principal (mapa y ruta) a la escena de simulación de realidad virtual.
- **Montar la escena de simulación en realidad virtual:** Crear una nueva escena para la simulación en realidad virtual donde se cargue el mapa, la misión, el rover y el *player* del usuario.
- **Mejoras de rendimiento de escena y efectos de iluminación:** Se añadirán elementos para mejorar el realismo de las escena como una iluminación similar a la que podría haber en la superficie de Marte, un *skybox* con un tono rojizo que simule el cielo de Marte y un efecto de niebla. Este efecto de niebla cumple dos funciones: Mejorar el rendimiento para no renderizar todo el modelo del mapa y mejorar la apariencia visual.

Coste	24
Dependencias	Historia de usuario 3

Cuadro 4.5: Coste y dependencias de la historia de usuario 5.

6. El usuario podrá realizar la simulación en 3D sin necesidad de dispositivos de realidad virtual en la aplicación de escritorio.

En la aplicación de escritorio el usuario podrá realizar la simulación 3D de la misión sin utilizar dispositivos de realidad virtual. Para ello se montará un nueva escena de simulación con una cámara convencional.

- **Montar escena de simulación 3D:** Crear una nueva escena para la simulación 3D donde se cargue el mapa, la misión, el rover y el *player* del usuario.
- **Ajustar el intercambiador de escenas:** Aprovechar el intercambiador de escenas realizado en la historia de usuario 3 para esta nueva escena de simulación 3D.

Coste	6h
Dependencias	Historia de usuario 3

Cuadro 4.6: Coste y dependencias de la historia de usuario 6.

7. El usuario debe poder salir de la misión en cualquier momento.

Se debe realizar una interacción que permita al usuario salir de la escena de simulación y volver al menú principal.

- **Salir de la simulación:** Cuando el usuario pulse la tecla ‘Esc’ se producirá el cambio de escena entre la escena de simulación y la del menú principal.
- **Salir de la simulación de realidad virtual en la aplicación Cardboard:** Se construirá un botón tridimensional que aparecerá en el parte posterior del rover, que cuando el usuario apunte durante un periodo de tiempo se accionará devolviendo los a la escena del menú principal.

Coste	2h
Dependencias	Historia de usuario 5, 6

Cuadro 4.7: Coste y dependencias de la historia de usuario 7.

8. El usuario podrá caminar libremente por la superficie de Marte.

Se diseñará un sistema de teletransporte que funcione de la siguiente manera: El usuario apuntará con un láser que sale del controlador al punto que desea ir y en el caso de que esté dentro del rango de teletransporte se realizará el cambio de posición (Se utiliza un método de teletransporte para evitar mareos en el usuario).

- **Realizar controles de teletransporte:** Se utilizará un rayo que salga del mando virtual de la simulación y tras apretar el gatillo primario del mando derecho se realizará el intercambio de posición. Este movimiento estará permitido siempre que la distancia a la que se quiere mover esté dentro de un rango establecido por el programador, la luz del rayo indicará si está disponible el movimiento, en color verde, o si por el contrario no está disponible, en color rojo.
- **Realizar controles FPC para la simulación 3D de la aplicación de escritorio:**
Se realizaran unos controles *first person controller* (FPC) típicos de videojuegos (teclas W, A, S, D para realizar el movimiento por el espacio y el movimiento del ratón para controlar la cámara).

Coste	8h
Dependencias	Historia de usuario 5, 6

Cuadro 4.8: Coste y dependencias de la historia de usuario 8.

9. El usuario realizará la simulación posicionado en la parte superior de rover.

El *player* del usuario se posicionará sobre el rover de tal forma que se pueda realizar la simulación acompañado al modelo tridimensional del vehículo espacial. En las plataformas en las que esté disponible el libre movimiento por la superficie se creará una interacción para incorporar esta funcionalidad.

- **Seguimiento sobre el rover:** Vincular el movimiento del rover al del *player*.
- **Interacción para el seguimiento del rover:** Realizar una interacción que posicione al *player* sobre el rover o permita su libre movimiento por la superficie. En caso de que se deshabilite el movimiento libre por la superficie el láser permanecerá invisible hasta que se vuelva a habilitar. Para realizar esta interacción se utilizará uno de los botones del controlador del mando del dispositivo de realidad virtual.

Coste	6h
Dependencias	Historia de usuario 3

Cuadro 4.9: Coste y dependencias de la historia de usuario 9.

10. La simulación contará con un modelo de baja resolución que emule los movimientos del rover de sobre la superficie de Marte.

Se creará un modelo tridimensional del rover Curiosity con un número reducido de vértices para mejorar el rendimiento de la aplicación, se añadirán físicas al modelo y se programará su comportamiento dentro de la simulación.

- **Modelado del rover:** Encontrar un modelo de rover y reducir los vértices para optimizar el rendimiento. Despiecear cada componente del rover para luego establecer un sistema de físicas.
- **Añadir físicas al modelo:** Aplicar un sistema de físicas que simulen la suspensión y los movimientos del rover.
- **Programar el comportamiento del rover:** El rover deberá seguir la ruta marcada por las misiones y cuando la complete deberá permanecer inmóvil.

Coste	18h
Dependencias	Historia de usuario 5, 6

Cuadro 4.10: Coste y dependencias de la historia de usuario 10.

11. El usuario podrá elegir mapa, misión y realizar la simulación desde la plataforma Samsung GearVR.

Esta historia de usuario se añadió durante el desarrollo del proyecto. En esta historia se realizará un nuevo proyecto de Unity para compilar la aplicación para la plataforma de Samsung GearVR.

- **Montar escena de menú principal:** Se creará una nueva escena con el menú principal con las mismas características que las realizadas para las aplicaciones de escritorio y Cardboard.
- **Interacciones menú principal:** Se empleará un láser para seleccionar las funcionalidades del menú y se utilizará el gatillo principal del controlador para accionar los elementos interactivos. Se indicarán que elementos son interactivos dependiendo del color del láser, si es interactivo será verde, en caso contrario será rojo.
- **Interacciones escena de simulación:** En esta tarea se realizarán las interacciones para la escena de simulación, estas interacciones serán muy similares a las realizadas para Oculus.

Coste	12h
Dependencias	Historia de usuario 8, 9, 10

Cuadro 4.11: Coste y dependencias de la historia de usuario 11.

4.2. Organización del equipo

En este apartado se presenta el equipo necesario para gestionar y desarrollar el proyecto. Para explicar los distintos roles que participan en este modelo de metodología ágil se van a presentar en dos categorías diferentes.

Primero los roles que están comprometidos con el proyecto y con la metodología Scrum:

- **Scrum Master:** Se encarga de hacer que funcione la metodología dentro del proyecto. Trata de eliminar todos los elementos bloqueantes que hacen que el proyecto no fluya y así garantizar el cumplimiento de los hitos.
- **Equipo de desarrollo:** Son los encargados de diseñar e implementar las aplicaciones. También tienen autoridad para tomar decisiones y para estimar el coste de las tareas del *backlog*.
- **Equipo de modelado:** Son los encargados de modelar y diseñar los elementos visuales de la escena.
- **Product Owner:** Es la persona que conoce el negocio del cliente y su visión del producto.

Por último las personas que no forma parte del proceso Scrum pero son necesarias para comprobar el proceso de desarrollo.

- **Clientes:** Son las personas que demandan el producto y expresan la situación del problema que quieren resolver.
- **Usuarios:** Destinatarios finales del producto. Participan en las fases de pruebas para corregir posibles errores en las funcionalidades.

5. IMPLEMENTACIÓN

En este capítulo se explica detalladamente el proceso de implementación del proyecto siguiendo las fases de análisis y diseño de los capítulos anteriores.

5.1. Etapas de desarrollo

Aplicando la metodología ágil Scrum, la implementación del proyecto se va dividir en varias iteraciones o *sprints*. Cada *sprint* se centra en una o varias historias de usuario y su duración estimada es de veinte horas para cada semana, realizándose un *sprint review* una vez a la semana coincidiendo con el fin del *sprint*.

Para cada *sprint* se diseña una lista de tareas o *Taskboard*. En esta lista de tareas se ponen los objetivos marcados para completar el *sprint* y al lado de cada objetivo las tareas necesarias para completarlo. Estas tareas se representan habitualmente en forma de *post-it*s que se van moviendo hacia la derecha indicando en cada columna si está pendiente de ser iniciada, en progreso o terminada.



Figura 5.1: Captura del taskboard del sprint 5.

Sprint 1

El *sprint* 1 se va a centrar en el diseño de la primera versión del menú principal y en establecer las conexiones con la aplicación de cálculo de rutas.

Historia de usuario	Tarea	T.Estimado	T.Real
Historia de usuario 1	Diseñar menú	2h	1h
Historia de usuario 1	Editar <i>skybox</i>	1h	1h
Historia de usuario 1	Modelado 3D de Marte	1h	2h
Historia de usuario 2	Menú nueva ruta	2h	2h
Historia de usuario 2	Cancelar ruta	2h	1h
Historia de usuario 2	Ejecutar cálculo de rutas	8h	12h
Historia de usuario 3	Lectura de ficheros	4h	6h

Cuadro 5.1: Tareas para el Sprint 1.

Revisión de Sprint 1

La estimación de horas no ha sido realmente buena, para poder completar todas las tareas del *sprint* se han necesitado más horas de las que estaban estimadas. Uno de los factores más determinantes que han condicionado la necesidad de dedicar más horas de las esperadas en este *sprint*, ha sido la conexión con la aplicación de cálculo de rutas por los siguientes motivos:

- La mala gestión de los hilos y de los procesos externos en Unity.
- No se ha podido implementar una solución basada en corrutinas que funcionase correctamente.
- Para ejecutar la aplicación de cálculo de rutas y el editor de Unity se necesita reservar espacio de memoria RAM muy amplio y esto ha supuesto un inconveniente.

Finalmente se optó por una solución basada en un hilo secundario que ejecuta el proceso de cálculo de rutas en un segundo plano de forma muy simplificada para evitar problemas con el hilo principal del motor de Unity. En menor medida el *sprint* también se

vio afectado por una mala estimación del tiempo necesario para implementar la lectura de ficheros desde la aplicación de escritorio y por añadir efectos de partículas al modelo 3D de Marte, ya que esto último no estaba reflejado en la planificación del *sprint*.

Sprint 2

Este segundo *sprint* se enfoca en terminar la lectura de ficheros de texto de la aplicación de rutas y mostrar la información de las misiones sobre el menú principal.

Historia de usuario	Tarea	T.Estimado	T.Real
Historia de usuario 2	Dibujar nueva ruta	2h	1h
Historia de usuario 3	Lectura de ficheros Andorid	4h	4h
Historia de usuario 3	Dibujar ruta	4h	4h
Historia de usuario 3	Seleccionar misión	4h	3h
Historia de usuario 3	Mostrar información misión	2h	1h
Historia de usuario 3	Añadir rutas	4h	4h

Cuadro 5.2: Tareas para el Sprint 2.

Revisión de Sprint 2

Teniendo en cuenta la mala estimación del primer *sprint*, esta vez se ha optado por una optimización pesimista de los tiempos de desarrollo. Ha resultado ser una estimación más precisa en comparación con la anterior. Como resultado de esta iteración se ha obtenido una primera versión de la aplicación en la cual se pueden crear nuevas rutas, seleccionar las rutas creadas y ver la información de las misiones sobre el menú principal.

Sprint 3

Este tercer *sprint* se centrará en la creación de los modelos tridimensionales de los mapas y en su importación a Unity. También se empezará a desarrollar el cambio entre la escena principal y la escena de simulación en realidad virtual.

Revisión de Sprint 3

Historia de usuario	Tarea	T.Estimado	T.Real
Historia de usuario 4	Crear modelos 3D de los mapas	4h	4h
Historia de usuario 4	Crear <i>prefabs</i> de los mapas	4h	6h
Historia de usuario 4	Mostrar información mapas	2h	2h
Historia de usuario 4	Seleccionar mapas	2h	1h
Historia de usuario 5	Crear gestor cambio de escenas	4h	3h
Historia de usuario 5	Intercambiador de información	4h	5h
Historia de usuario 5	Montar escena de simulación VR	4h	7h

Cuadro 5.3: Tareas para el Sprint 3.

La estimación de horas para este *sprint* ha sido una estimación optimista, aunque en general se podría decir que no es del todo mala. El principal inconveniente de este *sprint* ha sido el montaje de la escena, las causas de este sobrecoste de tiempo han sido la experimentación con los distintos efectos visuales y ajustes de iluminación.

Como resultado de esta iteración se dispone de una aplicación que no solo permite crear y seleccionar rutas de un determinado mapa, sino que también permite el cambio entre escenas para empezar la simulación.

Sprint 4

Este *sprint* está enfocado en terminar las funcionalidades del menú principal y el intercambio entre escenas, tanto de la aplicación de escritorio como la de Cardboard. Se realizarán también mejoras de rendimiento para conseguir una mejor fluidez en la escena de simulación.

Revisión de Sprint 4

Como resultado de esta iteración la aplicación cuenta todas las funcionalidades relativas al menú principal y al cambio de escenas.

Sprint 5

Este *sprint* se centra en la escena de simulación, se creará y se importará el modelo de

Historia de usuario	Tarea	T.Estimado	T.Real
Historia de usuario 5	Transición a VR	4h	2h
Historia de usuario 5	Mejoras de rendimiento escena	8h	8h
Historia de usuario 6	Montar escena 3D	4h	4h
Historia de usuario 6	Ajustar intercambiador de escenas	2h	1h
Historia de usuario 7	Salir de la simulación escritorio	1h	1h
Historia de usuario 7	Salir de la simulación Cardboard	1h	2h

Cuadro 5.4: Tareas para el Sprint 4.

rover para que empiece a recorrer las rutas, y se añadirán los controles para desplazarse por el entorno de simulación.

Historia de usuario	Tarea	T.Estimado	T.Real
Historia de usuario 8	Controles teletransporte VR	4h	4h
Historia de usuario 8	Controles FPC	4h	2h
Historia de usuario 10	Modelado del rover	8h	12h
Historia de usuario 10	Comportamiento del rover	2h	2h

Cuadro 5.5: Tareas para el Sprint 5.

Revisión de Sprint 5

Debido a la falta de conocimientos sobre cómo modelar el rover, la estimación de tiempos ha sido bastante mala. Después de realizar estas tareas se dispone de un modelo *low poly*²² de rover capaz de recorrer las rutas sobre los mapas.

Sprint 6

Este *sprint* es el más corto, en él se van a implementar las interacciones de seguimiento del rover y se van a añadir las físicas al modelo para mejorar el realismo de sus movimientos.

²²Este término se emplea para definir una malla poligonal con un número bajo de polígonos.

Historia de usuario	Tarea	T.Estimado	T.Real
Historia de usuario 9	Seguimiento del rover	4h	4h
Historia de usuario 9	Interacción seguimiento del rover	2h	2h
Historia de usuario 10	Añadir físicas al modelo	8h	10h

Cuadro 5.6: Tareas para el Sprint 6.

Revisión de Sprint 6

El peso de la tarea de añadir las físicas se estimó por lo alto debido a que iba a ser una tarea complicada y aun así la tarea llevó más tiempo. Tras finalizar los 6 *sprints* se obtiene la versión final del proyecto, la aplicación para escritorio que incorpora la plataforma de Oculus y la aplicación de Cardboard.

Sprint 7

Durante el desarrollo del proyecto se presentó la posibilidad de realizar la aplicación para la plataforma de Samsung GearVR. Esto es una opción interesante ya que compilar la aplicación para esta plataforma requiere unos cambios mínimos. En este *sprint* se realizarán estos cambios y se dará por concluida la fase de implementación.

Historia de usuario	Tarea	T.Estimado	T.Real
Historia de usuario 11	Montar escena menú	4h	4h
Historia de usuario 11	Interacción menú principal	4h	2h
Historia de usuario 11	Interacción en la simulación	4h	4h

Cuadro 5.7: Tareas para el Sprint 7.

Revisión de Sprint 7

La implementación durante este *sprint* fue bastante sencilla gracias a la reutilización de código de la aplicación de escritorio. Las plataformas de Oculus y Samsung GearVR utilizan las mismas herramientas de desarrollo por lo que los cambios fueron mínimos. La mayoría del tiempo destinado para la realización de esta iteración se utilizó para realizar

pruebas y comprobar el funcionamiento de la aplicación.

5.2. Diseño de escenas

Los proyectos realizados con Unity están divididos en escenas, cada escena contiene todos los objetos de la aplicación. A continuación se tratan todos los elementos que han intervenido en el diseño y el montaje de las escenas del proyecto.

5.2.1. Menú aplicación de escritorio

La aplicación de escritorio tiene la característica de que algunas de sus escenas son en realidad virtual y otras no. La escena del menú principal no está diseñada para realidad virtual por lo que los controles que se utilizan son el teclado y el ratón. Por lo tanto se utilizan los elementos convencionales de interacción de un menú, como botones o desplegables.



Figura 5.2: Captura del menú de la aplicación de escritorio.

5.2.2. Menú en realidad virtual

En la aplicación de Samsung GearVR se utiliza el mando como medio de interacción con el sistema. Se utiliza un láser, que sale del mando virtual, para indicar el botón con el cual se quiere interactuar. Si el láser se torna de un color verde quiere decir que ese objeto de la escena tiene interacciones disponibles, en caso contrario el láser se volverá de color rojo. Esta interacción es muy similar a la que se utiliza en la escena de simulación de esta misma aplicación.

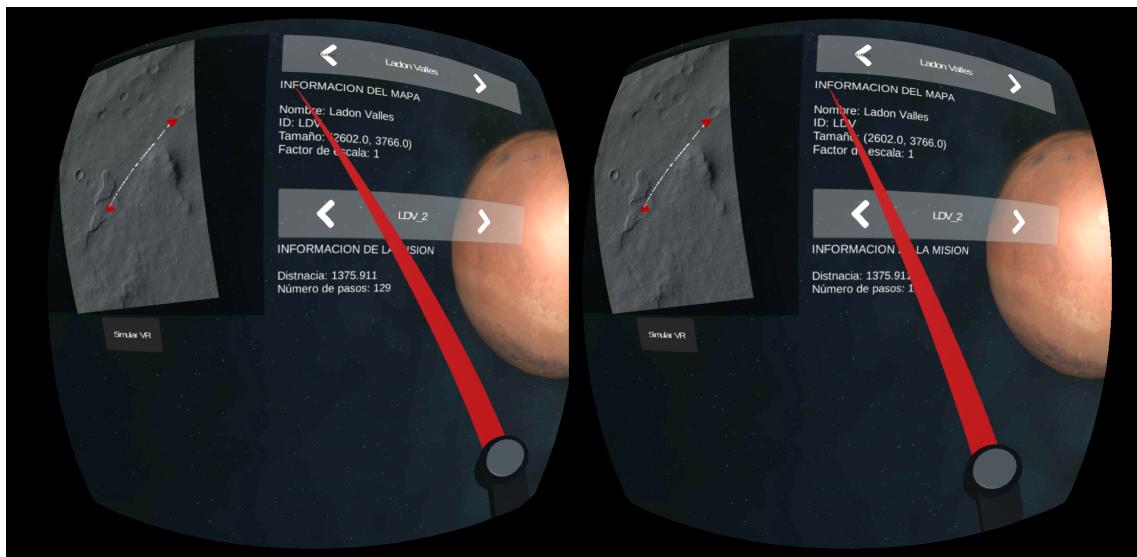


Figura 5.3: Captura del menú de la aplicación de Samsung GearVR.

Para la aplicación de Cardboard, no necesita de controladores por lo que las interacciones serán totalmente diferentes a las descritas anteriormente. El usuario apuntará con un puntero, situado en el centro de la pantalla, al botón con el que quiera interactuar. Una vez coincidan el puntero y el botón se iniciará un temporizador que cuando complete su espera accionará la interacción.

5.2.3. Efectos visuales del menú

Cabe destacar los principales efectos visuales utilizados en la escena del menú principal, el skybox y el sistema de partículas. Los sistemas de partículas son elementos muy complejos utilizados para simular una gran variedad de efectos como el fuego, agua, humo, polvo, etc. En el caso mencionado, se ha utilizado los efectos de partículas para

simular la atmósfera que recubre el planeta rojo. Para ello el sistema genera partículas en torno a una forma semi esférica que desaparecen después de un periodo corto de tiempo. Cada partícula es una imagen de humo blanco de tamaño reducido que en conjunto forma un efecto de niebla sobre la superficie del planeta. Para la elaboración del *skybox* se ha utilizado una técnica mediante la cual 6 imágenes forman un cubo cerrado y crea la sensación de estar inmerso en espacio. Estos efectos son únicamente decorativos y se emplean para mejorar la experiencia del usuario.

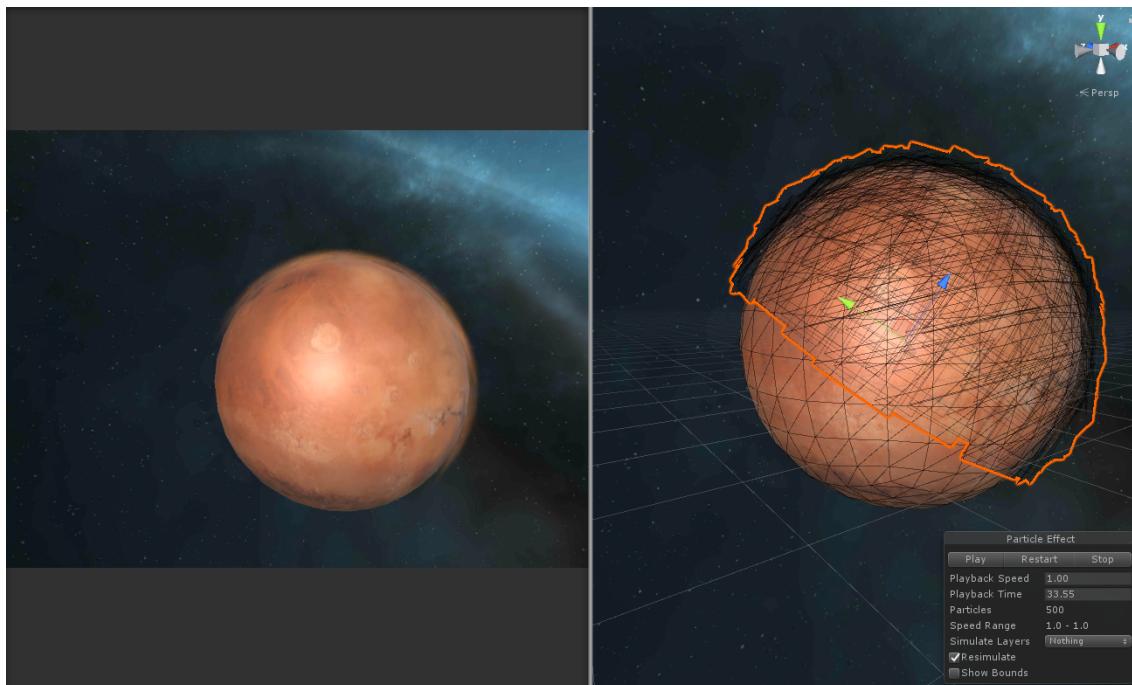


Figura 5.4: *Captura del editor de Unity en el proceso de incorporación de efectos de partículas para el modelo de Marte.*

5.2.4. Proceso de incorporación de los modelos en la escena

Tanto los mapas tridimensionales como el modelo virtual del rover han sufrido modificaciones antes de ser importadas en Unity, por motivos de optimización de rendimiento. En este apartado se explican los procesos necesarios para obtener los modelos que se incluyen en la escena de simulación del proyecto y los motivos por los cuales se hacen estos cambios.

5.2.4.1. Modelo del rover

La NASA facilita en su página web²³ un modelo de rover con un alto nivel de detalle y gran calidad. Este modelo tiene una gran cantidad de polígonos, a priori, no debería suponer ningún problema, pero en caso de utilizar esta malla para un aplicación de realidad virtual en un dispositivo Cardboard, el modelo no sirve. Para ajustar este modelo a las condiciones del proyecto se utilizó la aplicación Autodesk Maya para eliminar elementos decorativos del modelo original y reducir su número de polígonos de forma significativa. Esta aplicación de modelado se utilizó también para trocear el modelo, se separaron todas las partes móviles del rover para después juntarlas en Unity utilizando físicas.

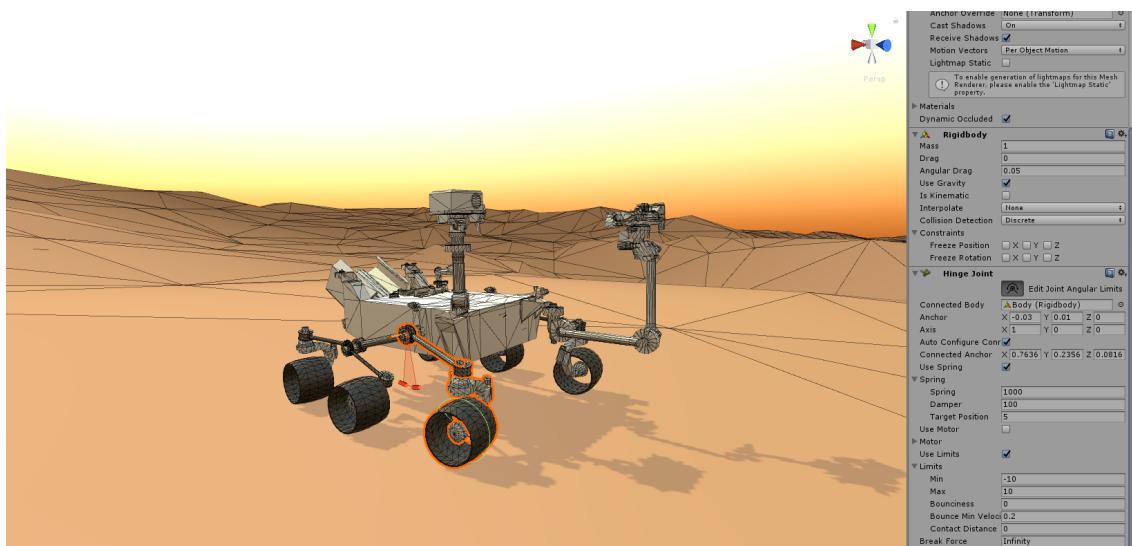


Figura 5.5: *Captura del editor de Unity en el proceso de incorporación de físicas del rover.*

La malla 3D del rover resultante del proceso mencionado anterior, al tratarse de un archivo con extensión ‘.fbx’, se puede incorporar directamente a Unity. Dentro de Unity se añade el modelo a la escena. En este punto de la incorporación del rover en la aplicación, el objeto que emula el rover no tiene ninguna propiedad, es un elemento intangible que no se ve afectado por ninguna fuerza física, únicamente es visible. Para dotar de peso y por tanto, hacer que el objeto se vea afectado por fuerzas físicas, como la gravedad, se añade a cada pieza del modelo un componente *Rigid Body* o cuerpo rígido. Para hacer tangible el objeto y empiece a generar colisiones con el entorno se le añade un componente *Mesh*

²³Disponible en: <https://nasa3d.arc.nasa.gov/detail/mars-rover-curiosity>

Collider que envuelve la malla tridimensional en otra malla simplificada que determina el espacio que ocupa dentro del entorno virtual. No es necesario que todas las piezas tengan un componente *collider*, por lo que solo se añadirá este componente en las partes que van a colisionar. Tampoco es necesario que la malla de colisión se ajuste exactamente a la malla que define la forma del modelo, por ello se ha utilizado la técnica *convex hull* que simplifica la malla de colisión y mejora su rendimiento. Llegado este punto se utilizaron los componentes del motor de físicas de Unity *Hinge Joint* y *Wheel Collider* para realizar un sistema de amortiguación del rover y realizar el movimiento de las ruedas respectivamente. Los *Hinge Joint* se utilizan para simular el comportamiento de un resorte uniendo dos piezas, mientras que los *Wheel Collider* se encargan del movimiento y la potencia del motor del rover.

5.2.4.2. Modelos de los mapas

El proceso para la importación de los terrenos a la escena ha sido igual para todos ellos. En primer lugar se ha utilizado la herramienta para el programa de modelado Blender facilitada por el HiRISE. Esta herramienta transforma la matriz con la información del mapa, en formato DMT, en una malla de vértices que se puede exportar a Unity u otros programas de modelado. En este caso, se exportará en formato ‘.fbx’ para tratar la malla de vértices en el programa de modelado Autodesk Maya. Desde este programa se realizará la reducción de polígonos (similar a la aplicada en el proceso de reducción de polígonos del modelo del rover), independiente del modelo del mapa y su tamaño, todos los mallas de los mapas han sido reducidas hasta tener una número aproximado de 80.000 triángulos.

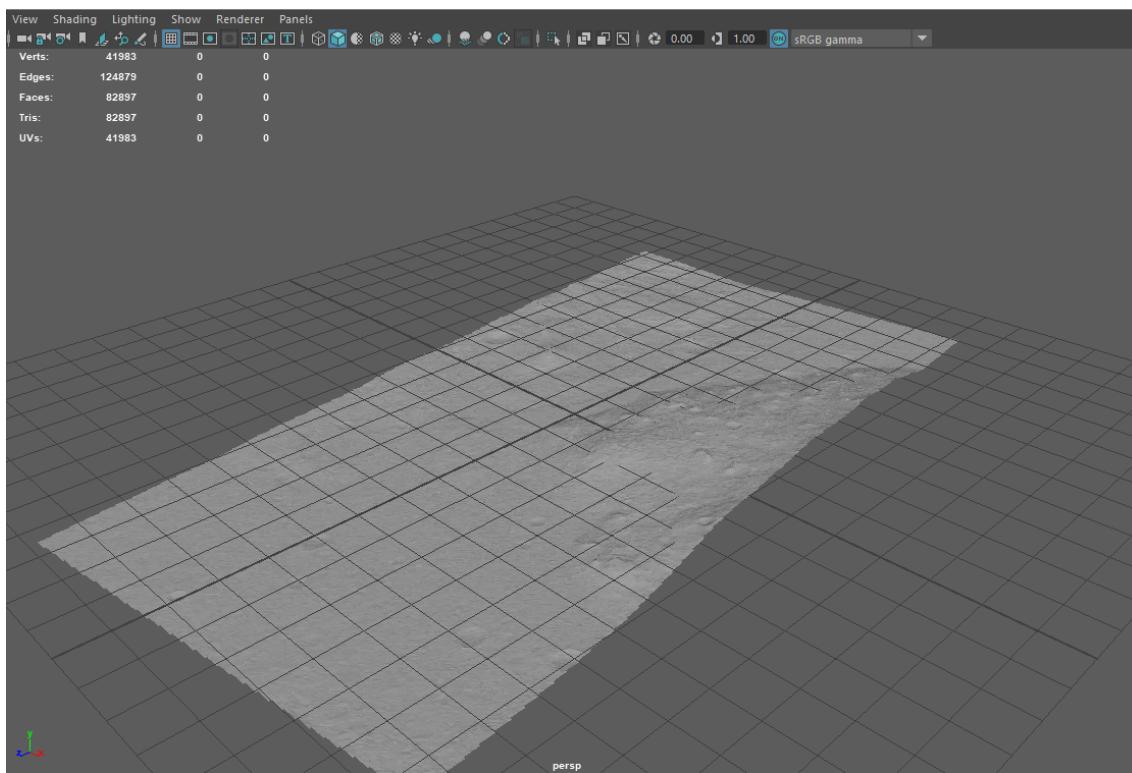


Figura 5.6: Captura de la aplicación Autodesk Maya.

Una vez completado el proceso de reducción de polígonos, el terreno se añade la escena de Unity. Los terrenos a diferencia del modelo de rover no necesitan tener un comportamiento de cuerpo rígido que se vea afectado por las físicas, únicamente necesitan gestionar las colisiones con el resto de objetos. Para añadir colisiones se utilizó un componente *Mesh Collider* igual que con el modelo de rover, pero en esta ocasión, si era necesario que la malla de colisión y la malla de vértices coincidieran de manera precisa.

5.2.5. Diseño de simulación 3D

La escena de simulación 3D al igual que el resto de escenas de simulación cuentan con tres elementos principales: el modelo del rover, el terreno y el *player*. La simulación 3D en particular la opción del utilizar un *player* con unos controles en primera persona se sustituyó por una cámara que pivota sobre el modelo rover para realizar una simulación más cómoda.

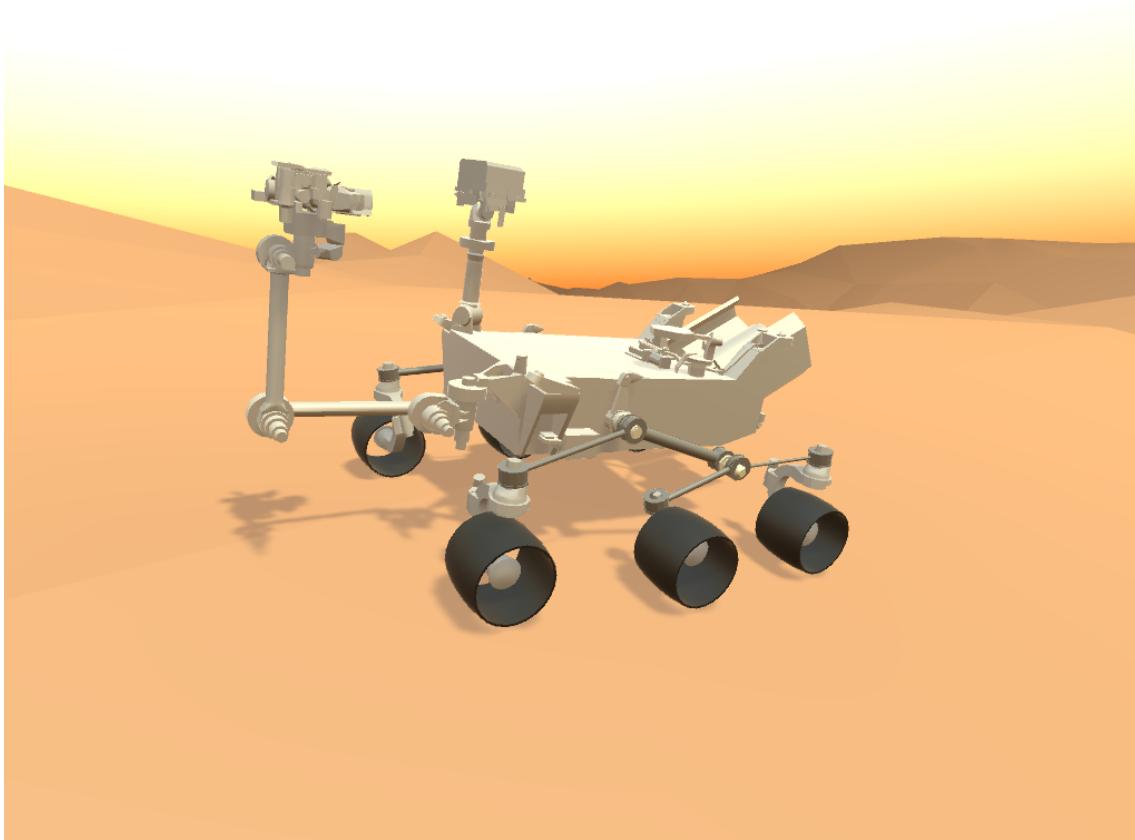


Figura 5.7: Captura de la escena de simulación 3D en la aplicación de escritorio.

Las interacciones disponibles dentro de la simulación 3D son las siguientes:

- **Q:** Inicia/Detiene el movimiento del rover.
- **A, D:** Rotan la cámara entorno al rover en el plano horizontal.
- **W, S:** Elevan o bajan la posición de la cámara.
- **E, R:** Aplican un efecto de *zoom in* o *zoom out* respectivamente.

5.2.6. Diseño de simulación inmersiva

Para las aplicaciones de Samsung GearVR y Oculus se ha implementado un sistema de teletransporte que permite al *player* moverse por la escena de forma libre. El usuario tiene un láser que se vuelve de color verde si la distancia de teletransporte es la adecuada y pulsando el gatillo principal realiza el intercambio de posición. Esta forma de movimiento es muy común en las aplicaciones de realidad virtual y se utiliza para evitar causar sensación de mareo.

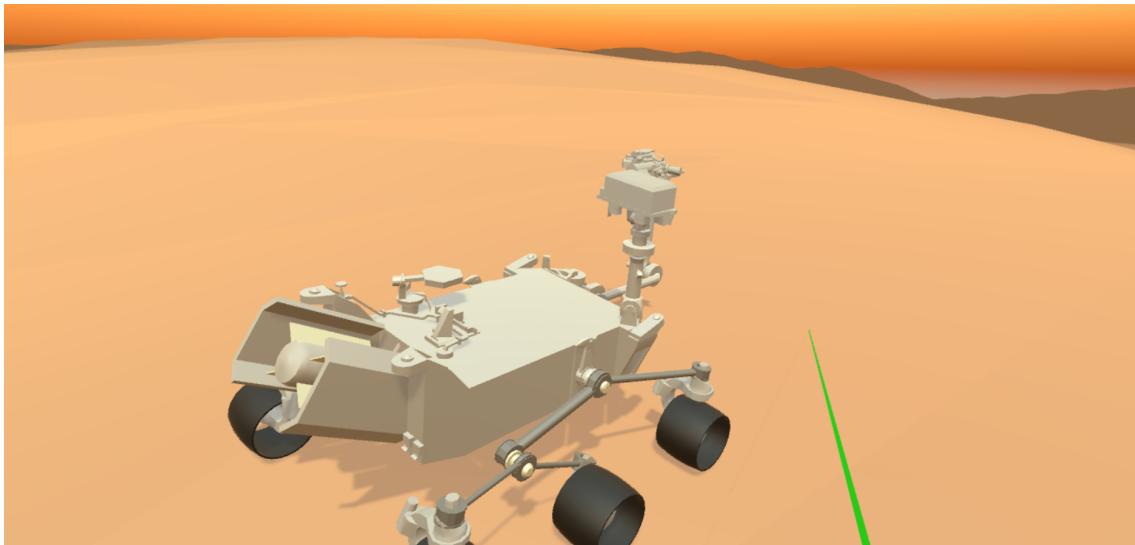


Figura 5.8: *Captura de la escena de simulación de realidad virtual en la aplicación de escritorio utilizando Oculus.*

En caso de que el usuario pierda de vista el rover o quiera posicionarse sobre él, podrá hacerlo pulsando uno de los botones del controlador. Esta opción deshabilita la posibilidad de teletransporte hasta que se vuelva a pulsar el mismo botón.

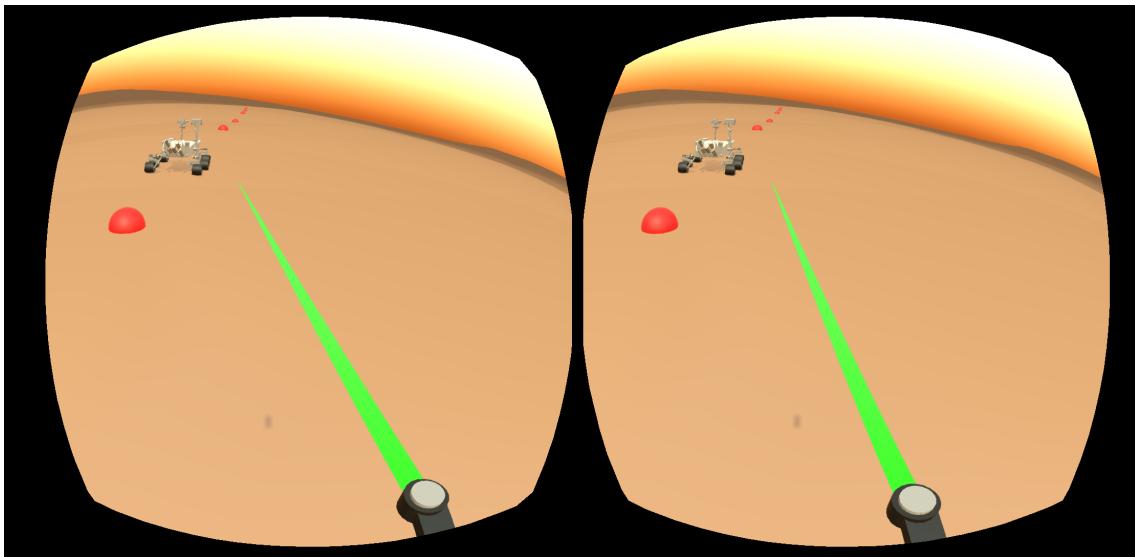


Figura 5.9: *Captura de la escena de simulación en la aplicación de Samsung GearVR.*

La aplicación de Cardboard no está diseñada para utilizar ningún tipo de controlador por lo que únicamente podrá realizar la simulación situado sobre el vehículo de exploración marciana.

5.3. Experiencia de usuario

La experiencia de usuario o *user experience* (UX) hace referencia a las sensaciones que tiene una persona al interactuar con un sistema. En el caso de las tecnologías inmersivas aún falta por encontrar una forma más humana para comunicarse con el entorno artificial [17]. Esto hace que las interacciones de las aplicaciones de VR sean muy dispares entre sí. Continuamente se están desarrollando nuevas formas de comunicación, control por voz, *tracking* de manos, guantes hápticos o nuevos diseños de controladores.

Para la experiencia de usuario dentro de las aplicaciones realizadas en este proyecto se han utilizado dos formas distintas de interactuar. En el caso de las aplicaciones para las plataformas Oculus y Samsung GearVR se han utilizado sus propios controladores, en caso de Cardboard se ha utilizado un puntero que se dirige con la el movimiento de la cabeza.

6. CONCLUSIONES

6.1. Objetivos cumplidos

A continuación se exponen todos los objetivos que se marcaron al inicio del proyecto y aquellos objetivos que surgieron durante el desarrollo del proyecto.

1. Realizar una adaptación de la aplicación encargada de la planificación de rutas a la nueva aplicación encargada de la simulación en realidad virtual.
2. Transformar los mapas en formato DTM en un modelo tridimensional.
3. Crear un modelo de vehículo de exploración espacial capaz de simular las operaciones que realizaría el rover en la superficie marciana.
4. Diseñar una solución software que permita la telemetría, y que esta sea representada dentro del entorno virtual.
5. Proporcionar un entorno amigable con el usuario, desde el cual pueda hacer uso de todas las funcionalidades de la aplicación.
6. Modificar los componentes proporcionados por Google VR SDK para ajustarlos a las necesidades de la aplicación.

El términos generales los objetivos han sido cumplidos satisfactoriamente, con la excepción del punto número cuatro. La telemetría se ha tratado de forma muy superficial debido a la baja resolución de los modelos que reflejaban aproximaciones muy poco precisas; este punto se trata en el apartado de líneas futuras. Además de estos objetivos, conforme avanzaba el proyecto se añadieron otros nuevos para mejorar la calidad de la solución y la experiencia del usuario. A continuación se exponen los objetivos cumplimentados que surgieron durante el proceso de diseño.

1. Aplicar una metodología ágil para el desarrollo del proyecto.
2. Añadir la funcionalidad de simular la aplicación de escritorio con un HMD de Oculus en un entorno inmersivo.

3. Añadir físicas al modelo tridimensional del rover para realizar una simulación más realista de la misión.
4. Compilar la aplicación para la plataforma Samsung GearVR modificando la solución de Oculus.

6.2. Lineas futuras

En este apartado se van a exponer las posibles modificaciones que se pueden realizar para mejorar la solución presentada. La intención de los cambios que se exponen a continuación es mejorar la calidad del *software* si el proyecto se hubiera realizado sin limitaciones de tiempo, *hardware* y presupuesto.

6.2.1. Mejoras de rendimiento

En primer lugar, para optimizar la gestión de recursos del ordenador se implementará el algoritmo de *path planning* en C# para que no sea necesario realizar la ejecución del proceso con hilos, sino incorporarlo directamente dentro del proyecto de Unity. De esta manera, Unity se encargará de la gestión de los recursos del ordenador y se tendrá un mayor control sobre el proceso de cálculo de rutas.

Diseñar una solución para realizar la simulación de forma más fluida sin perder la experiencia de usuario. Este punto queda abierto, ya que existen muchas posibles soluciones, que van desde realizar una carga dinámica e inteligente del terreno a utilizar efectos especiales de niebla para evitar el renderizado total de los elementos de la escena. A pesar de que estas mejoras aumentarían el rendimiento de la aplicación, necesitan de un *hardware* capaz de soportar la carga de trabajo que supondría. Es decir, un dispositivo móvil Carboard no sería capaz de realizar una simulación visualmente buena con un rendimiento óptimo o se vería muy limitado, para ello se necesitarían dispositivos más potentes.

Otro de los aspectos que queda pendiente para la optimización del proyecto es iluminación de las escenas, debido al diseño del proyecto no se ha podido realizar una iluminación estática, la cual hubiera mejorado la tasa de *frames* por segundo. Los cambios necesarios para añadir este tipo de iluminación hubieran llevado un gran coste de tiempo y para las condiciones de la aplicación tampoco habrían supuesto un amplio beneficio.

6.2.2. Mejoras de realismo

Uno de los inconvenientes de utilizar la realidad virtual es la dependencia de un *hardware* de alto rendimiento, en este aspecto el equipo que utilice la aplicación final debe contar con un ordenador que cumpla estas características si quiere realizar simulaciones más precisas de las misiones. Un ordenador de alto rendimiento, también sería útil el desarrollo y para poder crear modelos tridimensionales de los mapas de alta calidad. Los mapas que se han incorporado dentro de las aplicaciones tienen una precisión del 10 % del total que se puede llegar a obtener.

Debido a la intención de mejorar el rendimiento por las limitaciones del *hardware* el modelo del rover tiene una resolución muy baja y no se ajusta en detalle a un rover real. Contando con un equipo de modelado profesional se podría realizar un modelado de calidad, con animaciones y un sistema de huesos que se ajuste de forma precisa a los movimientos que pueda realizar un rover. Añadir efectos de partículas como polvo en las ruedas cuando se mueva por la superficie, agregar texturas al modelo o mejorar sus físicas conociendo las limitaciones, el peso y la potencia del rover original, son otras de las tareas que conseguirían que la aplicación se conviertan en una aplicación de alta precisión de simulación de misiones sobre la superficie de Marte. En estas circunstancias tendría sentido la realización de un sistema de telemetría, ya que los resultados obtenidos se acercarían a los que se dan en la realidad.

6.3. Conclusiones del proyecto

En este apartado se presentan las conclusiones personales de lo que ha supuesto la realización del proyecto y las conclusiones que se pueden obtener después de aplicar la metodología Scrum para el diseño y la implementación de la solución.

6.3.1. Burn down chart

La gráfica de avance o *burn down chart* representa el esfuerzo que queda por realizar frente al tiempo. Esta gráfica permite conocer el estado actual del proyecto y poder predecir cuando va a finalizar, por este motivo el *burn down chart*, es uno de los componentes críticos dentro de la metodología Scrum. La gráfica se actualiza a diario por lo que refleja

de forma muy precisa el estado del proyecto, es decir, después de cada jornada los desarrolladores estiman el coste de las tareas que les quedan por realizar en el *sprint* y esta información se añade a la gráfica. Es tarea del Scrum Master dirigir al resto del equipo para acercar la recta real (en color azul) a la recta ideal (en color rojo).



Figura 6.1: Gráfica de avance o burn down chart del proyecto.

La figura 6.1 refleja el proceso de desarrollo del proyecto. La gráfica representa el trabajo que queda por realizar en horas frente al tiempo en días. A simple vista se puede apreciar que el tiempo de duración estimado del proyecto es menor que el tiempo real, también se puede apreciar un pequeño pico de trabajo en los últimos días. Este último pico refleja el trabajo realizado en el *sprint* 7, el cual no estaba planificado en el inicio del desarrollo. Por lo tanto, si se deja al margen el último *sprint* la estimación de duración del proyecto se desvía únicamente 5 días, en total 20 horas. Otra de las conclusiones que se pueden sacar viendo la gráfica de avance es que, el coste en horas de las tareas de cada *sprint* ha sido subestimado en general, por lo que las predicciones realizadas han sido demasiado optimistas.

6.3.2. Conclusiones personales

Las conclusiones personales que se han obtenido una vez se ha finalizado el proyecto se resumen los siguientes puntos:

- Una satisfacción personal por haber realizado un proyecto de estas características y

ser capaz de aplicar los conocimientos aprendidos en el grado para diseñar, construir e implementar soluciones con tecnologías de realidad virtual.

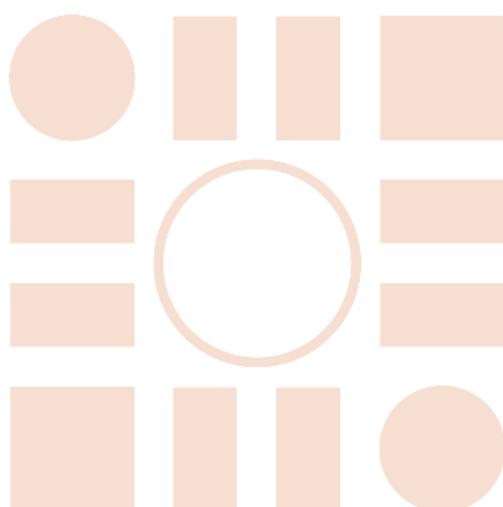
- Ampliar los conocimientos en el campo de la informática utilizando motores gráficos como Unity.
- Aprender conocimientos básicos sobre modelado, animación, iluminación, físicas y realización de efectos con partículas.
- Ampliar los horizontes del conocimiento para encaminar mi vida profesional después de concluir mis estudios.

BIBLIOGRAFÍA

- [1] Statista. (mayo de 2018). Forecast augmented (AR) and virtual reality (VR) market size worldwide from 2016 to 2022 (in billion U.S. dollars), [Online]. Disponible en: <https://www.statista.com/statistics/591181/global-augmented-virtual-reality-market-size/>.
- [2] Hector Costello. (abr. de 2017). Global Virtual Reality Market Forecast 2020 by Major Players such as Sony, Microsoft, Facebook, HTC, Google, Samsung Electronics, GoPro, etc, [Online]. Disponible en: <https://www.reuters.com/brandfeatures/venture-capital/article?id=4975>.
- [3] M. R.-M. P.Muñoz B.Castaño, “3Dana: A Path Planning Algorithm for Surface Robotics”, *International Scientific Journal of Engineering Applications of Artificial Intelligence*, vol. 60, pp. 175-192, 2017.
- [4] University of Arizona. (n.d.). ABOUT US: Principal & Co-Investigators, [Online]. Disponible en: <https://www.uahirise.org/epo/about/>.
- [5] E. Bahit, *Scrum y eXtreme Programming para Programadores*. Buenos Aires, Argentina: Creative Commons Atribución-NoComercialSinDerivadas 3.0 Unported., 2011-2012.
- [6] M. B. Donald Hearn, *Gráficos por computadora con OpenGL*. Ribera del Loira, 28 28042 Madrid: Pearson Educación S.A., 2006, ISBN - 10: 84-205-3980-5.
- [7] Design Wiki. (2018). A Brief History of Computer Graphics, [Online]. Disponible en: https://deseng.ryerson.ca/dokuwiki/mec222:brief_history_of_computer_graphics.
- [8] Alejandro Sacristán. (ene. de 2018). Nos movemos hacia una realidad extendida, [Online]. Disponible en: https://retina.elpais.com/retina/2018/01/26/tendencias/1516965980_952713.html.
- [9] Pablo G.Bejerano. (ago. de 2014). El origen de la realidad aumentada, [Online]. Disponible en: <https://blogthinkbig.com/realidad-aumentada-origen>.

- [10] xatakandroid. (mar. de 2018). Ikea Place, su aplicación de realidad aumentada para decorar tu casa llega a los móviles Android con ARCore, [Online]. Disponible en: <https://www.xatakandroid.com/aplicaciones-android/ikea-place-su-aplicacion-de-realidad-aumentada-para-decorar-tu-casa-llega-a-los-moviles-android-con-arcore>.
- [11] Visbox. (n.d.). CAVE Automatic Virtual Environment, [Online]. Disponible en: <http://www.visbox.com/products/cave/>.
- [12] C. M. A. David Vallejo Fernández, *Desarrollo de Videojuegos: Arquitectura del Motor*. Bubok, 2013, ISBN: 978-84-686-4028-0.
- [13] Unity3D. (n.d.). El motor de creación de contenido líder en el mundo, [Online]. Disponible en: https://unity3d.com/es/unity?_ga=2.202731025.1640267260.1528731971-324731097.1518535970.
- [14] Guy Webster, Veronica McGregor, Dwayne Brown. (ene. de 2015). NASA, Microsoft Collaboration Will Allow Scientists to 'Work on Mars', [Online]. Disponible en: <https://www.jpl.nasa.gov/news/news.php?feature=4451>.
- [15] Steam. (n.d.). Mars Odyssey, [Online]. Disponible en: https://store.steampowered.com/app/465150/Mars_Odyssey/.
- [16] NASA Jet Propulsion Laboratory. (n.d.). Experience Curiosity, [Online]. Disponible en: <https://eyes.nasa.gov/curiosity/>.
- [17] Pablo F. Iglesias. (ene. de 2018). ¿Por qué es tan importante el User Experience o Experiencia del Usuario?, [Online]. Disponible en: <https://www.pabloyglesias.com/interaccion-realidad-virtual/>.

Universidad de Alcalá
Escuela Politécnica Superior



ESCUELA POLITECNICA
SUPERIOR

