UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA DEL SOFTWARE

LABORATORIO DE INTERFACES DE REALIDAD VIRTUAL

VIRTUAL REALITY INTERFACE LAB

Realizado por
JOSÉ MANUEL RASCÓN ALCÁNTARA

Tutorizado por
EDUARDO GUZMÁN DE LOS RISCOS

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

MÁLAGA, febrero de 2020

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DEL SOFTWARE

LABORATORIO DE INTERFACES DE REALIDAD VIRTUAL

VIRTUAL REALITY INTERFACE LAB

Realizado por
José Manuel Rascón Alcántara

Tutorizado por
Eduardo Guzmán de los Riscos

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, FEBRERO DE 2020

Resumen

El objetivo de este trabajo de fin de grado es desarrollar una aplicación de realidad virtual para SteamVR con uso del visor HTC Vive. Ofrece a las personas interesadas en el desarrollo de la realidad virtual una formación teórica sobre todo lo relacionado con esta tecnología y sobre las cosas a tener en cuenta a la hora de desarrollar para este formato específico. Esta formación se realiza por medio de la lectura, el ejemplo y la interacción que la realidad virtual ofrece. El usuario vivirá la experiencia de encontrarse en pleno espacio profundo dentro de una estación espacial, pudiendo manipular distintas interfaces de usuario, moverse e interactuar con el entorno libremente, a la par que leer sobre el diseño y uso de los diferentes elementos que encuentra, aprendiendo no solo a entender los controles de la experiencia, sino también los conocimientos teóricos en los que se basa, de forma amena y entretenida.

Palabras clave:

Realidad Virtual, Enseñanza, Desarrollo

Abstract

The goal of this Bachelor thesis is the development of a virtual reality application for SteamVR using the HTC Vive headset. It offers people interested in the development of virtual reality a theoretical training, especially related to this technology and things to take into account when developing for this specific format. This training is done through reading, the example and the interaction that virtual reality offers. The user will live the experience of being in deep space within a space station, being able to manipulate different user interfaces, move and interact with the environment freely, while reading about the design and use of the different elements that one finds, learning not only to understand the controls of the experience, but also the theoretical knowledge which it is based on, in an entertaining way.

Keywords:

Virtual reality, Teaching, Develop

Agradecimientos

Gracias a Virtual Pixel por su formato, sin el cual no tendría los conocimientos suficientes para realizar esta memoria y haber encontrado algo a lo que realmente gustaría dedicarme en un futuro, la realidad virtual.

A mi pareja por su apoyo, ayuda con la memoria y ayudas en el día a día; sin ella dudo mucho que hubiera cumplido los tiempos de entrega.

A mi tutor por permitirme realizar el trabajo y darme su apoyo.

Y a todos los anónimos que dedicaron parte de su tiempo para el capítulo de pruebas.

Índice

1

Introducción

1.1 Motivación

La realidad virtual está naciendo en nuestra sociedad con fuertes promesas de ser de ayuda para multitud de sectores como la educación, la industria, la medicina y el entretenimiento. Hoy en día es difícil encontrar perfiles adecuados con suficiente experiencia previa o conocimientos sobre las cosas a tener en cuenta al desarrollar en este nuevo formato, provocando que la mayoría de las experiencias sean experimentales y llenas de imperfecciones que hacen que los usuarios no terminen de sentirse cómodos en la realidad virtual.

Documentar la experiencia ganada desarrollando para realidad virtual y transmitirla es fundamental para que poco a poco los errores más comunes desaparezcan, además de que se empiecen a crear cursos especializados en la materia tanto teóricos como prácticos. Muchos cursos aseguran estar especializados en realidad virtual pero solo se centran en la parte de desarrollar con los plugins actuales como SteamVR plugin y Oculus Integration, dando de lado todos los conocimientos teóricos que envuelven las tecnologías de realidad virtual.

1.2 Objetivos

El objetivo es implementar una aplicación de realidad virtual donde enseñar, a través de ejemplos, las cosas a tener en cuenta cuando se desarrolla para realidad virtual y mostrar las diferentes interfaces que existen en ella. Además, esta memoria la cual podrá verse desde dentro de la propia experiencia, también introduce al lector de manera teórica muchos términos y nociones del mundo que envuelve a las tecnologías de realidad virtual. Alguien que pruebe la experiencia podrá estudiar la teoría mientras asienta sus conocimientos con la experiencia mediante la práctica.

Pero, ¿cuál es el motivo de usar la realidad virtual para enseñar a desarrollar realidad virtual?
Muchos estudios aseguran que usar la realidad virtual para la enseñanza tiene muchos beneficios [1]. Algunos de esos beneficios son:

- Mejora en la comprensión
- Se aprende más rápido
- Los conocimientos son más duraderos
- Es una enseñanza mucho más amena

Todo esto ocurre gracias a que están involucrados muchos sentidos, a su potencial práctico y capacidad de interactuar en un mundo en tres dimensiones. Poder cambiar el punto de vista al observar algo y poder manipularlo con las manos hace que la experiencia sea muy cercana al mundo real.

Por tanto, la aplicación tiene un papel de enseñanza fundamental pues podremos probar de primera mano todos los fundamentos teóricos que se introducen en esta memoria. La experiencia también contiene 3 tipos de interfaces de usuario distintas: un menú holográfico, un casco de astronauta y unas gafas de realidad aumentada, con las que aprenderemos diferentes maneras de presentar información al usuario en un entorno tridimensional.

1.3 Tecnologías utilizadas

A continuación se listan todas las herramientas que se han utilizado para el desarrollo de la experiencia:

1.3.1 Herramientas de desarrollo:

- **Unity 2019.2.11f1**

Figura 1.1: Logo de Unity.

Es un motor de videojuegos multiplataforma creado por Unity Technologies, muy popular actualmente. Tiene una inmensa comunidad y buena documentación. Además permite, gracias a los assets gratuitos de su tienda, realizar una demo sin gastar dinero en modelos, efectos de partículas y demás cosas necesarias para el desarrollo [2].

Se decidió hacer uso de esta herramienta porque ya se había trabajado con ella anteriormente y así poder ahorrar tiempo en la parte de formación del proyecto. Otro de los puntos a favor es que es una herramienta que se está usando actualmente para el desarrollo de software de realidad virtual y entre sus assets se encuentran SteamVR Plugin y Oculus Integration, fundamentales en el desarrollo de realidad virtual.

- **Microsoft Visual Studio 2019**

Figura 1.2: Logo de Microsoft Visual Studio 2019.

Es un entorno de desarrollo integrado (*IDE*), compatible con múltiples lenguajes de programación entre ellos C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP. Tiene muchos componentes que ayudan al desarrollo. Por ejemplo, para dispositivos móviles con .Net lo cual añade entre muchas cosas un compilador para iOS, Android y Xamarin. También en el desarrollo con Unity añade muchas ayudas para trabajar con sus funciones como sugerencias de completado, sugerencias de corrección de errores, etc. Su desarrollador es Microsoft [3].

Se decidió hacer uso de esta herramienta porque aunque no es el IDE predeterminado de Unity, al tener soporte para C# y un componente para Unity, facilitaba el desarrollo del código de la experiencia. Además al ser una herramienta tan polivalente, trabajar con ella otorga una experiencia que es útil en muchas otras situaciones.

- **SQLite DB Browser**

Figura 1.3: Logo de SQLite DB Browser.

Es una interfaz gráfica de usuario (GUI) para SQLite que es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp [4].

Se decidió hacer uso de esta herramienta porque se necesitaba guardar información de manera permanente y local. SQLite también tiene otras ventajas tales como: que no requiere instalación local, es más rápida que los sistemas de archivos y es un formato muy ligero.

1.3.2 Herramientas de gestión:

- **Git Kraken**

Figura 1.4: Logo de Git Kraken.

Es una GUI para Git que es un software de control de versiones diseñado por Linus Torvalds. Git Kraken tiene soporte para Window, MacOs y Linux [5].

Se decidió hacer uso de esta herramienta porque se necesitaba un software de control de versiones que tuviera una interfaz sencilla y ya se tenía experiencia previa con su interfaz.

- **Trello**

Figura 1.5: Logo de Trello.

Es un software de administración de proyectos con interfaz web y con cliente para iOS y android para organizar proyectos. Desarrollado por Atlassian [6].

Se decidió hacer uso de esta herramienta porque da muy buen control sobre el estado del proyecto gracias a todas sus capacidades y su parecido a la metodología SCRUM, vista en clase. Es gratis y muy cómodo.

1.3.3 Herramientas de documentación:

- **Google Document**

Figura 1.6: Logo de Google Document

Es un procesador de texto en línea gratuito que nos permite crear, editar documentos y compartirlos en la red con otros usuarios. Todo ello sincronizado en línea en tiempo real. Tiene un control de versiones, capaz incluso de recuperar el archivo si es borrado desde

Drive. Soporta gran cantidad de formatos como Word, PDF, documento de texto y convertirlos de unos a otros. Desarrollado por Google.

Se decidió hacer uso de esta herramienta porque da la posibilidad de compartir la memoria con el tutor, ver las modificaciones en tiempo real, pasar el documento a cualquier formato y ser gratuita.

- **StarUML**

Figura 1.7: Logo de StarUML

Es una herramienta de UML desarrollada por MKLab. Nos permite hacer diagramas de clases, casos de uso, de secuencia, etc y es capaz de generar documentación desde los diagramas utilizando sus extensiones [7].

Al descubrir esta herramienta no se dudó en hacer uso de ella. Sencilla, completa y gratuita en su versión de pruebas de tiempo indefinido.

1.3.4 Herramientas varias:

- **GIMP 2**

Figura 1.8: Logo de GIMP 2

Es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un programa de software libre y gratuito [8].

Se decidió hacer uso de esta herramienta porque ya se tenía mucha experiencia en el tratamiento de imágenes con ella y a diferencia de Photoshop no es de pago y tiene Licencia Pública General GNU (GPLv3).

1.4 Estructura de la memoria

En este apartado se dará una breve descripción de los futuros capítulos de este documento:

- **Capítulo 2: Nomenclatura**
  En este capítulo se describe alguno de los términos que vamos a utilizar a lo largo de la memoria.

- **Capítulo 3: Estudio del arte**
  En este capítulo se hablará sobre las características de los visores más importantes en el mercado actual y sobre experiencias con objetivos parecidos a los de este proyecto y que fueron de inspiración para hacerlo.

- **Capítulo 4: Buenas prácticas**
  En este capítulo se repasará algunos consejos y advertencias que nos ayudarán a tomar decisiones de diseño y a que nuestra experiencia virtual mejore.

- **Capítulo 5: Especificación y análisis**
  En este capítulo se mostrará todos los requisitos que componen la experiencia y cómo evolucionaron a lo largo del desarrollo, cambiando a requisitos distintos o simplificarlos para cumplir los tiempos de desarrollo.

- **Capítulo 6: Diseño e implementación**

  En este capítulo se mostrará las estructuras de diseño más relevantes para el desarrollo de la experiencia, como la estructura de la base de datos, la jerarquía de carpetas del proyecto unity, etc. También se comentará el código de las partes más relevantes y complicadas como la interacción simple con objetos.

- **Capítulo 7: Pruebas**

  En este capítulo se recogen las estructuras de los formularios y pruebas que tuvieron que realizar los usuarios reales, así como los datos recolectados por dichas pruebas y las conclusiones obtenidas.

- **Capítulo 8: Conclusión y trabajos futuros**

  En este capítulo se comentarán las conclusiones a las que hemos llegado al finalizar la memoria y posibles continuaciones.

2

Nomenclatura

A continuación se describirá alguno de los términos que vamos a utilizar a lo largo de la memoria. Términos que son importantes para comprender la teoría en la que está envuelta la tecnología de la realidad virtual y en los que nos basaremos para tomar las decisiones de diseño.

Seis grado de libertad (6DOF): Se refiere al movimiento en un espacio tridimensional, es decir, la capacidad de moverse hacia delante/atrás, arriba/abajo, izquierda/derecha (traslación en tres ejes perpendiculares), combinados con la rotación sobre tres ejes perpendiculares (Guiñada, Cabeceo, Alabeo) [14]. Siguiendo la figura 2.1 sería:

-Delante/Atrás (Verde)
-Arriba/Abajo (Amarillo)
-Izquierda/Derecha (Violeta)
-Guiñada (Naranja)
-Cabeceo (Rosa)
-Alabeo (Azul)

Figura 2.1: Seis grados de libertad en colores

Tracking: Es una palabra anglosajona que significa “rastreo” en realidad virtual hace referencia a la posición y rotación de los controladores o del visor en el mundo virtual. Se dice que el tracking de los controladores está fallando cuando la posición y rotación de los controladores en el mundo real difiere de la del mundo virtual.

Sistema de Tracking Inside Out: Se produce cuando los componentes encargados de realizar el seguimiento de la posición y rotación del visor y controladores, se encuentran dentro del visor. Exceptuando al acelerómetro, giroscopio y magnetómetro encargados del seguimiento rotacional del visor, si no de los sistemas de cámaras, infrarrojos u otros que tenga un visor de realidad virtual.

Sistema de Tracking Outside In: Es el inverso de Inside Out, donde algunos de los componentes que realizan el seguimiento se encuentran fuera de visor, normalmente suelen ser bases emisoras o receptoras de infrarrojos o cámaras equipadas con procesamiento de imagen.

FOV (field of view): El campo de visión o campo de perspectiva, es el ángulo de visión máximo comprendido en nuestra vista cuando tenemos los ojos en la línea de visión cómoda. Medir con exactitud el FOV es imposible pues depende de nuestras características faciales, la forma del visor y la distancia de nuestro ojo a la pantalla.
El FOV humano binocular es de aproximadamente 114º horizontal, 150º verticales y un periférico restante de +40º horizontales en cada ojo (porque solo un ojo puede ver esas partes del campo visual).

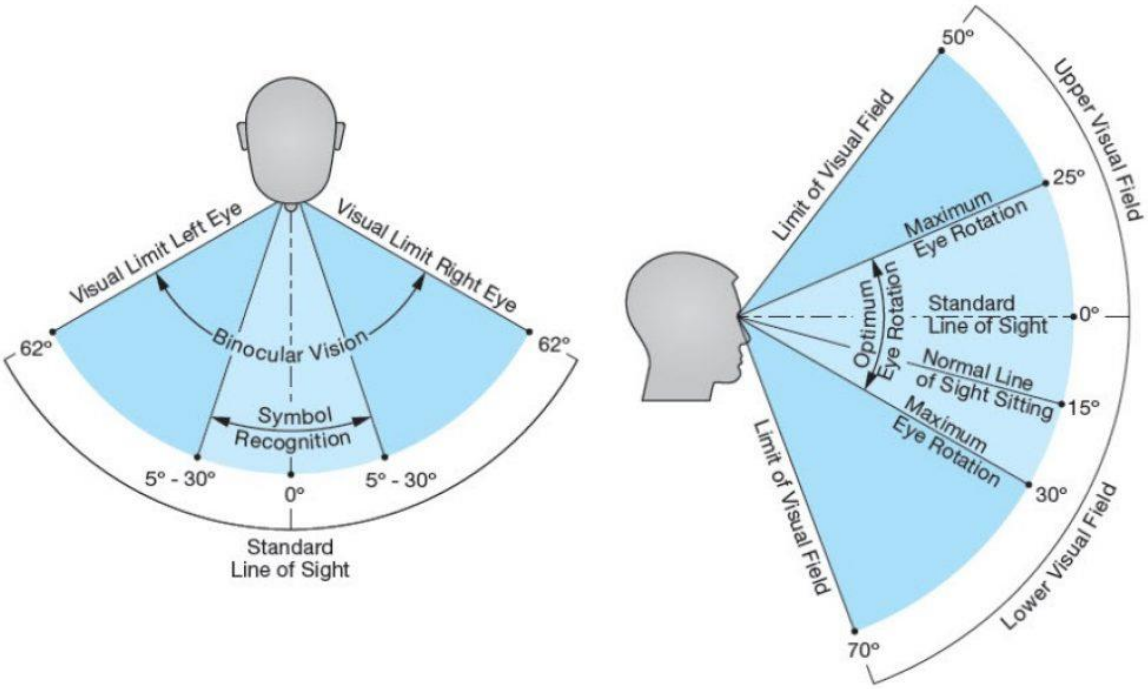

Figura 2.2: El campo de visión o campo de perspectiva, horizontal y vertical.

IPD (interpupillary distance): Es la distancia medida en milímetros entre los centros de las pupilas de los ojos. Es muy importante tenerlo bien ajustado pues si el IPD no está mínimamente en un valor cercano de quién lo está usando, la visión general en el visor empeora y por tanto los ojos se esfuerzan mucho más llegando a producir algunos síntomas como dolores de cabeza, fatiga y dolor ocular. Este valor es asimétrico, lo que significa que no es el mismo en ambos ojos y su media en humanos está entorno a 61.7 mm en mujeres y 64 mm en hombres. Medirse el IPD en una óptica suele ser completamente gratuito y muy recomendable para poder disfrutar plenamente la realidad virtual.

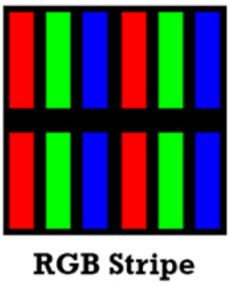

Figura 2.3 : Subpixeles en distribución RGB Stripe.

RGB Stripe: Distribución de subpíxeles en forma de matriz donde cada píxel tiene 3 subpíxeles rectangulares de diferente color uno rojo (Red), uno verde (Green) y uno azul (Blue).

Figura 2.4: Subpixeles en distribución Diamond PenTile.

Diamond PenTile: Distribución de subpíxeles en la que intervienen dos matrices: una compuesta de subpíxeles azules y rojos con forma de rombo y otra compuesta de subpíxeles verde más pequeños con forma circular. Ambas matrices se entrelazan haciendo que entre los huecos producidos por los leds rojos y azules haya siempre un led verde. Esto se aprovecha de que el ojo humano detecta de forma diferente las diferentes frecuencias de color, necesitando más luz para colores como el verde y menos para colores como el morado o el rojo como bien indica la figura 2.5. Por tanto, aunque haya menos píxeles rojos y azules que verdes nuestro ojo no nota una diferencia y podemos aumentar así la cantidad de píxeles por espacio (resolución).

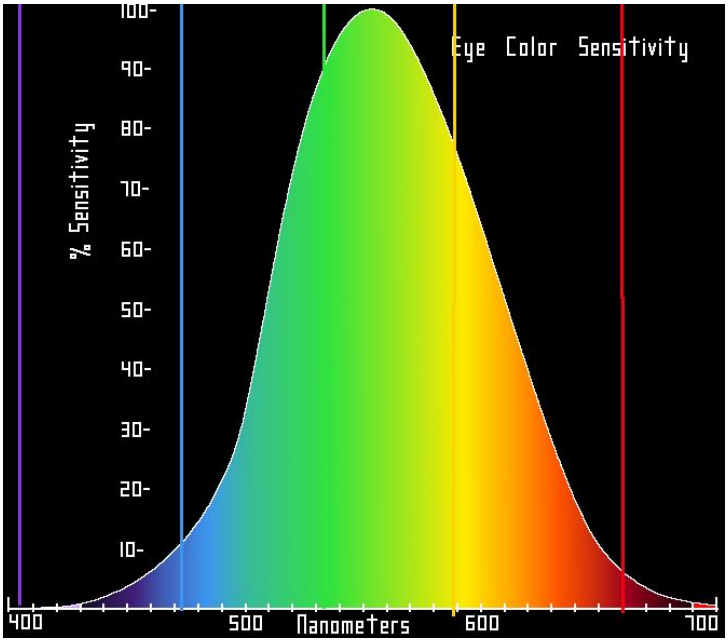

Figura 2.5 : Brillo relativo de la sensibilidad del sistema visual humano en función de la longitud de onda.

Glare: Es un efecto lumínico que ocurre cuando hay colores muy claros en fondos oscuros, se produce cuando la luz refractada por la lente sigue un patrón indeseado y su efecto es una especie de deslumbramiento como podemos ver en la figura 2.6.

Figura 2.6 : Deslumbramiento producido por el Glare en una lente Frenel.

Oclusión: Se habla de oclusión en la realidad virtual cuando un objeto se interpone entre el sistema de Tracking y lo que se quiere posicionar. Sería un ejemplo el de una persona que pone su mano entre las cámaras de un Sistema Inside Out y su controlador, su mano está haciendo oclusión a su controlador. Frente a esto los Sistemas de Tracking deben estar preparados haciendo estimaciones de posición o haciendo desaparecer el elemento en el mundo virtual.
Hoy en día la oclusión no afecta a la rotación, pues los componentes encargados de ello (magnetómetro, giroscopio y acelerómetro) se hallan siempre integrados con el propio sistema.

El sistema vestibular: Es el sistema del oído interno que recopila información sensorial crítica para el movimiento, el equilibrio y la orientación espacial. Nótesr que el sistema vestibular humano no percibe velocidad constante, sólo aceleración y desaceleración.

Cinetosis (Motion Sickness): Es el trastorno debido al movimiento. En particular en la realidad virtual ocurre cuando el sistema vestibular y el de la vista no son coherentes el uno con el otro. Los síntomas que produce en realidad virtual son la desorientación y/o mareos.

Piernas Virtuales: Se dice tener piernas virtuales, cuando una persona no sufre los síntomas de la cinetosis gracias a su adaptación a la realidad virtual cuando realiza movimientos virtuales consciente, como el de desplazarse hacia delante pulsando un botón del controlador. La mejor manera de conseguirlas es cumplir una simple regla: cuando se está inmerso en una experiencia y esta empieza a provocar incomodidad, debe detenerse la experiencia y descansar. Con el tiempo se producirá una adaptación.

Valle Inquietante (Uncanny Valley): Es una hipótesis propia del campo de la robótica que debido a la inmersión que produce la realidad virtual es igualmente válida. La hipótesis afirma que cuando las réplicas antropomórficas se acercan en exceso a la

apariencia y comportamiento real, provocan una respuesta de rechazo entre los observadores humano. Este efecto ocurre sobre todo en los modelos de manos. Un ejemplo de valle inquietante en animación es el del bebé que aparece en el cortometraje de Pixar llamado Tin Toy (1988) [13][15].

3

Estudio del arte

3.1 Visores actuales en el mercado

A continuación se listarán algunos visores, destacando algunas curiosidades e informando de sus propiedades. Nótese que no será lo suficientemente exhaustivo como para servir de ayuda a la hora de tomar una decisión de compra, ya que entre muchas cosas no se hablará de su precio actual. Este listado busca ser la base desde la cual tomaremos las decisiones de diseño.

Los visores que mayor impacto han tenido en el mercado en este último año son:

- **Oculus Quest**

Figura 3.1 : Modelo Oculus Quest.

Es un visor de la compañía de Oculus, que salió recientemente en mayo de 2019. Es el único visor 6DOF del mercado que no requiere de un ordenador para su uso, lo que comúnmente es conocido con el término “Standalone”.

Al ser el único visor en el mercado de su formato, junto al precio de 450 euros en su salida para la versión de 64Gb, ha logrado ser el visor con mayores ventas en sus primeros meses de la historia de realidad virtual actual.

Usa un sistema de Tracking Inside Out gracias a su sistema de 4 cámaras e infrarrojos, junto al software correspondiente de gestión y procesamiento de imágenes, almacenados en unos procesadores especializados solo para esta tarea.

Aun siendo un dispositivo standalone puede conectarse a un ordenador mediante un cable USB USB-C aunque no todos los cables son válidos, Oculus lo vende como un componente independiente llamado Oculus Link.

Características:

- Su FOV es de 104º horizontales y 120º verticales.
- Dispone de ajuste de IPD mecánico.
- Tiene una pantalla OLED 1440x1600 píxeles de resolución por ojo, con distribución de subpixel en Diamond penTile.
- El peso del visor 571 gramos.
- La ergonomía de sus controladores es muy cómoda pero la del visor deja mucho que desear sobretodo en el equilibrio de peso del visor que está muy desequilibrado hacia el rostro. Sus controladores son los mismos que los de Oculus Rift S.

- **Oculus Rift S**

Figura 3.2 : Modelo Oculus Rift S.

Es un visor de la compañía de Oculus, que junto a Oculus Quest salió en mayo de 2019.

Usa un sistema de Tracking Inside Out igual al de Oculus Quest pero en este modelo tiene una cámara más y están distribuidas de manera diferente.

Características:

- Su FOV es de 88º horizontales y 118º verticales.
- No dispone de ajuste de IPD mecánico pero sí de ajuste de IPD por software.
- Pantalla LCD 2560x1440 píxeles de resolución para ambos ojos, con distribución de subpixel en RGB Stripe.
- Peso del visor 563 gramos.
- La ergonomía de sus controladores es muy buena y la de su visor es muy cómoda gracias a su forma de diadema. Sus controladores son los mismos que los de Oculus Quest.

- **Valve Index**

Figura 3.3 : Modelo Valve Index.

Es el primer visor de la compañía Steam, salió en Junio de 2019.

Es compatible con algunos componentes de las HTC Vive. Esto se debe a que anteriormente fue Steam quien financió a HTC y se sospecha que han usado gran parte los diseños de esa compañía para sacar su visor.

Usa un sistema de Tracking Outside In gracias a sus dos bases y un software de posicionamiento por infrarrojos, almacenados en unos procesadores especializados solo para esta tarea.

Características:

- Su FOV es de 108º horizontales y 132º verticales.

- Dispone de ajuste de IPD mecánico.

- Pantalla LCD 1440x1600 píxeles de resolución por ojo, con distribución de subpixel en RGB Stripe.

- Peso del visor 781 gramos.

- La ergonomía de sus controladores y del visor es muy cómoda, el visor trae una diadema bien equilibrada en peso y los controladores tienen un agarre especial para poder agarrarse sin usar ningún dedo de la mano, lo cual es un gran avance.

- **Play Station Virtual Reality (PSVR)**

Figura 3.4 : Modelo Play Station VR .

Fue el primer visor de la compañía Sony, salió en octubre de 2016.

Aunque este visor no es muy reciente, su competitivo precio y su gran cantidad de juegos hacen que siga siendo un visor que compite con otros más nuevos en el mercado, llegando a alcanzar, a comienzos de 2020, la cantidad de cinco millones de visores vendidos desde su salida.

Usa un sistema de Tracking Outside In gracias a sus dos cámaras externas al visor y un software de procesamiento de imagen con el cual calcula su posicionamiento reconociendo las luces azules colocadas en el visor. También usa las cámaras y la luz del mando y los Move para posicionarlos. Esta tecnología tiene un gran defecto y es que el tracking solo funciona mientras no haya oclusión entre las luces y las cámaras. Se dice que es un tracking de 180º.

Características:

- Su FOV es de 100º horizontales y verticales.

- No dispone de ajuste de IPD mecánico pero sí de ajuste de IPD por software.

- Pantalla LCD 1920×1080 píxeles de resolución para ambos ojos, con distribución de subpixel en RGB Stripe.

- Peso del visor 300 gramos.

- Se puede jugar tanto con el mando de PS4 como con los Move. Su visor tiene una de las mejores diademas para la cabeza, ya que el visor tiene el peso equilibrado y se agarra muy bien en la cabeza. Pero lo que realmente le diferencia de los demás visores es que apenas tiene contacto con la cara, lo que ayuda a la transpiración de la piel.

- **HTC Vive**

Figura 3.5 : Modelo HTC Vive.

Fue el primer visor de la compañía de HTC, salió en febrero de 2016.

Aunque este visor no es de los últimos en salir al mercado, va a ser con el que se trabajará en este proyecto y, por tanto, se le hace una mención especial en este apartado para que podamos, entre otras muchas cosas, conocerlo y hacer la comparativa. También destacar que, junto a las Oculus Rift, fueron los primeros visores de realidad virtual con más éxito para PC dirigidos a usuarios finales.

Usa un sistema de Tracking Outside In gracias a sus dos bases y software de posicionamiento por infrarrojos.

Características:

- Su FOV es de 92º horizontales y 114º verticales.

- Dispone de ajuste de IPD mecánico.

- Pantalla OLED 1080×1200 píxeles de resolución por ojo, con distribución de subpixel en Diamond PenTile.

- Peso del visor 555 gramos.

- La ergonomía de sus controladores es pésima y también la de su visor; por suerte HTC sacaría un componente extra llamado Deluxe Audio Strap que mejora su comodidad en comparación con algunos visores actuales, pero no mejorando en absoluto la comodidad de sus controladores.

3.2 Aplicaciones actuales sobre realidad virtual

Actualmente muchas aplicaciones ofrecen experiencias de realidad virtual en ámbitos como la enseñanza y la educación, en donde pueden encontrarse aplicaciones para aprender sobre el sistema solar, la vida marina, etc. Pero aun así no se consiguió encontrar ninguna experiencia con un objetivo mínimamente parecido a los de este proyecto o que tuviera como objetivo a los desarrolladores de software de realidad virtual. No es descartable que exista alguna, pues el mercado de la realidad virtual actual todavía está formándose y no dispone de los grandes escaparates que pueden tener los videojuegos tradicionales o los cursos informativos, llegando solo al conocimiento de unos pocos.

Casualmente se tuvo la suerte de que la asociación XR (XRA) que cuenta entre sus miembros con los principales fabricantes de la industria (Oculus, HTC, Sony, Google, etc.), lanzara en diciembre de 2019 una versión actualizada de su guía de inicio para creadores de contenidos. Se trata de un nuevo paso en su misión de promover el desarrollo responsable y el avance de la tecnología de realidad virtual y aumentada a

nivel mundial. En su nueva guía aparecen muchos más conceptos que los que veremos en esta memoria y es una muy recomendada lectura para quienes quieran desarrollar para estas tecnologías [9].

Al no encontrarse experiencias similares de las que hablar en este apartado se ha decidido hacer mención a las que sirvieron de inspiración para algunas de las interfaces desarrolladas en este proyecto:

- **To the top**

Figura 3.6 : Portada de la experiencia To the top.

Desarrollado por Electric Hat Games, es un videojuego de plataformas de escalada contrarreloj. Sus mecánicas principales son poder agarrarse a superficies azules y saltar haciendo un gesto bastante realista con los controladores para así llegar a la meta en el menor tiempo posible.

Su menú holográfico portátil y su sistema de vuelo fueron inspiración para desarrollar el menú holográfico y el sistema de propulsión en las manos.

- **Narcosis**

Figura 3.7 : Portada de la experiencia Narcosis

Desarrollado por Honor Code, es un videojuego de terror psicológico. Sus mecánicas principales son la pérdida de oxígeno por motivo de estrés, apuñalar para atacar, un lanzador de bengalas para iluminar sitios oscuros y un pequeño propulsor para dar saltos. Aunque no salió en un principio para realidad virtual, más tarde publicaron una versión para este formato. Esta versión tiene una interfaz de realidad virtual muy interesante, pues además de ser muy intuitiva es también muy inmersiva, ayudando a la ambientación del juego a producirte auténtica angustia y temor.

Su interfaz fue inspiración para el casco de astronauta.

- **TestHMD**

Figura 3.8 : Logo de la experiencia Test HMD

Se hace mención especial de esta experiencia pues fue la principal inspiración para este proyecto. Es una experiencia que sirve para poder medir las propiedades de los visores con diferentes pruebas como el test de medición de agudeza visual lejana, test de contrastes, test de rejilla de Amsler, etc.

Con él se puede verificar cómo de bueno es un visor respecto a otro desde la experiencia propia y además aprender sobre los diferentes componentes y propiedades de un visor con paneles explicativos colocados en cada prueba [10].

4

Buenas prácticas

Tras definir los términos que se usarán a lo largo de la memoria y antes de entrar en los requisitos y en las decisiones de diseño específicos para nuestra experiencia, vamos a repasar algunos consejos y advertencias que nos ayudarán a tomar decisiones de diseño y a que nuestra experiencia virtual mejore.

4.1 Consejos y advertencias generales

- **Letras verdes en fondos oscuros:**
  Actualmente la resolución que nos dan las pantallas de los visores no es demasiado buena y supone un auténtico problema en la lectura, produciendo que los ojos se vean sometidos a un sobresfuerzo o que sea imposible su lectura.

  Ante esto debemos hacer uso de las buenas prácticas ya conocidas del formato tradicional, como elegir un buen tamaño de letra y evitar los contrastes de color fuertes.

  Pero en esta línea, se recomienda, elegir un color verde. El motivo se basa en todo lo que hemos aprendido anteriormente en esta memoria y es la diferencia de densidad de píxeles verdes en pantallas de tipo PenTile. Como podemos ver en

la figura 4.1 y tal y como hemos explicado anteriormente la densidad de subpíxeles verdes es mayor a la de los demás colores en las pantallas PenTile. Estas se encuentran en visores como Oculus Quest o HTC Vive. Por tanto si se está desarrollando para estos visores es una muy buena opción para aumentar la legibilidad de los textos. Además en caso de estar desarrollando para múltiples visores, al ser la segunda opción un RGB Stripe el cual tiene la densidad de subpíxeles de colores igual en todos los colores, no tiene una pérdida de ningún tipo ante cualquier otro color de texto, simplemente no tiene una ganancia de definición como en las PenTile.

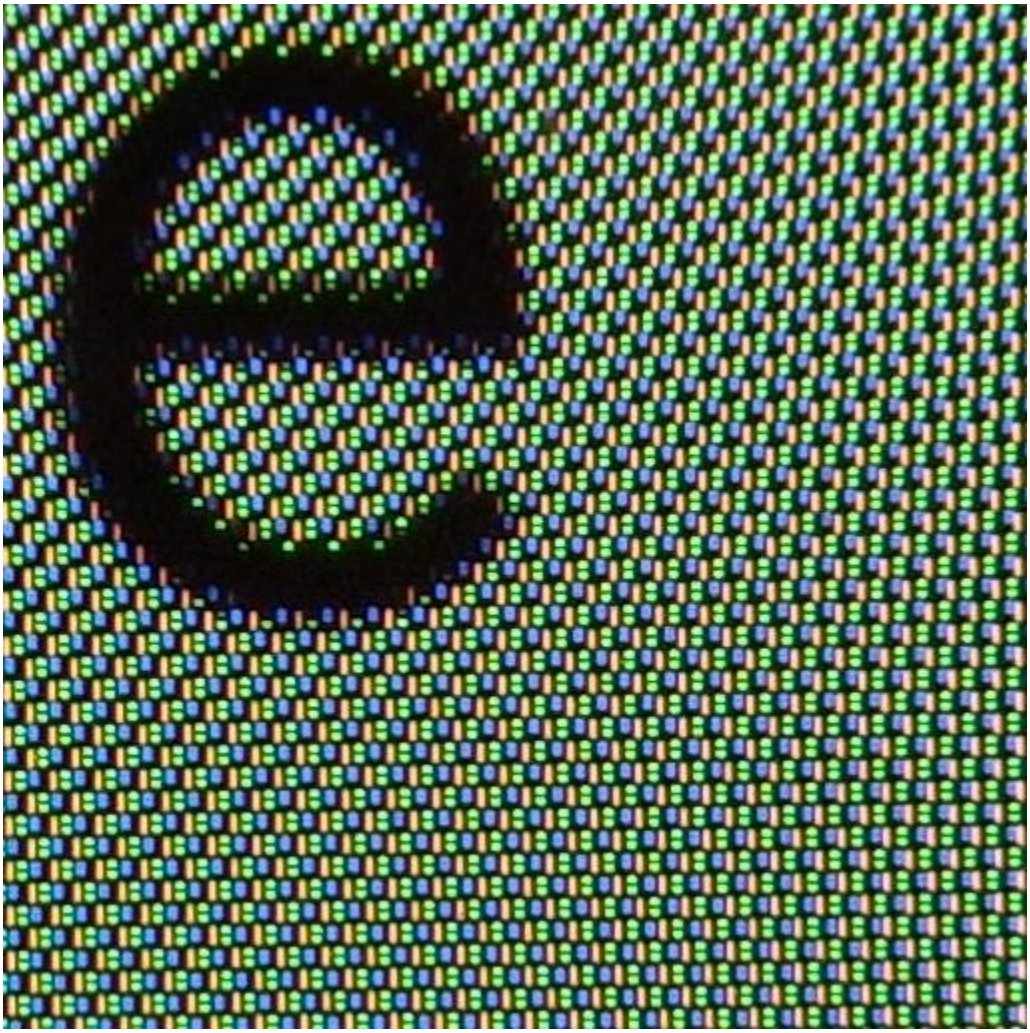

Figura 4.1: Visión de Subpíxeles de una pantalla PenTile.

- **Indicar los interactuables**

  Gracias a los controladores posicionados con tracking el usuario puede interactuar y manipular los objetos e interfaces que se le proporcionan en las experiencias desarrolladas en este TFG. No obstante, no hay nada más frustrante que intentar manipular algo y ver que no responde ante las acciones del usuario. Con esto se quiere destacar que el usuario necesita saber cuándo un elemento de nuestra experiencia se puede manipular y cuándo no.

  Hay muchas maneras de solucionar esto. La que se escogió para la experiencia es destacar el objeto mientras está en contacto con el controlador del usuario, como podemos ver en la figura 4.2 .

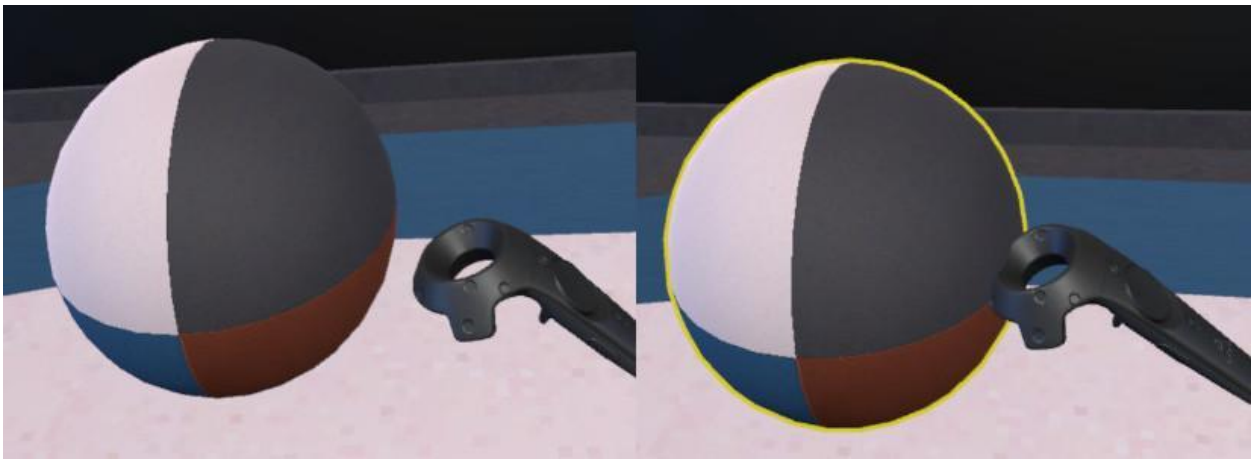

Figura 4.2: Captura de la experiencia, indicador visual de contacto.

  Con esto conseguimos evitar frustraciones innecesarias al usuario al mismo tiempo que le educamos en las reglas de nuestra experiencia.

- **Considerar también al resto de sentidos**

  Uno de los objetivos más importantes en las experiencias de realidad virtual es la inmersión. Una buena manera de conseguirla es usar todos los componentes de los que dispone un visor, por ejemplo, los auriculares y los motores de vibración de los controladores. Con esto podemos hacer que el usuario involucre en la

experiencia dos sentidos más además de la vista, añadiendo información al entorno.

Una muy buena manera de usarlo es añadiendo la vibración junto a sonidos, un ejemplo es cuando se tensa el arco en el juego desarrollado por Valve, The Lab. En este juego al tensar la flecha para lanzarla con el arco se reproduce un sonido a medida que se va tensando, acompañado de una vibración equivalente al sonido que escuchamos. Todo ello nos hace creer que la cuerda del arco nos ejerce una pequeña tensión y aunque la mayoría de usuarios no es consciente, algunos llegan a apretar la musculatura de los brazos inconscientemente para enfrentarla, lo cual es síntoma de la inmersión que produce.

También podemos agregar esta idea al punto anterior añadiendo una pequeña vibración y/o sonido al entrar en contacto con algo que podamos interactuar, esto nos aporta que podamos saber cuándo hemos tocado algo que se puede manipular aunque no lo estuviéramos mirando con el visor.

- **Opción de rotación virtual**

  La rotación virtual consiste en girar la visión que tiene el usuario sin necesidad de que él se mueva en el mundo real. Algunos visores hoy en día no tienen un tracking completo o por variables del entorno su tracking falla en algunas partes, tal vez simplemente el usuario está sentado y su silla no puede darle la libertad de girar. Estos pueden ser algunos de los muchos motivos para añadir entre nuestros controles la posibilidad de que el usuario realice una rotación virtual.
  Las ventajas de añadirla como una opción es que si al usuario no le gusta o no lo necesita puede quitarlo en los ajustes de la experiencia.
  La gran ventaja de la Rotación Virtual es hacer sentir al usuario que tiene el control, dándole una herramienta con la que evitar situaciones incómodas para él.

  Para dejar claro el beneficio de la rotación virtual, un ejemplo:

A veces el usuario tiene un espacio limitado para jugar. Dichos usuarios tienden a aprovechar todo lo que pueden su espacio, por ejemplo, no jugando en el centro de la sala sino con la espalda a unos centímetros de alguna pared o mobiliario. Si por alguna casualidad la escena se encuentra girada en comparación al mundo real de manera que el usuario no puede jugar cómodamente, la rotación virtual les da esa herramienta para que puedan seguir jugando cómodamente con tan solo unos pequeños ajustes antes de empezar.

- **Reset**

  Sigue los mismos principios de la rotación virtual. El reset consiste en volver a la posición correcta o cómoda de la experiencia, sobre todo en aquellas experiencias en las que el usuario no tiene medios para moverse. El reset es otra herramienta de control para que el usuario pueda decidir en qué lugar del mundo real colocarse sin afectar al juego o incluso para corregir algún error en la escena, que se haya desplazado por algún fallo y necesite corregirse.

  Un buen ejemplo de experiencias sin movimiento son las montañas rusas donde el usuario puede decidir en qué lugar del mundo real está el asiento del vagón, por ejemplo el sofá.

  Para realizar un reset lo más aconsejable es colocarlo en un botón del controlador de rara frecuencia de uso, que al presionarse tome la dirección en la que observa el visor para reposicionar la escena.

  Un ejemplo de esto es la primera escena de ‘’I Expect you to die’’ (Figura 4.3).

Figura 4.3: Captura de la primera escena de la experiencia I Expect you to Die.

- **La Escala**

  Es un factor más para una buena inmersión. Si las cosas son desproporcionadas produce un efecto de rechazo con el entorno y sobre todo no se ve real. Buen ejemplo de esto es Skyrim VR donde sin importar que raza se escoja, se sentirá que los personajes del juego son enormes por culpa de un mal escalado.

Una buena manera de paliar la escala es tenerla muy en cuenta cuando se empiece a desarrollar y dedicarle su debido tiempo cuando se vayan introduciendo nuevos modelos en la experiencia. Otra buena práctica es dejar una opción para restablecer la altura, pues los sistemas de tracking actuales cometen fallos, que suelen notarse cuando el suelo real no está en la misma posición que el suelo virtual y esto repercute en nuestra altura virtual. En esta experiencia hay un ejemplo de ello en los ajustes del menú holográfico, en el que nos indican que toquemos el suelo real con los controladores, para calibrar esos posibles fallos.

Recalcar que si tal vez el objetivo de la experiencia es hacer sentir pequeño al usuario, eso no exime de tener cuidado con las escalas, aunque en este caso están pensada en unas medidas distintas.

- **Controladores o manos**

  ¿Qué es mejor, el modelo original de los controladores reales o un modelo de manos? Esta no es una pregunta que se pueda resolver con una sola respuesta, ya que ninguna opción es mejor que la otra.

  Las manos pueden otorgar una presencia mucho mayor a los controladores, pero no se podrá dar indicaciones de los botones del controlador, pues no están en su modelo.

  Por tanto, la elección de los controladores es mejor si el usuario objetivo de la experiencia es más novato y/o no conoce la forma y disposición de los botones del controlador.

  Se pueden combinar, poner un modelo de mano virtual que sujete nuestros mandos teniendo así lo mejor de los dos mundos, pero por supuesto aun así no tendrá la inmersión que da la ausencia del controlador, incluso puede que la mano tape algunos botones impidiendo ver su disposición.

  Lo recomendable es dedicar un tiempo a la decisión de qué modelo elegir y, en caso de elegir los controladores, modificar su modelado para darle un toque más parecido a la ambientación de la experiencia, intentando no dificultar la comprensión de la forma y disposición de los botones, pero consiguiendo ganar algo de inmersión y dándole un toque personal.

- **Efecto valle inquietante en manos**

  Esto se produce cuando el modelo de las manos no cumple las escalas de una mano humana. La manera más simple de arreglarlo es buscando un buen modelo que no produzca rechazo al usuario, esto último puede sonar fácil pero no lo es en absoluto, por lo que para paliar el rechazo son recomendadas las siguientes decisiones: evitar manos desnudas, utilizar modelos de guantes, manos robóticas o manos holográficas, estos al alejarse levemente de una mano real pueden llegar a no producir un efecto de valle inquietante.

  Hay que puntualizar que por mucho que un modelo de manos fuera perfecto, las medidas antropomórficas de las diferentes razas humanas son distintas, incluso dentro de las razas los pequeños detalles que diferencian nuestras manos harán muy difícil la completa desaparición del efecto valle inquietante.

  Un buen ejemplo de un mal modelado de manos es el de la experiencia Arizona Sunshine.

- **Control de la pausa**

  En muchas ocasiones el usuario es interrumpido por cosas externas del mundo real: le llaman por teléfono, un conocido le pide atención para poder pasar por su lado sin correr peligro de ser golpeado o el usuario se golpea con el mobiliario de la sala. Ante todas estas interrupciones no podemos hacer nada para reducirlas, pero sí podemos dar al usuario una herramienta de pausa para que pueda estar tranquilo de que ninguna interrupción hará que se pierda parte de la experiencia. Deben tenerse en cuenta las interrupciones externas y sugerir el uso del sensor de proximidad del visor como una ayuda para la pausa. Se trata de un sensor colocado en el visor apuntando al entrecejo del usuario, el cual nos indica si hay algo cerca, o dicho de otro modo, si el usuario tiene puesto el visor. Podemos encontrarlo por dentro del visor colocado por encima de las lentes.

- **Aclimatización a un entorno virtual**

  Los usuarios tardan tiempo en adaptarse a un nuevo mundo virtual o una nueva escena dentro de un mundo virtual. Las instrucciones o elecciones inmediatas son

más difíciles de procesar para los usuarios mientras están bajo la carga cognitiva de adaptación al nuevo entorno.

Al presentar a los usuarios un nuevo entorno virtual, comienza lentamente. Comienza con interacciones más tranquilas y de ritmo más lento que faciliten a los usuarios la experiencia y les permitan adaptarse al nuevo entorno poco a poco. Hay que evitar presentar información o hacer preguntas al usuario inmediatamente después de un cambio de escena. Es poco probable que los usuarios presten atención a las señales informativas durante los primeros momentos en una nueva escena.

- **Movimiento virtual frente al movimiento real**

  El movimiento real es el que realizamos a diario sin necesidad de un visor, caminar, darse la vuelta, agacharse, etc.

  El movimiento virtual es todo movimiento dentro de la experiencia que no ha sido producido por un Tracking de movimientos reales.

  Este consejo intenta advertirnos sobre esta diferencia, pues puede llegar a ocurrir que no esté contemplado por algunas variables de la experiencia. Sería el caso de una experiencia en la que el avatar virtual no se moviera si realizas movimientos reales. Esto produciría, por ejemplo, que en un juego de disparos los enemigos no jugadores que disparan, apuntarán a donde estaba el jugador antes de que se moviera por la habitación alejándose de ese punto, provocando que las balas no le dieran porque los enemigos siguen creyendo que está allí, aunque virtual y realmente no lo esté.

  Este consejo seguramente no haya ni que prestarle atención en un futuro, pues los plugins están saliendo con software preparado para que esto no ocurra, aun así se menciona porque en el plugin Oculus Integration para Unreal, esto no está corregido todavía. Por ello el desarrollador debe suplir esta falta con códigos propios o de otros compañeros desarrolladores que los publicarán en la red.

Un dato curioso es que en la salida de Robo Recall para Oculus Quest este error ocurría, produciendo que nunca derrotaran al usuario pues los robots enemigos atacaban al aire.

Por otro lado, está el problema de no poder impedir que el usuario se mueva en la vida real, por lo hay que hacer software específico para evitar consecuencias como que el usuario traspase paredes, manipule objetos que están dentro de un baúl cerrado con llave, etc. La solución a este problema es totalmente personal a la experiencia según sus mecánicas y objetivos. De la solución se hablará más adelante en el apartado 6.5.

4.2 Consejos y advertencias sobre la cinetosis

La cinetosis es uno de los grandes problemas de la realidad virtual, no tenerla en cuenta puede llegar a provocar un fracaso absoluto para una experiencia y, más importante, que el usuario habrá sufrido un malestar.

‘’Yo no quiero probarlo a mí eso me marea’’ es la gran frase de muchas personas que cierran la puerta a la realidad virtual por culpa de la cinetosis y el miedo producido por ella.

En adelante se encuentran los casos o reglas a tener en cuenta y cómo evitar o paliar la cinetosis en la medida de lo posible.

- **El usuario es dueño de la cámara**

  Los desarrolladores deben evitar quitar el control de la cámara a los usuarios. Mover la cámara sin que los usuarios lo hagan conscientemente rompe la inmersión y rápidamente provoca molestias.

  Es una regla de oro, no importa el motivo, el control de la cámara no pertenece a otro sino al usuario.

  Si se quiere que el usuario no se pierda algún evento que está ocurriendo en escena deberán buscarse estrategias para captar su atención. Por ejemplo, con la luz de la escena o los sonidos, puede que hasta un fecha flotante sirva. No

obstante, cabe destacar que hoy en día este problema no tiene una solución definitiva.

- **Velocidad controlable**

  Tal vez el usuario aún no tenga sus piernas virtuales pero tiene claro querer conseguirlas. Lo adecuado es proporcionarle una opción para reducir la velocidad con la que se mueve por el mundo, para que así su adaptación le sea más amena y reducir los efectos de la cinetosis. Siempre hay que tener esto en cuenta intentando que no afecte a las mecánicas de la experiencia.

- **Percepción de la aceleración**

  Una buena manera de evitar el efecto de la cinetosis es reduciendo la percepción de aceleración del usuario. La parte más alejada del centro de visión, llamada la visión periférica, es una pieza muy importante para esta percepción. Al perder visión periférica el usuario pierde la referencia de movimiento y por tanto información sobre su velocidad. Para aprovechar este hecho es bueno dejar como opción de los desplazamientos una versión en la cual la vista periférica del usuario se reduce, por ejemplo, tapando lo que ve con un efecto negro en la periferia. Como siempre el dejar este efecto como una opción nos permite hacer que el usuario sienta que tiene el control y pueda elegir de qué manera se mueve, pues este efecto es muy molesto para usuarios que tengan piernas virtuales.

- **Colisiones con la cámara**

  Uno de los grandes problemas de la realidad virtual es qué hacer cuando algo colisiona con nuestra cabeza. Bajo ninguna circunstancia se debe permitir que tenga comportamiento de físicas, pues las físicas producirán que la cámara se mueva en contra de los movimientos reales del usuario y por tanto provocándole cinetosis. ¿Entonces hacemos que la cámara ignore cualquier tipo de colisión o física? Pero, ¿y si el usuario moviera la cabeza hacia una pared intencionadamente? Los desarrolladores han optado por dos opciones: traspasar

y cegar al usuario mientras esté en contacto con objetos sólidos o alejar la pared según el usuario se acerca.

Ambas opciones tienen inconvenientes, por ejemplo, la segunda opción puede producir cinetosis, al fin y al cabo, no es un movimiento natural por mucho que el usuario lo haga conscientemente.

Y en la opción en la que traspasamos, no hay nada que impida que hagamos trampas y pasemos a través de las paredes, cosa que en muchos casos no debería permitirse. Lo que lleva al problema de las trampas, el cual se hablará más adelante en el apartado 6.5.

- **Fundidos en negro**

  Consiste en hacer una pequeña transición, normalmente con un fundido en negro en la visión del usuario, antes de hacer cualquier tipo de movimiento instantáneo como puede ser el de teleport o el de cambio a otra escena.

  Esto reduce la impresión del usuario al ver de repente cambiada su realidad pues tal vez no era consciente de que dicho cambio se iba a producir.

  Un consejo especial es la idea de hacer un efecto visual como el de cerrar los ojos dando una sensación más natural a un fundido en negro. Este efecto se conseguiría haciendo que el fundido comience en la parte superior e inferior de la vista del usuario y oscureciendo hacia el centro.

- **No mover horizonte**

  Esta advertencia consiste en la cinetosis ocurrida cuando el usuario cree sentir la fuerza de la gravedad distinta a la del mundo real, provocada por una inclinación del horizonte virtual en un ángulo fijo que no coincida con el mundo real o si el suelo que pisa es muy amplio y inclinado.

  La manera de solucionarlo parece sencilla simplemente no inclinar el horizonte y prestar especial atención si algún terreno inclinado de nuestra experiencia es especialmente amplio. Pero a veces este problema no puede ser controlado por el desarrollador, siendo un problema provocado por el tracking de nuestro visor.

Ante esto no tenemos más opción que cerrar la experiencia y ajustar el tracking de nuestro dispositivo.

Especial atención debe ponerse en situaciones como aquellas en las que el suelo se inclina y está en constante movimiento. Un ejemplo de esta situación podría ser la superficie de un barco surcando las olas. Estas situaciones pueden provocar cinetosis rápidamente. Las experiencias de las montañas rusas suelen producir cinetosis por esta misma causa.

- **Vección: ilusión del movimiento**

  A diferencia de la vida real, los usuarios de realidad virtual no pueden confiar en su sistema vestibular para determinar si se están moviendo. En su lugar, deben emitir un juicio utilizando sólo información visual. En algunos casos, los usuarios pueden no diferenciar cuándo los objetos se están moviendo y cuándo están quietos, pensando que ellos son los que se mueven. Esta percepción ilusoria del movimiento propio producida por la visión se denomina vección. La vección consiste, de esta forma, en que aun estando quietos, nuestro cerebro y nuestra visión nos hacen creer que nos movemos. Un ejemplo real de vección tiene lugar cuando una persona se encuentra sentada en un coche, observando el coche que se encuentra delante del suyo; ese coche al ir a más velocidad se aleja, dando la sensación de que su propio coche está viajando hacia atrás.

  Una manera de paliar este efecto es poner en el entorno referencias visuales; no es lo mismo un suelo monocromo que un suelo con textura. Una pradera vacía o una con arbustos y hierbas silvestres.[11].

5

Especificación y Análisis

En este capítulo se mostrarán los requisitos del proyecto, separándolos en dos grupos los iniciales y los requisitos incluidos durante la ejecución del proyecto. Estos dos grupos a su vez estarán divididos en funcionales y no funcionales. También se comentarán los cambios producidos en algunos de los requisitos según avanzaba el desarrollo y el porqué de dichos cambios.

5.1 Requisitos iniciales

Estos son los requisitos que aparecen recogidos en el anteproyecto.

Requisitos funcionales:

ID	NOMBRE	DESCRIPCIÓN
RIF01	Interacciones básicas	Poder coger, manipular y lanzar objetos.
RIF02	Interactuables	Poder ponerse y quitarse sombreros. Accionar botones y palancas. Interactuar con mando de control.

RIF03	Diferentes desplazamientos	Poder desplazarse en la experiencia mediante teletransporte y movimiento suavizado.
RIF04	Brazalete holográfico	Desarrollar una interfaz de un brazalete que muestra un menú holográfico en el que poder cambiar los ajustes, ver la consola de debug, ver la memoria y salir de la experiencia.
RIF05	Casco astronauta	Desarrollar una interfaz de un casco espacial que muestra información del nivel de oxígeno y distancia de la estación de recarga.
RIF06	Lentes de contacto	Dar la capacidad al usuario de ver cosas por encima de la visión natural humana a modo de realidad aumentada, destacando objetos en el entorno o obteniendo información adicional de ellos.
RIF07	Ajustes de usuario	Poder configurar y guardar los ajustes de movimiento e idioma del usuario.
RIF08	Memoria	Introducir una copia modificada de la memoria en la experiencia, permitiendo su lectura.

Requisitos no funcionales:

ID	NOMBRE	DESCRIPCIÓN
RINF01	Entorno de laboratorio	Acondicionar la escena dentro de la experiencia con los modelos y sonidos apropiados para dar la inmersión de estar en un laboratorio tecnológico.
RINF02	Preparación didáctica	Distribuir paneles informativos por la escena a modo de resumen de esta memoria y los controles de la experiencia.
RINF03	Facilidad de integración de idiomas	Estructurar el sistema de multilenguaje de manera que puedan añadirse más idiomas a la experiencia sin necesidad de realizar una nueva versión.
RINF04	Multidiomas	Preparar la experiencia para idioma español e inglés.

5.2 Requisitos incluidos durante la ejecución del proyecto

Estos son los requisitos que surgieron conforme iba avanzando el desarrollo y los cambios en los requisitos iniciales que fueron modificados.

Requisitos funcionales:

ID	NOMBRE	DESCRIPCIÓN
RIDEF01	Gafas de realidad aumentada	RIF06 modificado. Ya no forma parte de la visión natural del usuario, sino que se vuelve un elemento más de la escena, pudiendo quitarlas y ponerlas.
RIDEF02	Interactuables	RIF02 modificado. Se reduce a poder ponerse sombreros y gafas. Desarrollar un escáner de controladores para accionar elementos del entorno.
RIDEF03	Movimiento por fuerzas	Desarrollar un sistema de propulsión en los controladores con los que moverse por el espacio.

Requisitos no funcionales:

ID	NOMBRE	DESCRIPCIÓN
RIDENF01	Entorno estación espacial	RINF01 modificado. Acondicionar la experiencia a la de una estación espacial, añadiendo un fondo apropiado y efecto de gravedad cero en los elementos.
RIDENF02	No poder traspasar paredes	Gestionar las colisiones para evitar que el usuario pueda traspasar objetos sólidos pero evitando provocar cinetosis.
RIDENF03	No permitir trampas	Gestionar las entradas y colisiones del controlador para evitar que el usuario realice movimientos no controlados, como teleportarse fuera de la estación espacial.

Cambio de requisitos:

El cambio de requisitos más visible es el denominado Entorno de estación espacial (RIDENF01) que fue una modificación del requisito Entorno de laboratorio (RINF01). Dicha modificación consiste en que el entorno pasaría de estar acondicionado en un ambiente de laboratorio a uno de estación espacial.

Los motivos de dicho cambio fueron:

- La dificultad encontrada al buscar modelos gratuitos de laboratorio encontrando más fácilmente algunos ambientados en el espacio.

- La idea de que ya que es un mundo virtual, ¿por qué no transportar al usuario a vivencias más fantásticas? Como puede ser ir al espacio profundo.

- La idea de objetos en gravedad cero.

- Una manera mucho más sutil de introducir el casco de astronauta en la ambientación.

Otro de los requisitos que fueron modificados fue Lentes de contacto (RIF06) que se modificó a Gafas de realidad aumentada (RIDEF01). Esta modificación consiste en transformar la idea conceptual y otorgarle presencia en la experiencia siendo algo con lo que interactuar y no una capacidad del usuario.
Los motivos de dicho cambio fueron:

- Atrasar la aparición de esta característica para más así facilitar su comprensión, dedicando un espacio de la experiencia exclusivamente a ella.

- Aprovechar características ya desarrolladas para otras mecánicas de la experiencia (poder ponerse y quitarse gorros, requisito RIF02).

- -Evitar ideas confusas, como la de llevar unas lentillas, volviendo lo material y dándole una forma más interactuable en la experiencia, esperando facilitar así la comprensión de su uso.
-

Por último, el requisito Interactuables (RIF02) que pasó a ser (RIDEF02) su modificación consiste en reducir el requisito cambiando los botones, palancas y mando de control con lo que interactuar por un único elemento, el escáner de controlador.

Los motivos de dicho cambio fueron:

- Tras ver la velocidad a la que se estaba desarrollando la aplicación se decidió reducir la carga de trabajo de este requisito y así poder cumplir las fechas establecidas sin afectar a las horas acordadas, pues otras partes del desarrollo estaban necesitando mucho más tiempo.
- Por ser el requisito que menos aportaba.

6

Diseño e Implementación

6.1 Estructura de la base de datos

La base de datos cumple un papel fundamental en la preservación de la información de la experiencia. Su objetivo es poder almacenar los siguientes datos:

- Todos los textos en todos los idiomas que intervienen en la experiencia (Text).
- Los datos de configuración del usuario (PlayerSettings).
- -La posición y rotación de un objeto interactuable en referencia a un expositor si es que ese interactuador en concreto pudiera exponerse en ese expositor en concreto. (RelationExpositorInteractable).
-

Estructura de las tablas en formato SQL:

```
CREATE TABLE IF NOT EXISTS "PlayerSettings" (
    "activateControlRotation"    INTEGER NOT NULL,
    "rotationValue"    INTEGER NOT NULL,
    "MovementType"    INTEGER NOT NULL,
    "MovementValue"    INTEGER NOT NULL,
    "languaje"    TEXT NOT NULL
);
```

```
CREATE TABLE IF NOT EXISTS "Text" (
    "TextKey"    TEXT NOT NULL,
    "Languaje"    TEXT NOT NULL,
    "Content"    TEXT NOT NULL,
    PRIMARY KEY("Languaje","TextKey")
);
CREATE TABLE IF NOT EXISTS "RelationExpositorInteractable" (
    "ExpositorKey"    TEXT NOT NULL,
    "InteractableKey"    TEXT NOT NULL,
    "Position_X"    TEXT NOT NULL,
    "Position_Y"    TEXT NOT NULL,
    "Position_Z"    TEXT NOT NULL,
    "Rotation_X"    TEXT NOT NULL,
    "Rotation_Y"    TEXT NOT NULL,
    "Rotation_Z"    TEXT NOT NULL,
    PRIMARY KEY("InteractableKey","ExpositorKey")
);
```

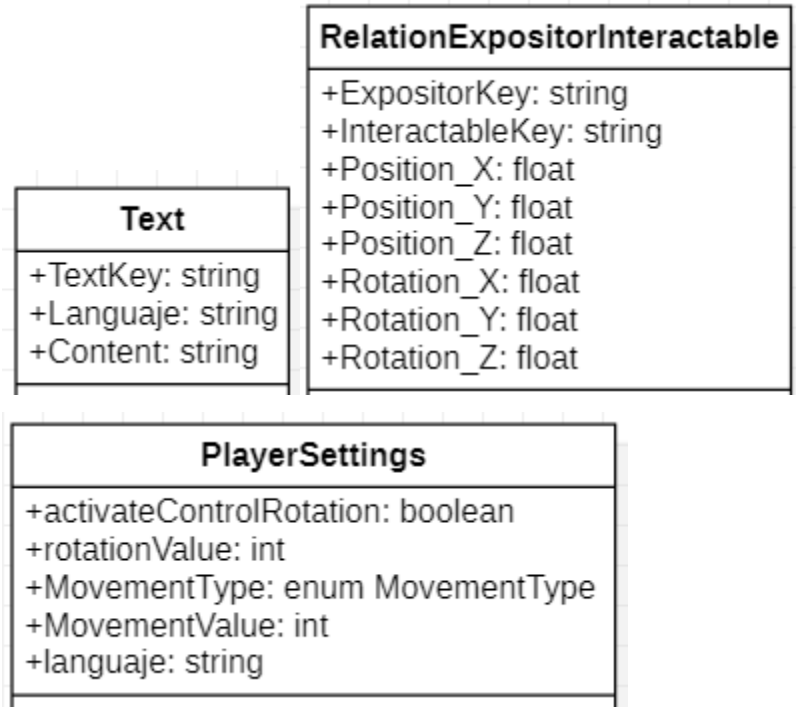

Figura 6.1: Tablas de la base de datos con sus atributos ya casteados.

Tanto los atributos de PlayerSettings como los de RelationExpositorInteractable son de solo lectura y jamás serán modificados en ejecución.

6.2 Jerarquía de carpetas del proyecto Unity

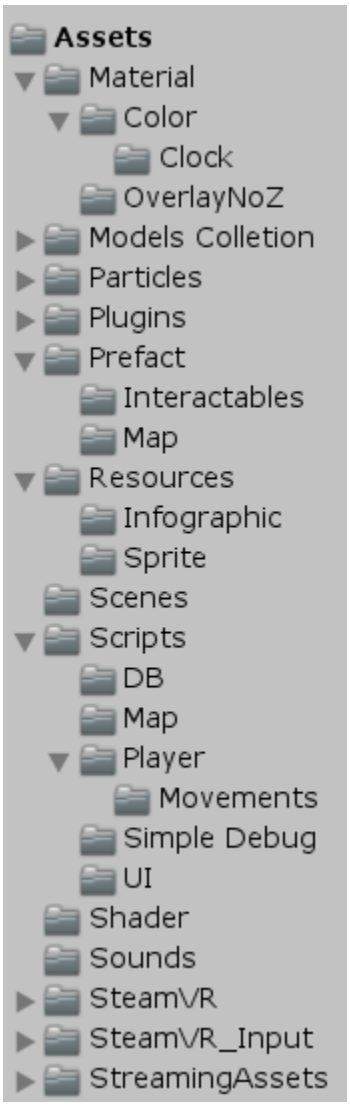

Figura 6.2: Captura de la jerarquía de carpetas del proyecto Unity.

A continuación, se explicarán los diferentes contenidos para dichas carpetas para así poder entender su significado:

- Material: En esta carpeta se guardan todos los archivos con formato .mat.

- Material/Color: Los archivos con formato .mat que son monocromo Opaque.

- Material/Color/Clock: Los archivos .mat monocromo Opaque que forman parte de el modelo Reloj (Clock).

- Material/OverlayNoz: Los archivos con formato .mat que se usan en el shader overlayNoz.

- Models Collection: En esta carpeta se guardan todos los archivos pertenecientes a assets descargados con la intención de aprovechar sus modelos.

- Particles: En esta carpeta se guardan todos los archivos que conforman los efectos de partículas.

- Plugins: En esta carpeta se guardan todos los archivos pertenecientes a assets descargados con la intención de usar sus scripts y el skybox.

- Prefact: En esta carpeta se guardan todos los archivos con formato .prefab.

- Prefact/Interactables: Los archivos con formato .prefab que heredan del prefab Interactuable.

- Prefact/Map: Los archivos con formato .prefab que son elementos del entorno.

- Resources/Infographic: En esta carpeta se guardan todas las imágenes de la memoria.

- Resources/Sprite: En esta carpeta se guardan todos los archivos pertenecientes a imágenes 2D, exceptuando la de la memoria.

- Scenes: En esta carpeta se guarda el archivo .unity de la escena de Unity.

- Scripts: En esta carpeta se guardan todos los archivos con formato .cs.

- Scripts/BD: Los archivos con formato .cs que forman la gestión de la base de datos.

- Scripts/Map: Los archivos con formato .cs que están destinados a elementos del entorno.

- Scripts/Player: Los archivos con formato .cs que forman parte de las muchas partes del jugador: cabeza, manos, etc.

- Scripts/Player/Movement: Los archivos con formato .cs que gestionan las diferentes maneras de moverse por el entorno: teleport, movimiento suavizado y propulsión.

- Scripts/Simple Debug: Los archivos con formato .cs que sirven para dar algún tipo de respuesta simple por consola o por interfaz.

- -Scripts/UI: Los archivos con formato .cs que componen las diferentes interfaces de realidad virtual así como el gestor de idiomas.

- Shader: Los archivos con formato .shader de tipo overlayNoz.

- Sound: En esta carpeta se guardan todos los archivos con formato .mp3 y .wav.

- SteamVR y SteamVR_Input y StreamingAssets: Carpetas que contienen todos los datos del plugin de SteamVR Plugin.

6.3 Manipulación simple de objetos

La realidad virtual tiene una gran cualidad, la inmersión. A lo largo de esta memoria se muestra cómo intentar aumentarla lo máximo posible y evitar perderla. La inmersión no es solo poder mirar al alrededor y sentir que realmente se está en otro lugar, una auténtica inmersión necesita presencia y la mejor manera de conseguirla es un entorno reactivo, permitiendo manipularlo y interactuar con él.
En este apartado, veremos las dos clases principales con las que interactuamos con el entorno a través de los controladores y las partes del código más relevantes.

Clase HandVR: Es la clase que hereda de MonoBehaviour, gestiona las variables que tienen que ver con nuestros controladores. Por ejemplo, si es la izquierda o derecha, o si tienes un objeto agarrado. Esta clase se encuentra como un componente en cada uno de los controladores por separado.

En la figura 6.3 podemos ver una de sus variables llamada hoveringInteractable de tipo InteractableVR. En ella almacenamos el objeto que está en contacto con el controlador.
Como vemos en la figura 6.3 cuando la variable hoveringInteractable toma un nuevo valor, es la encargada de comunicar tanto al propio objeto como al controlador que un objeto nuevo entró en contacto con él o dejó de estarlo, enviando unos mensajes a dichos GameObjects para que cualquier componente pueda reaccionar a este evento.

```
private InteractableVR _hoveringInteractable;
public InteractableVR hoveringInteractable
{
    get { return _hoveringInteractable; }
    set
    {
        if (_hoveringInteractable != value)
        {
            if (_hoveringInteractable != null)
            {
                _hoveringInteractable.SendMessage("OnHandHoverEnd", this,
                    SendMessageOptions.DontRequireReceiver);

                if (_hoveringInteractable != null)
                {
                    // let objects attached to the hand know that a hover has ended
                    this.BroadcastMessage("OnParentHandHoverEnd", _hoveringInteractable,
                        SendMessageOptions.DontRequireReceiver);
                }
            }

            _hoveringInteractable = value;

            if (_hoveringInteractable != null)
            {
                _hoveringInteractable.SendMessage("OnHandHoverBegin", this,
                    SendMessageOptions.DontRequireReceiver);
                //touchSound.Play();

                if (_hoveringInteractable != null)
                {
                    // let objects attached to the hand know that a hover has begun
                    this.BroadcastMessage("OnParentHandHoverBegin", _hoveringInteractable,
                        SendMessageOptions.DontRequireReceiver);
                }
            }
        }
    }
}
```

Figura 6.3 : Captura de la variable hoveringInteractable de la clase HandVR.

Las condiciones que tiene que cumplir un objeto de la escena para considerarse en la variable hoveringInteractable son:

- Tener un componente InteractableVR.
- Tener un componente Collider en alguno de sus hijos.

Además de las condiciones del objeto, el controlador también tiene sus propias condiciones para permitir que un objeto se considere en contacto con él, como por ejemplo que la mano esté bloqueada (couldHandInteractuate()). Véase la figura 6.4.

```
public void OnTriggerEnter(Collider other)
{
    if (couldHandInteractuate()) {
        InteractableVR interactable = other.GetComponentInParent<InteractableVR>();
        if (interactable)
        {
            if (((1 << other.gameObject.layer) & holoMask) == 0){
                if (!hoverLocked)
                {
                    vibrationAction.Execute(0, 0.1f, 0.2f, 1f, handType);
                    hoveringInteractable = interactable;
                }
            }
        }
        else
        {
            Expositor ex = other.gameObject.GetComponentInParent<Expositor>();
            if (ex) hoveringsExhibitors.Add(ex);
        }
    }
}
```

Figura 6.4 : Captura de la función OnTriggerEnter de la clase HandVR.

La variable currentAttachedObject es en comportamiento idéntica a hoveringInteractable, salvo que esta almacena el objeto que está siendo agarrado por el controlador. Ver figura 6.5 .

```
private AttachedObject _currentAttachedObject;

public AttachedObject currentAttachedObject { get { return _currentAttachedObject; }
    protected set {

        if (_currentAttachedObject.attachedObject != value.attachedObject)
        {
            if (_currentAttachedObject.attachedObject != null)
            {
                _currentAttachedObject.attachedObject.SendMessage("OnDetachedFromHand", this,
                    SendMessageOptions.DontRequireReceiver);

                if (_currentAttachedObject.attachedObject != null)
                {
                    this.BroadcastMessage("OnParentDetachedFromHand", _currentAttachedObject.attachedObject,
                        SendMessageOptions.DontRequireReceiver);
                }
            }
            _currentAttachedObject.attachedObject = value.attachedObject;
            _currentAttachedObject.interactable = value.interactable;

            if (_currentAttachedObject.attachedObject != null)
            {
                _currentAttachedObject.attachedObject.SendMessage("OnAttachedToHand", this,
                    SendMessageOptions.DontRequireReceiver);
                if (_currentAttachedObject.attachedObject != null)
                {
                    this.BroadcastMessage("OnParentAttachedToHand", _currentAttachedObject.attachedObject,
                        SendMessageOptions.DontRequireReceiver);
                }
            }
        }
    }
}
```

Figura 6.5 : Captura de la variable currentAttachedObject de la clase HandVR.

Las condiciones que deben cumplirse para que currentAttachedObject tome un nuevo valor son:

- El usuario realice la acción de agarrar con las entradas del controlador específico.
- hoveringInteractable debe tener un valor distinto de null, que significa que hay algún objeto que agarrar.
- La mano no esté bloqueada.

Una vez cumplidas estas condiciones, ejecutaremos la función GrabObject() la cual podemos ver en la figura 6.6 .

```
private void GrabObject()
{
    //Soltarlo de la otra mano
    if (hoveringInteractable != null)
    {
        if (otherHand.Equals(hoveringInteractable.attachedToHand))
        {
            otherHand.ReleaseObject();
        }
    }
    //Si esta en un exhibitor cortar su relacion con el
    Expositor exp = hoveringInteractable.GetComponentInParent<Expositor>();
    if (exp)
    {
        Rigidbody rg = hoveringInteractable.GetComponent<Rigidbody>();
        if (rg)
        {
            rg.isKinematic = false;
            hoveringInteractable.gameObject.transform.SetParent(null);
            exp.leaveExhibitor();
        }

    }
    //Guardar el objeto
    AttachedObject att = new AttachedObject();
    att.attachedObject = hoveringInteractable.gameObject;
    att.interactable = hoveringInteractable;
    currentAttachedObject = att;
    hoveringInteractable = null;
    var joint = AddFixedJoint();
    joint.connectedBody = currentAttachedObject.attachedObject.GetComponent<Rigidbody>();
    hoverLocked = true;

        if (currentAttachedObject.interactable.hideHandOnAttach)
            Hide();
    if (velocityEstimator != null)
        velocityEstimator.BeginEstimatingVelocity();
}
```

Figura 6.6 : Captura de la función GrabOject de la clase HandVR.

Cuando el usuario culmine su acción de agarre se ejecutará la función para soltar el objeto. Esta función se llama ReleaseObject y podemos verla en la figura 6.7 .

```
public void ReleaseObject()
{
    if (GetComponent<FixedJoint>())
    {
        GetComponent<FixedJoint>().connectedBody = null;
        Destroy(GetComponent<FixedJoint>());
    }
    if (currentAttachedObject.interactable.hideHandOnAttach)
        Show();

    int cont = 0;
    bool found = false;
    while (!found && cont < hoveringsExhibitors.Count)
    {
        if (hoveringsExhibitors[cont].isExhibitorOf(currentAttachedObject.interactable)) {
            found = true;
        }
        else
        {
            cont++;
        }
    }
    if (found)
    {
        hoveringsExhibitors[cont].toExhibit(
            currentAttachedObject.attachedObject,currentAttachedObject.interactable);
    }
    else
    {
        if (velocityEstimator != null)
        {
            velocityEstimator.FinishEstimatingVelocity();
            currentAttachedObject.attachedObject.GetComponent<Rigidbody>().velocity =
                velocityEstimator.GetVelocityEstimate();
            currentAttachedObject.attachedObject.GetComponent<Rigidbody>().angularVelocity =
                velocityEstimator.GetAngularVelocityEstimate();
        }
    }
    hoveringInteractable = currentAttachedObject.interactable;
    currentAttachedObject = new AttachedObject();
    hoverLocked = false;
}
```

Figura 6.7 : Captura de la función ReleaseOject de la clase HandVR.

Con esto podremos agarrar y soltar objetos.

Pero además gracias al componente de steamVR VelocityEstimator podemos calcular la aceleración de la mano y lograr con ello que el objeto se lance al soltarlo.

Clase InteractableVR: Es la clase que hereda de MonoBehaviour que nos indica si se puede interactuar con un GameObject, también almacena si algún controlador está en

contacto con ella o agarrándola y qué controlador es. Así si algún componente necesitase dicha información tan solo tendría que comprobarla en esta clase.

Como vemos en la figura 6.8 esta clase implementa las funciones usadas por hoveringInteractable y currentAttachedObject para gestionar sus datos.
Es en esta clase también donde controlamos que cuando el objeto esté en contacto con el controlador se remarque con un borde de color, dicha variable es highlight.
Highlight nos ayuda a diferenciar cuál es el objeto que será agarrado cuando hagamos la acción de agarrar, pues sin esta ayuda, en el caso de que muchos objetos estuvieran juntos nunca sabríamos cuál de ellos es el que exactamente está interactuando.

```
public void OnHandHoverBegin(HandVR hand)
{
    isHovering = true;
    hoveringHand = hand;
    highlight.OutlineMode = Outline.Mode.OutlineVisible;
    highlight.OutlineColor = highlightColor;
    highlight.enabled = true;
}

public void OnHandHoverEnd(HandVR hand)
{
    isHovering = false;
    hoveringHand = null;
    highlight.enabled = false;
}

protected virtual void OnAttachedToHand(HandVR hand)
{
    attachedToHand = hand;
}

protected virtual void OnDetachedFromHand(HandVR hand)
{
    attachedToHand = null;
}
```

Figura 6.8 : Captura de la funciones llamadas por HandVR de la clase InteractableVR.

Todos estos fragmentos de código se ha desarrollado gracias al ejemplo de los componentes que pone a disposición SteamVR y a los tutoriales de la web raywenderlich [12].

6.4 Interfaz holográfica

Vamos a ver ahora la interfaz más importante del proyecto: el brazalete que esconde un menú 2D holográfico cuando lo miramos. En este menú podremos:

- Ver la consola de depuración.
- Consultar la infografía.
- Cambiar los ajustes de idioma y movimiento.
- Salir de la experiencia.

Como ya se mencionó en el apartado 3.2, esta interfaz está inspirada en la del menú de la experiencia To the top.

Para hacer esta interfaz se tuvo que crear una clase abstracta llamada UI_ElementVR la cual hereda de MonoBehaviour. Esta clase contiene las funciones básicas que un elemento reactivo de una interfaz puede tener, como vemos en la Figura 6.9 .

```
public abstract class UI_ElementVR : MonoBehaviour
{

    public abstract void hoverBegin();
    public abstract void hoverStay();
    public abstract void hoverEnd();
    public abstract void Onclick();
}
```

Figura 6.9 : Captura de la clase abstracta UI_ElementVR.

De ella hereda la clase ButtonVR que es similar a la clase Button que proprociona Unity, pero que reacciona ante los eventos de realidad virtual de la aplicación.

Finalmente, la clase UI_Interactuator es la encargada de poder navegar por nuestra interfaz holográfica.

UI_Interactuator hereda de MonoBehaviour y se encarga de gestionar el puntero láser que nos permite interactuar con la interfaz. Su implementación está hecha con Raycast. Nos permite pulsar nuestros botones holográficos y indicar en qué lugar del holograma se está apuntando. También gestiona la desaparición de dicho láser en caso de no estar apuntando a la interfaz. Esta clase en realidad trabaja con cualquier clase que hereda de UI_ElementVR por lo que está preparada para futuras ampliaciones. Véase la Figura 6.10 .

La combinación de estas tres clases nos permite saber si el puntero se encuentra sobre un ButtonVR cambiando su color y si el evento de clic se ha ejecutado, con un momentáneo cambio de color. Tanto el color original como el de bloqueo, el de marcado y el de clic son seleccionables desde el editor de Unity en el componente ButtonVR.

```
// Update is called once per frame
void Update()
{
    bool ishitUI = false;
    RaycastHit hit;
    if (Physics.Raycast(controllerPoseHand.transform.position, transform.forward, out hit, 50))
    {
        if (((1 << hit.collider.gameObject.layer) & UIMask) != 0)
        {
            ishitUI = true;
            laser.enabled = holoMenu.activeInHierarchy || hit.collider.gameObject.GetComponent<InfoButton>();
            laser.SetPosition(0, controllerPoseHand.transform.position);
            laser.SetPosition(1, hit.point);

            UI_ElementVR elementVR = hit.collider.gameObject.GetComponentInParent<UI_ElementVR>();

            if (elementVR == elementHoverCurrent){
                if (elementHoverCurrent)
                {
                    elementHoverCurrent.hoverStay();
                }
            }
            else
            {
                if (elementHoverCurrent)
                {
                    elementHoverCurrent.hoverEnd();
                }
                elementHoverCurrent = elementVR;
                if (elementHoverCurrent)
                {
                    elementHoverCurrent.hoverBegin();
                }
            }
            if (selectAction.GetStateDown(handType) && elementHoverCurrent)
            {
                elementHoverCurrent.Onclick();
            }
        }
    }
    if (!ishitUI)
    {
        //hide laser
        laser.enabled = false;
    }
}
```

Figura 6.10 : Captura de la función Update de la clase UI_Interactuator.

Las Figuras 6.12 ,6.13 ,6.14, 6.15 y 6.16 muestran el diseño de las pantallas interfaz holográfica, mientras que la Figura 6.11 muestra el diagrama de navegación.

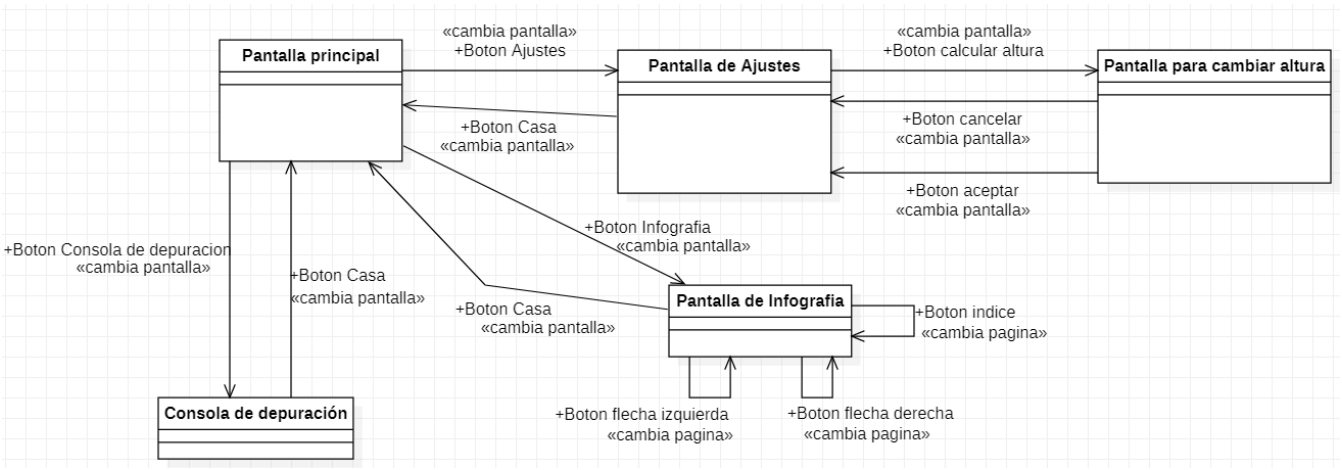

Figura 6.11 : Diagrama de navegación del menú holográfico.

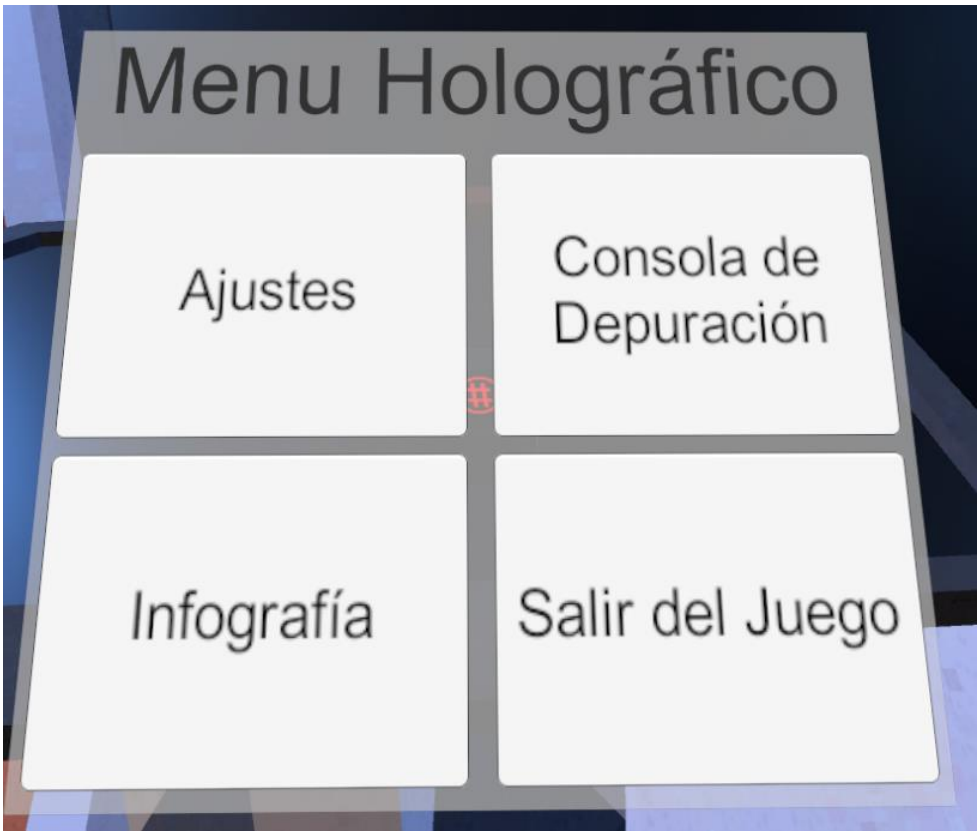

Figura 6.12 : Captura de la experiencia, pantalla principal.

Figura 6.13 : Captura de la experiencia, consola de depuración.

Figura 6.14 : Captura de la experiencia, pantalla de ajustes.

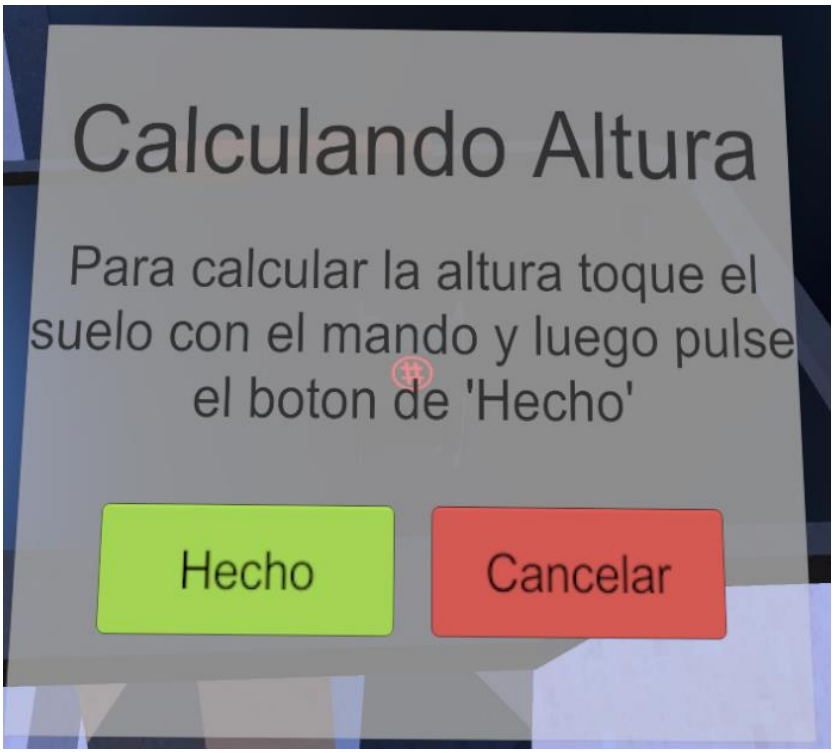

Figura 6.15 : Captura de la experiencia, pantalla para cambiar altura.

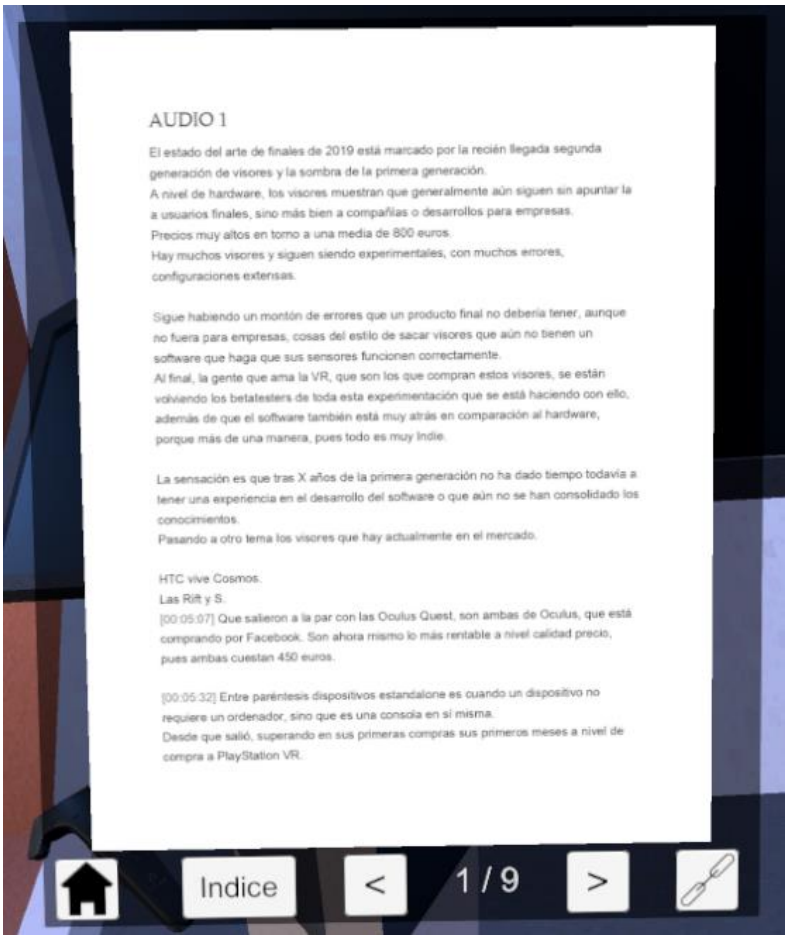

Figura 6.16 : Captura de la experiencia, pantalla de infografía.

6.5 Evitar las Trampas

Anteriormente en el capítulo 4, se mencionaron, como buenas prácticas, el tratamientos de las colisiones con la cabeza y el movimiento real frente al virtual. Es aquí donde se comentará la solución que se implementó ante dichos problemas desde el punto de vista de evitar que el usuario no pueda evadir la dinámica de la experiencia.
Lo que intentamos evitar en este apartado es:

- Que el usuario pueda atravesar ciertos objetos sólidos como pared, suelos, techos, puertas, etc.
- La máxima cinetosis posible, aunque el usuario conscientemente esté intentado hacer trampas.
- Errores en el tracking de manos que puedan ocasionar lo anterior indirectamente.

- Que el usuario pueda ver desde dentro de los modelos.
- Que el usuario en ningún momento piense que la experiencia se ha congelado o detenido.

Para esto se llevaron a cabo cuatro pasos: La ceguera, el bloqueo de controladores, los colliders especiales y el teleport oculto.

- **La ceguera**

Es algo muy simple, tanto los controladores como la cabeza del usuario no reaccionan a las físicas de Unity. Cada vez que la cabeza del usuario estuviera en colisión con un objeto como, por ejemplo, paredes, puertas, etc., su visión se volvería totalmente oscura y le aparecería un mensaje indicando que no es un error del visor, sino que está fuera de los límites de la experiencia.

- **El bloqueo de controladores**

Mismo principio de la ceguera, esta vez en los controladores. Un controlador que estuviera en colisión con un objeto sólido produciría que el controlador bloqueara las acciones del usuario, como por ejemplo la de moverse.
Además de eso, si entre la cabeza del usuario y el controlador hay algún objeto de los indicados, también se bloqueará aunque el controlador no estuviera en contacto con el objeto. Este caso es para cuando el usuario pasa su controlador a través de la puerta hasta el otro lado e intenta, por ejemplo, teletransportarse.
Además, si el usuario se encuentra bajo el efecto de la ceguera anteriormente explicada también tendría bloqueado sus controladores.

- **Collider especiales**

Tanto bajo el suelo de la nave como por encima del techo hay un enorme objeto invisible que se comporta igual que las paredes, son muy anchos por lo que por muy alta que fuera una persona no perdería su efectividad, estos sirven por si el usuario intentará colarse bajo el suelo en otra sala haciendo uso de la configuración de altura.

- **Teleport oculto**

Intenta evitar que el movimiento real del usuario le permite traspasar paredes con simplemente unos pasos y inclinar la cabeza al mismo tiempo. Consiste en guardar la posición de la cabeza constantemente excepto si está bajo efecto de la ceguera explicada anteriormente. Con esto conseguimos la última posición en la que su cabeza estuvo antes de colisionar. Una vez tenemos esta variable, vamos comprobando que en su estado de ceguera no aleja la cabeza de dicho punto más de cierto valor constante. Si no lo hace, simplemente habría colisionado brevemente con algún elemento sólido, pero si realmente se alejara de dicho punto, se considera que está intentando traspasar la pared o algún tipo de trampa similar y, aprovechando su ceguera se teletransporta al usuario al punto de nuestra variable en el que aún no había colisionado. Para el usuario es una sensación natural como si traspasará una cortina oscura, pero cuando mira a su alrededor nada ha cambiado desde que perdió la visión.

7

Pruebas

Llega el momento de probar la aplicación, comprobar que todo lo explicado anteriormente da resultados en usuario finales. En este capítulo no se hablará de las muchas correcciones que se realizaron durante el desarrollo, ni de los cientos de intentos de traspasar paredes para evitar las trampas por parte del usuario. Este capítulo se centrará en las pruebas, los formularios que se prepararon para los testers y sus resultados.

7.1 Evaluación

La evaluación que se preparó por cada usuario de pruebas, a partir de ahora testers, constó de las siguientes etapas:

1. Rellenar de forma anónima un primer formulario pre-experiencia, para identificar al tester y comprobar si es el tipo de usuario objetivo de este proyecto.
2. Se le informa sobre el paso siguiente. Deberá completar unos objetivos que, según avance en la experiencia, solo si es necesario, se le revelarán: que debe intentar pensar en voz alta, que sobre todo intente disfrutar y olvidarse de la presencia del evaluador.
3. Los objetivos solo son un recurso para que el tester pruebe todas las partes de la experiencia o que se pueda saber cuáles no ha podido completar.

4. El tester prueba la experiencia en presencia del evaluador sin ningún tipo de ayuda. El papel del evaluador en esta parte es solo escuchar y procurar que no deje ninguna parte de la experiencia sin visitar. Además, se tomarán nota de los tiempos que tarda el tester en las diferentes zonas de la experiencia.
5. Cuando el tester haya completado los objetivos y quiera retirarse de la experiencia, se le retirará el visor y se le da unos minutos de descanso.
6. Rellenar de forma anónima un segundo formulario post-experiencia. En este se intenta que se expresen sobre la experiencia y ver qué han podido completar y qué no, etc.

Tras realizar la evaluación individual se reunió a los tester en una sala para hacer un grupo de discusión, con el objetivo de pulir la información de los formularios post-experiencia.

En el anexo I se pueden encontrar los formularios mencionados en este apartado.

7.2 Resultados obtenidos

A continuación se muestran los resultados conseguidos durante las pruebas con testers. Destacar que estas pruebas se realizaron en la etapa final del proyecto, sin estar completado el requisito: Memoria (RIF08), y cada tester la realizó independiente a los demás.

Con este formulario podemos saber si nuestros testers cumplen las condiciones de nuestro público objetivo, en el que tenemos testers como 1, 2 y 5 que sí cumplen dichas condiciones: ser desarrolladores y que no desarrollen para realidad virtual. También tenemos información del perfil del tester que nos puede interesar como, por ejemplo, que todos son de aproximadamente 24 años y que, por lo general, tienen poca experiencia con la realidad virtual.

Mientras los testers probaban la experiencia, se les pidió que pensaran en voz alta. A continuación se recogen las mejoras más importantes que señalaron:

- Atraer objetos desde lejos con los controladores para no tener que agacharse.
- Añadir más sonidos para los choques de objetos con el entorno.
- Añadir un panel explicativo para los distintos tipos de movimiento.
- Explicación de los botones informativos.
- Colocar números encima de las puertas.
- Explicar mejor el funcionamiento de las gafas de realidad aumentada.
- Explicar mejor el casco de astronauta.
- Se detectó un error en el cual un sombrero colocado sobre la cabeza del usuario no se podía quitar con un controlador pero con el otro sí (Tester 2).
- Se detectó un error al morir por asfixia en el espacio, en el cual no permitía moverse tras dicha muerte, impidiendo la continuación de la experiencia.

Ante estos puntos solo se corrigieron los que afectan a los paneles explicativos, se corrigió el error de la muerte por asfixia y se intentó resolver el error ocurrido con el tester 2 pero no se consiguió reproducir.

Tras la prueba los tester recibieron un formulario post-experiencia, además después de que todos los testers involucrados hicieran sus pruebas, se realizó un grupo de discusión con todos, en el que se conversó sobre diferentes temas:

- Nivel de Inmersión
- Apariencia general
- Narrativa
- Realismo de las físicas
- Interacción con objetos
- Menú holográfico
- Gafas de realidad aumentada
- Casco de astronauta

La tabla de tiempos que se ve en la figura 7.1, guarda los valores de tiempo que tardaron los testers en las diferentes salas en las que se divide la experiencia.
Sus filas tienen en siguiente significado:

- **Tester:** Valor identificativo del tester.
- **Fase de introducción:** Tiempo en minutos desde que comienza la experiencia hasta que entra en la nave.
- **Habitación de interacción simple:** Tiempo en minutos desde que entra en la nave hasta abrir la puerta al pasillo del menú holográfico.
- **Pasillo menú holográfico:** Tiempo en minutos desde que abre la puerta al pasillo del menú holográfico hasta que sale del pasillo.
- **Habitación casco astronauta:** Tiempo en minutos desde que entra a la habitación del casco de astronauta hasta que sale tras haber ido al espacio.
- **Habitación gafas de realidad virtual:** Tiempo en minutos desde que entra a la habitación de las gafas de realidad virtual hasta que sale de la misma.
- **Tiempo total:** Tiempo en minutos desde que comienza la experiencia hasta que se quita el visor.

Tester	1	2	3	4	5
Fase de introducción	<1	1	4	1	<1
Habitación de interacción simple	4	4	5	3	1
Pasillo menú holográfico	7	2	2	3	3
Habitación casco astronauta	4	6	7	7	3
Habitación gafas de realidad virtual	<1	6	5	5	3
Tiempo total	20	37	30	25	14

Figura 7.1 : Tabla de tiempos de las pruebas con testers

Las conclusiones que se obtuvieron de todo esto fueron:

- La experiencia se puede completar sin necesidad de descanso.
- Se ha conseguido aminorar los efectos de la cinetosis.
- Se ha conseguido un efecto de inmersión aun a falta de sonidos.
- La experiencia fue entretenida y la temática escogida fue un acierto.

- La pequeña narrativa dentro de la experiencia resultó floja, dos posibles mejoras: no poner nada o colocar cosas que den a entender que la estación espacial está auténticamente dañada.
- Los controles están bien elegidos, pero en el caso del propulsor la postura para propulsarse es incómoda.
- Los paneles explicativos destacaron por no ser muy claros y la importancia que los testers le dieron es muy alta, por lo que es un importante aspecto a mejorar.
- Las interacciones como lanzar objetos o colocarse sombreros resultaron fáciles y fueron lo que más gustó con diferencia.
- Lo que más difícil fue de usar fueron los propulsores y las gafas de realidad aumentada, aunque también gustó bastante.
- La posibilidad de salir al espacio fue un gran acierto.
- La experiencia sin hacer uso de la memoria tiene una duración de unos 25 min.
- Las gafas de realidad aumentada resultaron complejas de usar pero de gran utilidad.

8

Conclusiones y Trabajos Futuros

8.1 Conclusiones

La realidad virtual destinada al usuario ha marcado un punto de inflexión en 2019 con visores de precios asequibles y la salida de experiencias con la suficiente fuerza de marketing como para atraer a las masas, en juegos como Half Life Alyx, Medal of Honor y The Walking Dead.

Las experiencias aún tienen todavía un factor muy experimental/indie, ya que hay una gran ausencia de las grandes empresas de videojuegos aunque parece que esa etapa ya está terminando.

Los desarrolladores cada vez tienen más documentación sobre la manera de diseñar para este nuevo formato muy diferente a los demás, tanto a nivel de narrativa como del propio desarrollo software, con documentos como los de XRA.

El desarrollo realizado en este trabajo de fin de grado ha consistido en: la creación de las tres interfaces, menú holográfico, gafas de realidad aumentada y casco de astronauta. Otros aspectos a destacar son la interacción con el entorno a través de los controladores, la visualización de la memoria desde dentro de la experiencia y un menú

de ajustes en el que poder cambiar tanto la manera en la que nos movemos, como el idioma.

Tras el desarrollo de las múltiples interacciones, interfaces de usuario y el propio entorno, se ha podido ver que tanto los motores de videojuegos como los propios plugins de los visores, no están todavía suficientemente preparados como para aportar una ayuda sustancial en el desarrollo de interacciones sencillas. Esto obliga a crear un sistema propio para interactuar tanto con los elementos del entorno como con los de las interfaces de usuario.

Además de la propia aplicación, la memoria contiene gran parte de los fundamentos teóricos necesarios para la realidad virtual: definición de términos, estado del mercado de visores, buenas prácticas en el diseño, etc.

8.2 Trabajos futuros

En cuanto a las posibles líneas de trabajo que pueden ser continuaciones de este proyecto, se pueden destacar las siguientes:

- El diseño y desarrollo para poder hacer uso de cualquier visor del mercado en una aplicación procurando la menor cantidad de cambios, y realizando una buena arquitectura.

- Añadir nuevas interfaces a la experiencia y hacer un estudio más exhaustivo sobre su diseño, ergonomía, etc., con objetivo de perfeccionar las interfaces de usuario de realidad virtual.

Bibliografía

1. https://noticias.universia.es/ciencia-tecnologia/noticia/2017/03/07/1150198/5-beneficios-usar-realidad-virtual-aula.html
2. https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego)
3. https://es.wikipedia.org/wiki/Microsoft_Visual_Studio
4. https://es.wikipedia.org/wiki/SQLite
5. https://es.wikipedia.org/wiki/Git
6. https://es.wikipedia.org/wiki/Trello
7. https://en.wikipedia.org/wiki/StarUML
8. https://es.wikipedia.org/wiki/GIMP
9. https://xra.org/research-standards/
10. https://knob2001.itch.io/testhmd
11. https://www.dailymotion.com/video/x1esqyd
12. https://www.raywenderlich.com/
13. https://es.wikipedia.org/wiki/Valle_inquietante

14. https://es.wikipedia.org/wiki/Seis_grados_de_libertad

15. https://www.youtube.com/watch?v=ffIZSAZRzDA

Anexo I

Formularios

FORMULARIO PRE-EXPERIENCIA

Tester Nº:_____

He aquí el primero de los dos formularios que tendrás que rellenar, ambos completamente anónimos.
Todas las respuestas dentro de ambos formularios serán publicadas dentro de la memoria de esta experiencia y por tanto públicas para cualquier persona que consulte dicha memoria.
Rellene este formulario con bolígrafo azul o negro, intentando hacer su letra lo más legible posible. Muchas gracias por participar.

-Edad: _____

-¿Dedicas tu tiempo a desarrollar software? Marca la respuesta.

+Sí ( )
+No ( )

-¿Cuál es tu cercanía con la realidad virtual? Marca la respuesta.

+Nunca la he probado antes ( )
+La he probado antes ( )
+Lo he probado bastantes veces ( )
+Tengo mi propio visor ( )
+Desarrollo para Realidad virtual ( )

-¿Qué esperas encontrar en esta experiencia? _____
_____
_____
_____
_____

-¿Tiene algún tipo de problema de visión, auditivo o locomotor? _____
_____
_____
_____
_____

FORMULARIO POST-EXPERIENCIA

Tester Nº:_____

He aquí el segundo formulario.
Todas las respuestas dentro de ambos formularios serán publicadas dentro de la memoria de esta experiencia y por tanto públicas para cualquier persona que consulte dicha memoria.
Rellene este formulario con bolígrafo azul o negro, intentando hacer su letra lo más legible posible. Muchas gracias por participar.

-¿Te sientes desorientado o mareado? Marca la respuesta. Sí ( ) / Un poco( ) / No ( ).

-¿Te ha parecido inmersiva esta experiencia?¿Algo que destacar?:_____
_____
_____.

-¿Te sentiste cómodo con los controles?¿Algo que destacar?:_____
_____.

-¿Los paneles explicativos te fueron de ayuda?¿Algo que destacar?:_____
_____.

-¿Cuánta importancia das a los paneles en la experiencia? Del 1(mínima) al 10(máxima)? ___.

-Marca los objetivos completados con la primera letra de la dificultad que te haya supuesto, Fácil (F), Normal (N) o Difícil (D). No escribas nada si no la has completado:

+Usar el HoloMenú( ) +Cambiar altura ( ) +Navegar por la memoria ( )
+Ponerse sombreros( ) +Lanzar objetos ( ) +Usar las Gafas AR ( )
+Usar el Casco de Astronauta( ) +Usar los propulsores ( ) +Abrir puertas( )

-¿Tuviste alguna dificultad con algún objetivo en particular, desarrolla tu respuesta?_____
_____
_____
_____.

-¿Qué cosas te han gustado más de la experiencia?_____
_____
_____
_____
_____.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA