UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

"Desarrollo de una aplicación para simulación de deficiencia visual mediante Realidad Virtual"

TRABAJO FIN DE GRADO

Julio 2022

AUTOR: Pasqual Martínez Botella
DIRECTOR: Manuel Quesada Martínez

AGRADECIMIENTOS

Agradecer a mi madre, a mis hermanos y a toda mi familia por el apoyo durante toda mi vida académica.

Agradecer a todos los profesores del Grado en Ingeniería Informática en Tecnologías de la Información por el conocimiento que me han transmitido, y que ha hecho que sea capaz de materializarlo en este proyecto. En especial a mi tutor, Manuel Quesada Martínez, por su orientación a la hora de escribir esta memoria y sus opiniones cuando tuve que decidir que rumbo tomaría en varios puntos de este trabajo.

A mis compañeros durante la carrera, con los que he compartido largas horas de trabajo y estudio en las asignaturas.

Agradecer a la Escuela Politécnica Superior de Elche (EPSE) y al Instituto Centro de Investigación Operativa (CIO) de la Universidad Miguel Hernández de Elche por ceder temporalmente el equipamiento necesario para poder desplegar la aplicación de Realidad Virtual aquí desarrollada.

RESUMEN

En este trabajo se aborda la construcción de una aplicación de Realidad Virtual (RV) para la simulación de deficiencia visual. Concretamente, se han abordado defectos visuales no basados en la posición ocular como son las ametropías (miopía e hipermetropía).

Para la gestión del proyecto se ha seguido una metodología siguiendo los principios del Manifiesto Ágil siempre que ha sido posible. Primeramente, se analizó el estado del arte relacionado con los fundamentos que hay detrás de las ametropías para poder desarrollar los métodos que las simulen. Tras ello, se dirigió este análisis a las tecnologías de RV para definir un marco de trabajo que permita implementar una aplicación funcional.

Como resultado se han generado dos artefactos software. Por un lado, un programa ejecutable en C++ para simular las ametropías que hace uso de la librería OpenCV. Por otro lado, un Juego Serio de RV que hace uso de las imágenes simuladas para ofrecer al usuario cinco escenarios con un doble objetivo: (1) concienciar a los usuarios que no sufren este tipo de problemas simulando que los tienen, y (2) realizar test de agudeza visual para detectar usuarios que las padecen sin ser cocientes de ello. La aplicación se ha desplegado en unas Oculus Rift S, permitiendo realizar pruebas preliminares con usuarios controlados.

En conclusión, creemos que hemos desarrollado una aplicación novedosa que hace uso de una tecnología emergente como la RV en la simulación de ametropías. Además, este trabajo abre diferentes vías de trabaja futuro como la puesta en marcha de un piloto con usuarios reales que permita extender y ajustar algunos de los aspectos aquí desarrollados.

Palabras clave: realidad virtual, deficiencia visual, ametropía, miopía, hipermetropía, oculus, opencv

Índice de Contenidos:

1. INTRODUCCIÓN

En este trabajo se pretende construir una aplicación para la simulación de deficiencia visual que pueda ser utilizada a través de la tecnología de Realidad Virtual (RV). En este capítulo se contextualizarán las deficiencias visuales desde el punto de vista de la salud visual y se comentarán los beneficios de la Realidad Virtual en diferentes contextos.

1.1 SALUD VISUAL

En el mundo hay al menos 2200 millones de personas con deficiencias visuales, causadas por diversas razones, entre las que destacan los errores de refracción no corregidos. Se estima que de este tipo de deficiencias, 1000 millones se podrían haber evitado o se pueden solucionar aplicando un tratamiento médico (Organización Mundial de la Salud, 2021). Pese a no tener datos exactos de las deficiencias visuales a nivel mundial en la Figura 1 se observan las deficiencias visuales más populares que podrían haberse evitado siguiendo algún tipo de tratamiento médico (World Health Organisation, 2019).

Figura 1 Estimación de Deficiencias Visuales en el Mundo (World Health Organisation, 2019).

En el contexto de España, según la Encuesta de Discapacidad, Autonomía Personal y Situaciones de Dependencia 2008 realizada por el Instituto Nacional de Estadística, se estima que de 3,8 millones de personas que sufren discapacidades en España, 799,1 mil sufren deficiencias visuales. Esto supone más de un 20% de las personas con discapacidad. De ese 20%, la gran mayoría presenta problemas de visión frente a un pequeño porcentaje que representa a las personas que sufren ceguera. Podemos observar esta comparación en forma de gráficos en Figura 2 y Figura 3 (INE, 2008). Además, como se muestra en la Figura 4 extraída de (Gómez Ulla de Irazazábal & Ondategui-Parra, 2012), el porcentaje de deficiencia visual en el conjunto de la población española es del 2,14% sobre el total, lo que representa casi un millón de personas.

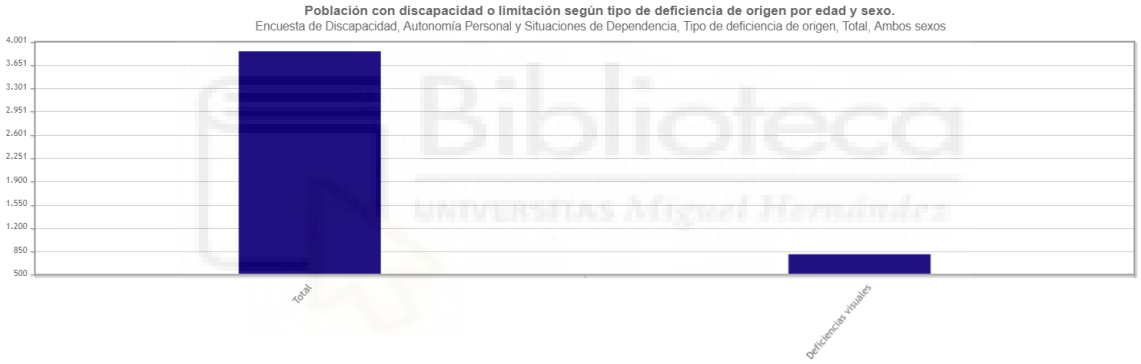


Figura 2 Población con discapacidad o limitación según tipo de deficiencia de origen por edad y sexo. Tipo de deficiencia de origen: Total, Ambos sexos (INE, 2008).

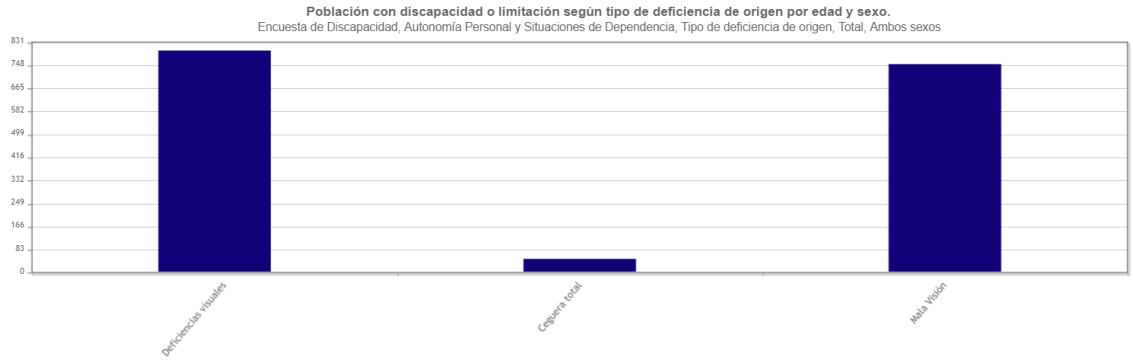


Figura 3 Población con discapacidad o limitación según tipo de deficiencia de origen por edad y sexo. Subdivisión de deficiencia visual (INE, 2008).

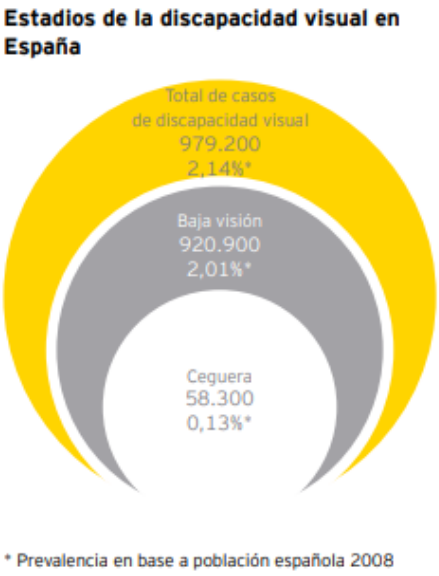


Figura 4 Población con Deficiencias visuales en España (Gómez Ulla de Irazazábal & Ondategui-Parra, 2012).

Cabe destacar que los informes sobre deficiencias visuales suelen basarse en la "agudeza visual de presentación", que incluye en la valoración el uso de lentes correctoras. Debido a esto, se excluyen de las estadísticas aquellos individuos a los que sus deficiencias no les afectan en su día a día. Para mejorar la salud visual, la Organización Mundial de la Salud (OMS) recomienda promover campañas de concienciación sobre la salud ocular, promover la detección de problemas oculares e informar de que existe tratamiento a poblaciones sub-atendidas (World Health Organisation, 2019).

Creemos por tanto que contribuir a detectar posibles problemas de salud visual y concienciar a la población sobre este tipo de problemas sería de utilidad, y además sería una contribución interesante al objetivo y meta de desarrollo sostenible número 3: "Salud y Bienestar" (Naciones Unidas, 2022). Decidimos además apostar por la creación de un entorno de simulación de deficiencias visuales que a su vez utilice como tecnología la Realidad Virtual.

1.2 REALIDAD VIRTUAL: SIMULACIÓN Y JUEGOS SERIOS

La Realidad Virtual (RV) permite mostrar imágenes al usuario sin variación de distancia ni interferencias a través de una pantalla colocada frente a los ojos que es la responsable de mostrar el mundo virtual. En los últimos años se ha popularizado el uso de este tipo de tecnología, y recientemente, la empresa Meta (Meta, 2022b) ha expresado su deseo por integrar y popularizar en su red social el uso de este tipo de tecnología (Meta, 2021).

Merece la pena destacar que, hasta la fecha, el uso de RV ha estado íntimamente ligado a la simulación de entornos intentando simular entornos (reales o ficticios). La tecnología RV intenta crear espacios verosímiles en los que el usuario se sienta inmerso (Lowood, 2013). Por ejemplo, el diseño de prototipos de forma virtual ha extendido su uso durante décadas hasta abarcar simulaciones para el manejo de vehículos, entrenamiento militar o videojuegos (Lowood, 2013). En el ámbito sanitario, con el fin de representar un entorno o entrenamiento médico (ver Figura 5), el uso de la Realidad Virtual como soporte a la comunidad médica lleva décadas siendo estudiada. Se usa tanto en el entrenamiento de cirujanos como en terapias de control del dolor y terapias psicológicas (Li et al., 2017).

Figura 5 Ejemplo de uso de la RV para entrenar a una cirujana en una operación de rodilla. Imagen extraída de (ElTerritorio, 2018)

Un Juego Serio se define como un juego, ya sea digital o analógico, que aprovecha el interés de la gente en el juego para otros fines, como pueden serlo el entrenamiento, la educación o la inclusión de publicidad (Spiegel & Hoinkes, 2009). Algunos ejemplos de Juegos Serios son los simuladores de los que hemos hablado previamente en este apartado, pero también juegos para mejorar la mecanografía; juegos de mesa como el Trivial Pursuit también entrarían en esta categoría, ya que con él se puede aprender datos sobre los varios temas que tratan sus preguntas. El uso de la RV y la capacidad para la creación de situaciones de entrenamiento o práctica se va extendiendo a medida que con el paso del tiempo la tecnología de RV se vuelve más accesible y fácil de utilizar. En este sentido su uso como parte integrada de un Juego Serio la hace más atractiva.

1.3 OBJETIVOS

Con este proyecto, se pretende crear un Juego Serio para concienciar de los efectos que los problemas de visión pueden ocasionar y favorecer su detección mostrando simulaciones de sus efectos, aprovechando las capacidades únicas que ofrece la Realidad Virtual para estandarizar el entorno y las pruebas demostrativas.

El objetivo general de este trabajo es:

- Crear una aplicación de Realidad Virtual que simule deficiencias visuales para ayudar a gente sin deficiencias aparentes a reconocer los problemas a los que se enfrentan día a día las personas qué si los tienen, y también para mostrar una aproximación de sus efectos para que puedan reconocer en el futuro indicios de que su visión se está deteriorando y puedan actuar para evitarlo.

Para conseguir este objetivo general será necesario alcanzar los siguientes objetivos secundarios:

- Definir un marco de trabajo apropiado para desarrollar un prototipo funcional de la aplicación. Analizar APIs y frameworks relacionados que puedan ser utilizados para similar las ametropías e integrarlas en un entorno de RV.

- Diseñar e implementar una solución que permita simular deficiencias visuales relacionadas con las ametropías sobre imágenes genéricas.

- Diseñar e implementar el primer prototipo funcional de la aplicación de RV que pueda ser desplegada en un dispositivo de RV y probada por los usuarios.

Como se explicará en el próximo capítulo, para desplegar este proyecto disponemos de un dispositivo Oculus Rift S (Hayden, 2019), por ello la deficiencia a simular elegida ha sido la ametropía (miopía e hipermetropía). De forma muy resumida, nuestro objetivo será crear un prototipo funcional con distintos escenarios en un entorno de Realidad Virtual donde el usuario podrá interactuar con la interfaz para observar o interactuar con imágenes distorsionadas.

Desde el punto de vista personal, este TFG ha supuesto un reto porque he tenido que hacer varias tecnologías por primera vez, como lenguajes de programación, desarrollar una aplicación en tres dimensiones e investigar temas no relacionados con la programación, como las deficiencias de visión, para después diseñar e implementar una simulación de los efectos que producen.

2. MATERIALES Y MÉTODOS

En este capítulo ahondaremos en las tecnologías disponibles, la elección de aquellas que han sido empleadas, y en la propuesta de solución propuesta para este proyecto. Comos e ha comentado anteriormente, para el despliegue del proyecto disponemos de unas Oculus Rift S, por ello el trabajo de este capítulo ha sido dirigido al análisis de tecnologías que permitan simular los efectos de las ametropías (miopía e hipermetropía) en la visión de una persona, y que sean compatibles con este dispositivo.

2.1 ESTADO DE LA CUESTIÓN

A continuación, hablaremos en profundidad de aplicaciones relacionadas y las tecnologías empleadas en el desarrollo de nuestra aplicación.

2.1.1. REALIDAD VIRTUAL: DEFINCIÓN Y COMPONENTES

La NASA define la Realidad Virtual (RV) como “el uso de tecnología informática para crear un espacio tridimensional interactivo en el que los objetos parecen tener presencia espacial” (NASA Advanced Supercomputing Division, 2013). La Realidad Virtual es una tecnología aún en desarrollo, y a fecha de la escritura de este documento los estándares de esta tecnología están en proceso de desarrollo por el IEEE (Yu Yuan, 2017). La RV es una tecnología con unos límites difusos con otras tecnologías que esos mismos estándares planean concretar, como lo son la realidad aumentada (RA) y la realidad mixta (RM)

Aunque existe una falta de estandandarización alrededor de los distintos componentes hardware y software de RV, el IEEE sigue trabajando en su definición (Yu Yuan, 2017). En la Figura 6 se muestran los componentes las partes de esta tecnología que se han vuelto lo suficientemente comunes para definirlas como un estándar de facto. Los componentes más sofisticados que proveen de buenas herramientas para recrear la experiencia

(Lowood, 2013) son: (1) *Head Mounted Display* (HDM) es un dispositivo de visualización parecido a un casco, que permite reproducir imágenes creadas por ordenador sobre un ″display″, situado muy próximo a los ojos, (2) tracker, un dispositivo que registra el movimiento de la cabeza o (3) mandos, varitas o ratones 3D que sirven para interactuar con el mundo virtual.

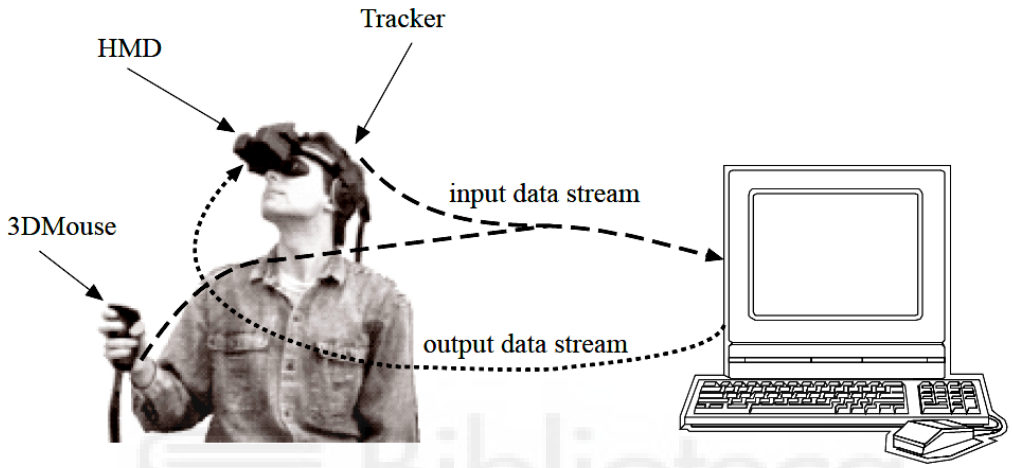


Figura 6 Representación de los componentes de un sistema VR. Imagen extraída de (Saraswat, 2021)

2.1.2. APLICACIONES DE RV EN EL ÁMBITO DE LA SALUD

Los usos de la RV dentro del campo de la medicina es una práctica con una larga historia, con el primer simulador de operaciones apareciendo en 1997, hace ya más de 2 décadas. El uso de la RV se ha popularizado en el campo de la medicina y la investigación industrial, gracias a la representación virtual de la anatomía humana (Li et al., 2017). El uso de simuladores de operaciones y anatómicos permite que cirujanos y médicos practiquen intervenciones y técnicas sin diferencias notables en el efecto producido, además de reducir los costes en material. Además de esto, también se ha comprobado la efectividad del uso de la Realidad Virtual como técnicas para asistir al manejo del dolor y la ansiedad como se muestra en la Figura 7.

Figura 7 Ejemplo del uso para reducir la ansiedad de pacientes mientras se le realizan procedimientos médicos. Imagen extraída de (Roberto, 2018).

Figura 8 Capturas de una simulación VR. Fila superior: degeneración macular, catarata. Fila inferior: miopía, glaucoma (Häkkilä et al., 2018).

En el ámbito de la visión, hay estudios que indican que un HMD es tan, o más fiable que una pantalla catódica para la medición de capacidades de procesamiento visual, y además eliminan condiciones externas como la iluminación y el resto del campo de visión, con lo

que la Realidad Virtual podría convertirse en un nuevo standard para este campo (Foerster et al., 2016). Por ejemplo, hay simulaciones que buscan recrear la experiencia de sufrir defectos visuales como el de la Figura 8 extraída de (Häkkilä et al., 2018); en esta figura se muestran capturas de pantalla de una aplicación de RV en la que se simulan diferentes tipos de deficiencias visuales cuyo objetivo es introducir las tecnologías de Realidad Virtual en estudiantes para concienciarlos de su importancia en sus futuros trabajos.

Más concretamente, en el campo de las deficiencias visuales, hay estudios y TFG basados en el tratamiento de dichas deficiencias que dan resultados favorables, demostrando el potencial que la RV tiene en su tratamiento (Navia Ávila, 2015; Rodriguez, 2018; Triviño Merida, 2014; Žiak et al., 2017). Otras aplicaciones buscan concienciar sobre la importancia y prevención de las deficiencias oculares, coincidiendo con uno de los objetivos de este trabajo.

2.1.2.1.SIMULACIÓN DE DEFICIENCIAS VISUALES

Se analizaron los artículos mencionados en el párrafo anterior para identificar los distintos tipos de deficiencias visuales recreadas. Para este análisis ha sido de utilidad consultar la clasificación propuesta por (Jones & Ometto, 2018) según su efecto en la visión:

- **Defectos visuales basados en la posición ocular:** aparecen siempre en la misma región del campo de visión y siguen el movimiento del ojo. Dentro de este grupo se pude diferenciar entre las deficiencias espaciales y el rellenado perceptivo. Las deficiencias espaciales son aquellas deficiencias o defectos causados por problemas de visión central y problemas en la retina que afectan a los fotorreceptores. Los problemas de rellenado perceptivo son causados por la falta de datos en parte del campo visual, como por ejemplo el punto ciego del ojo

donde el nervio óptico se conecta al ojo. El sistema visual rellena los espacios vacíos mediante interpolación.

- **Defectos visuales no basados en la posición ocular:** estos efectos no varían con la vista de la persona afectada. Entre los distintos ejemplos encontramos (1) los destellos que generan imágenes con un brillo excesivo en la imagen, (2) el emborronamiento que hace que las imágenes se difuminen, ya sea en distancias cortas, largas o que la difuminación sea generalizada, o (3) las deficiencias relacionadas con el color que hacen que ciertos colores no sean percibidos o su contribución al color final de la imagen sea inferior a lo normal.

Los defectos visuales basados en la posición ocular requieren para su simulación de tecnología *eye-tracking*, que no se considera uno de los componentes estándar de facto mencionados en la Figura 6 y se podría considerar una limitación hardware de los dispositivos más extendidos. El *eye-tracking* es una tecnología que permite saber la dirección a la que está mirando un usuario exactamente (Tobii Dynavox AB, 2022). Además, dado que el *eye-tracking* es una tecnología de la que carece el dispositivo en el que queremos desplegar el proyecto, las Oculus Rift S, decidimos seleccionar como deficiencia a simular en este trabajo una categorizada como problemas no basados en la posición ocular. Entre ellas, hemos seleccionado los emborronamientos ya que son los que nos resultan más familiares. De los emborronamientos, he decidido simular las ametropías por disponer de varias variables con las que trabajar a la hora de escribir un programa que simule dicha condición.

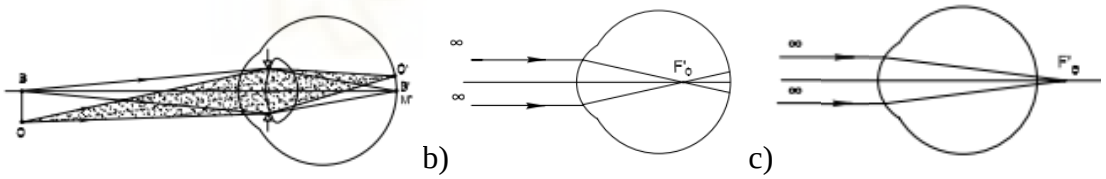
2.1.3. SIMULACIÓN DE AMETROPÍAS (MIOPÍA E HIPERMETROPÍA)

Antes de continuar, debemos explicar cómo funcionan las ametropías seleccionadas para poder explicar los métodos usados para simularlas. Es decir, debemos conocer la

deficiencia para saber qué tipo de procesamiento debemos realizar sobre una imagen original al ser tratada para emular las ametropías.

2.1.3.1.¿EN QUÉ CONSISTE UNA AMETROPÍA?

El cristalino es una lente biconvexa que desvía rayos de luz paralelos que pasen a través de él, haciendo que sus trayectorias converjan hacia un mismo punto. La retina es una capa de la pared interior del ojo donde se encuentran receptores fotosensibles, encargados de reaccionar a la luz que incide sobre ellos. Debido a la desviación de los rayos, se acaba formando una imagen invertida tanto horizontal como verticalmente que el sistema nervioso recibirá, y volverá a invertir al procesar la información de la retina. En la Figura 9 a) vemos como varios haces de luz emitidos desde un mismo punto del objeto BO se desvían al llegar a la retina (eje vertical) para acabar chocando con la retina del ojo, que a partir de esta información genera una imagen invertida del objeto observado (Puell, 2006).



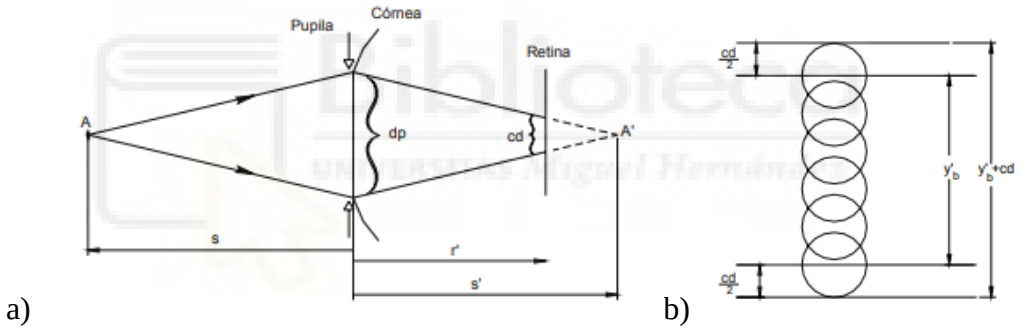
a) b) c)

Figura 9 a) Construcción de una imagen en el ojo emétrope. b) Posición del foco imagen en el ojo miope. c) Posición del foco imagen en el ojo hipermétrope.

El origen de las ametropías puede estar relacionado con aspectos como: (1) el exceso o falta de potencia de refracción del cristalino, o (2) una distancia demasiado larga o corta del propio ojo. Sin embargo, los efectos suelen ser los mismos en la creación de las imágenes en la retina. Mientras que en la Miopía (Figura 9 b) el punto donde el ojo espera construir la imagen de forma nítida se encuentra antes de que los rayos hagan contacto con la retina, en la Hipermetropía (Figura 9 c) el punto donde se espera construir la

imagen de forma nítida se encuentra más allá de la retina. En ambos casos al llegar los rayos a la retina, estos no coinciden y acaban mezclándose con los procedentes de otros haces de luz.

Los efectos anteriores generan círculos de difusión (ver Figura 10) en los que los haces que deberían coincidir se separan al contacto con la retina. Debido a esto, la imagen creada en la retina se compone de círculos de difusión superpuestos, que se combinan entre sí formando una imagen borrosa. Estos círculos de difusión se ven afectados por el tamaño de la pupila y por la distancia que separa la retina del foco imagen, donde la imagen se forma nítidamente. El error de enfoque se mide en dioptrías, y a mayor número de dioptrías mayor será el diámetro de estos círculos.



a) b)

Figura 10 a) Generación del círculo de difusión en un ojo miope. b) Superposición longitudinal de círculos de difusión en la retina.

2.1.3.2.HERRAMIENTAS PARA SIMULAR AMETROPÍAS

Aunque nuestra idea original fue integrar el procesado de las imágenes en la aplicación de RV, tras intentar sin éxito usar elementos nativos de RV para recrear los efectos de las ametropías en la visión (se explicará con más detalle en la sección de Unity) busqué otras alternativas como librerías basadas en C, C++ o C# para integrarlas fácilmente.

Entre las opciones encontradas, se descartaron alternativas como DevIL (DevIL, 2017) y AForge.NET (AForge.NET, 2022) por no disponer de una página web segura. Otras, como HALCON y AVL se descartaron por ser opciones de pago o no ofrecer todas sus opciones de forma gratuita (MVTec Software Gmbh, 2022; Zebra, 2022). Las últimas opciones analizadas fueron OpenCV (OpenCV, 2022g) y CImg Library (Tschumperlé, 2004) ambas de código abierto. Debido al gran número de módulos disponibles OpenCV y en comparación con CImg, se eligió OpenCV para ser usada en el proyecto.

Dentro de OpenCV se exploraron aquellos módulos centrados en la edición de imágenes o en la simulación de cámaras reales. Tras ojear el propósito de cada módulo y su descripción se redujo la lista a 3 opciones:

- **Modelos de visión basados en la biología y herramientas derivativas** (OpenCV, 2022b): Intenta simular el sistema de visión humano, pero se centra en simular la retina, los efectos sobre la luminosidad y los colores que percibe, por lo que no resulta útil para simular ametropías.
- **Fotografía computacional y Algoritmos de procesamiento adicionales** (OpenCV, 2022c, 2022a)**:** Incluyen módulos de rellenado de imágenes, reducción de ruido en las imágenes, gestión de los canales de las imágenes y renderizado no realista. Aunque hay campos parecidos, como la reducción de ruido y el rellenado, estos se centran en mejorar el aspecto de una imagen corrigiendo contrastes extremos entre píxeles, mientras lo que buscamos es agravar los problemas que presenta una imagen para simular defectos de visión, por lo que descartamos estas librerías.

- **Procesado de imágenes y Procesado de imágenes extendido** (OpenCV, 2022f, 2022d)**:** Incluyen funciones de procesado y transformación básica de imágenes, permitiendo girarlas, escalarlas, invertirlas, editar sus colores, analizarlas, etc.
- **Filtrado de imágenes** (OpenCV, 2022e): este es el módulo que más se acerca a nuestras necesidades debido a sus funciones de emborronado, que reducen los detalles de la imagen.

Entre los métodos seleccionadas del módulo Filtrado de imágenes, disponemos de '`blur()`', '`boxFilter()`' y '`GaussiaBlur()`'. Tras leer las funciones de emborronado que aplica cada método, descartamos '`GaussiaBlur()`' ya que el filtro Gaussiano que favorece los píxeles más cercanos al píxel tratado y queremos que todos los pixeles en la zona de efecto contribuyan lo mismo. Los dos restantes, '`blur()`'y '`boxFilter()`', son muy parecidos, en ambos los pixeles cercanos tienen el mismo efecto independientemente de la distancia, su única diferencia es que '`blur()`' siempre normaliza los resultados mientras que '`boxFilter()`' depende de un parámetro. Como vamos a normalizar los píxeles resultantes para que la imagen manipulada sea lo más familiar al ojo humano posible y no haya grandes discrepancias entre las imágenes originales y las que han pasado por el proceso varias veces (ScienceDirect, 2022), elegimos usar '`blur()`' por simplicidad.

Por este motivo, finalmente se decidió crear un programa que utilizando métodos de OpenCV permitiese simular las ametropías seleccionadas sobre una imagen original que se va a utilizar como parámetro. La salida de esta librería consistirá en una carpeta que contienen un lote de imágenes modificadas sobre las que se han aplicado distintas transformaciones simulando ametropías variando las dioptrías. Estas imágenes manipuladas se utilizarán posteriormente como recursos de la aplicación de RV.

2.1.4. TECNOLOGÍAS PARA EL DEARROLLO EN VR

Existen dos opciones para el desarrollo de aplicaciones VR nativas para Oculus Rift S que son descritas a continuación (Oculus (Meta), 2022a).

- **Desarrollo nativo con Mobile SDK y PC SDK** (Oculus (Meta), 2022c)**:** permite desarrollar aplicaciones nativas. Incluye el Mobile SDK, que permite crear aplicaciones con integración a dispositivos móviles, y el PC SDK que crea experiencias para Oculus Rift en lenguaje C++.
- ***Progressive Web Apps*** (Oculus (Meta), 2022b)**:** permite el desarrollo de aplicaciones web mostradas mediante un navegador propio, Oculus Browser, basado en Chromium y optimizado para el uso de WebXR y WebGL en productos Oculus. Al tratarse de una web, es posible utilizar tecnologías como JavaScript y APIs propias en el desarrollo de la aplicación.

Debido a que preferimos crear una aplicación portable a distintos dispositivos, descartamos el desarrollo en PWA ya que se centra en un navegador propio y optimizado con únicamente productos Oculus en mente. Aunque el desarrollo nativo con Mobile y PC SDK sería una opción válida, exploramos las posibilidades de un desarrollo multiplataforma para lo que las dos opciones más populares son: **Unity** (Oculus (Meta), 2022d) y **Unreal Engine** (Oculus (Meta), 2022e)**.** Decidimos continuar explorando estas dos opciones de desarrollo multiplataforma para llegar a un mayor número de personas.

2.1.4.1.COMPARATIVA UNITY / UNREAL ENGINE

Tanto Unity como Unreal Engine son plataformas de desarrollo potentes con capacidad de desarrollo en VR, en este apartado compararemos ambas plataformas (Unity Technologies, 2022c) para decidir cuál será la usada en el desarrollo de este proyecto.

Aunque ambas opciones incluyen la tecnología para la creación de aplicaciones y videojuegos, algunos aspectos que las diferencian son:

- **Lenguaje de programación:** mientras que Unity utiliza en sus scripts el lenguaje C#, Unreal Engine usa C++ o Blueprints, un lenguaje visual de scripting.
- **Popularidad**: Unity es la plataforma más usada para el desarrollo de videojuegos en dispositivos móviles, PC y consolas. En 2020 alcanzó las cifras de más del 50% de los juegos desarrollados llegando más de 5 mil millones de descargas.
- **Recursos disponibles en la tienda de cada plataforma:** la cantidad de recursos en la tienda de Unity es cinco veces mayor a la de Unreal Engine.
- **Comunidad de desarrolladores y documentación**: Unreal Engine cuenta con 12000 hilos sobre scripts (4600 sobre desarrollo RV), mientras que Unity dispone de 128000 (6100 sobre desarrollo RV). Es más probable encontrar información en los foros de Unity, ya que Unreal Engine ofrece dos lenguajes.

Por último, al ser dos soluciones que permiten desarrollo multiplataforma es de vital importancia los dispositivos para los que permite desplegar las aplicaciones desarrolladas con ellas. En este sentido, Unity dispone de 20 (7 relacionadas con la RV) frente a las 15 (5 relacionadas con la RV) ofrecidas por Real Engine (ver Figura 11 y Figura 12).

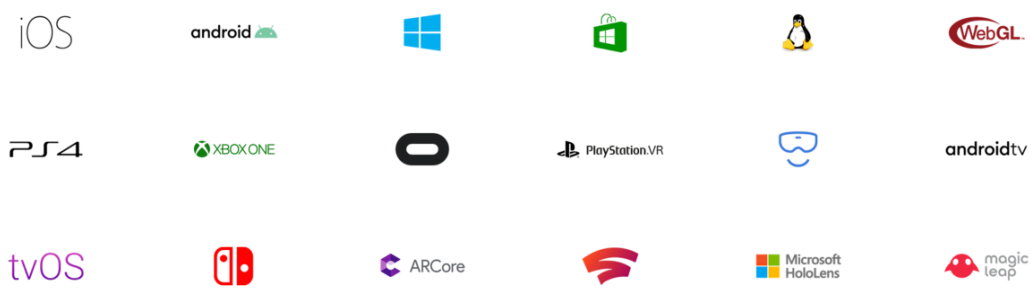


Figura 11 Plataformas soportadas por Unity. Imagen extraída de (Unity Technologies, 2022b).

Figura 12 Plataformas soportadas por Unreal Engine. Imagen extraída de (Epic Games Inc., 2022).

Según estos datos, he decidido elegir Unity para el desarrollo del proyecto, debido a la gran cantidad de hilos relacionados en sus foros y al mayor alcance que podrá tener la aplicación debido al número de plataformas disponibles.

2.1.4.2. LÍMITES DE UNITY PARA SIMULAR DEFICIENCIAS

En esta sección desarrollaremos con más detalles las limitaciones que nos han hecho decantarnos por el uso de Open CV en la simulación de deficiencias en lugar de usar compontes nativos de Unity.

- La primera opción que se exploró fue usar el objeto cámara de Unity descrito en la documentación (Unity Technologies, 2021) para recrear los efectos de las ametropías en la visión. Se encontraron otros componentes en los paquetes de la comunidad de Unity, como Physical-Camera (Unity Technologies, 2022a), pero los efectos aplicables no fueron los apropiados para la simulación de ametropías.

- La segunda opción analizada fue el uso de librerías de edición de imágenes para realizar la simulación que pudiesen integrarse dentro de los scripts C# de la aplicación VR en Unity. La integración no fue posible debido a que Unity necesita preprocesar las imágenes antes de poder usarlas como elementos en el juego (Gunjan04 & CHPedersen, 2014), por lo que crear imágenes derivadas de una imagen original durante la ejecución del programa no fue posible.

2.2 OTRAS HERRAMIENTAS UTILIZADAS

Además, se han utilizado otras herramientas como:

- **Google Scholar y otros buscadores bibliográficos:** Búsqueda de información durante la investigación.
- **VSCode:** IDE para el desarrollo en C# e integración en C++ con OpenCV.
- **Gimp:** Herramienta de edición de imágenes, usada para ajustar las imágenes originales a un formato adecuado para su manipulación, eliminando las versiones en distintos tamaños y girando su orientación.
- **Git, GithubDesktop, BitBucket y GitKraken:** Control de versiones y traspaso del proyecto entre el entorno de desarrollo y producción para realizar pruebas con las Oculus Rift S.
- **Herramienta de recortes de Windows:** Realización de capturas de pantalla.
- **Microsoft Word:** Elaboración de esta memoria.
- **Microsoft Excel:** Desarrollo del cronograma.
- **Microsoft Visio y Dibujos de Google:** Dibujo de diagramas.
- **Mendeley:** Gestión de referencias bibliográficas de la memoria.
- **Aplicación de Oculus:** Aplicación necesaria para operar las Oculus Rift S.

2.3 DISPOSITIVOS Y PROPUESTA DE SOLUCIÓN

Para el desarrollo de esta aplicación ha sido necesario el uso de varios dispositivos de hardware con los que trabajar:

- **Ordenador personal:** Ordenador portátil MSI GL62M 7REX, en el que se han desarrollado la aplicación de preprocesado de imágenes y la aplicación VR.

- **Oculus Rift S:** Dispositivo cedido por la Escuela Politécnica Superior de Elche (EPSE) de la Universidad Miguel Hernández de Elche (UMH) donde se han realizado las pruebas reales y la calibración de la aplicación VR.

- **Ordenador sobremesa:** Debido a que mi ordenador personal no soportaba el uso de las gafas Oculus Rift S, hizo falta un ordenador sobremesa con una tarjeta gráfica capaz de cumplir con los requisitos mínimos de este dispositivo (Meta, 2022a). Este ordenador fue proporcionado por el Instituto Centro de Investigación Operativa (CIO) de la UMH.

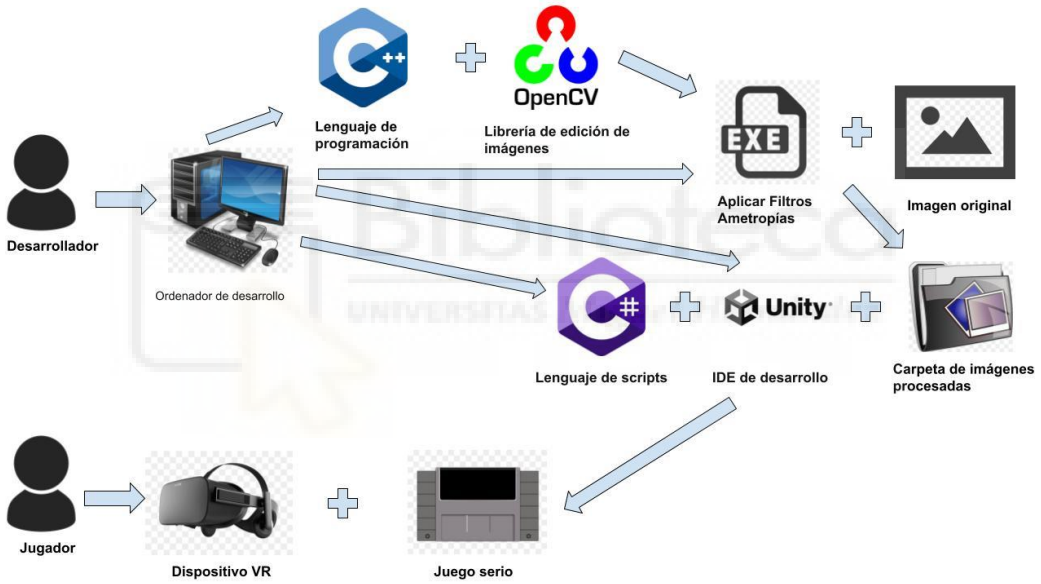


Figura 13 Diagrama con la propuesta hardware y software de nuestra solución.

Tras exponer las tecnologías a usar y los dispositivos disponibles, vamos a mostrar en la Figura 13 un esquema de la relación entre los dispositivos y tecnologías usadas. Aunque se explicará con más detalle en el próximo capítulo se van a generar 2 programas distintos: (1) un ejecutable de preprocesado de imágenes que simula ametropías, basado en C++ y OpenCV. Y (2) un Juego Serio desarrollado en Unity y C# que usará las imágenes preprocesadas con el fin de divulgar sobre la salud visual.

3. RESULTADOS Y DISCUSIÓN

En este capítulo describiremos la metodología y los aspectos técnicos más relevantes para desarrollar la aplicación de RV. En cuanto a los aspectos técnicos, los vamos a describir agrupados en los siguientes grupos: (1) desarrollo de la librería con OpenCV y (2) desarrollo del Juego Serio con Unity. Finalmente, se describirá la aplicación desarrollada incluyendo capturas de la puesta en marcha en un entorno con usuarios controlados.

3.1 METODOLOGÍA

El trabajo se ha desarrollado de manera individual, por lo que no ha sido posible aplicar metodologías ágiles centradas en grupos, como lo son Scrum (Schwaber, 1997) o *pair-programming* (Williams et al., 2000). Dada la naturaleza con requisitos sin definir y cambiantes de este proyecto de investigación, durante todo el proceso he seguido aquellos principios del Manifiesto Ágil que se podían aplicar (Beck et al., 2000).

Se han realizado iteraciones semanales, lo que ha permitido desarrollar el software de forma rápida y continua en reuniones semanales del equipo (yo) con el cliente (el director de este TFG y yo) para mostrar los resultados del sprint. En las reuniones semanales se han discutido y ajustado los requerimientos del proyecto, que no eran fijos y fueron adaptándose a los requerimientos semanales de la próxima iteración del proyecto. Siguiendo esta metodología, se consiguieron las siguientes metas cuya distribución temporal se puede consultar en el cronograma de la Figura 14:

- **<u>Bloque 1:</u> Análisis del estado del arte:**
    - **1.1.** Investigación y elección de defectos visuales a simular.
    - **1.2.** Investigación y elección de tecnologías de desarrollo VR.

- **1.3.** Investigación y elección de librerías y tecnologías para simular deficiencias en VR e imágenes.

- **<u>Bloque 2:</u> Diseño de la simulación de ametropía:**

  2.1. Instalación y creación de un primer proyecto en C# con OpenCV.

  2.2. Diseño del método para generar la distorsión en pseudocódigo.

  2.3. Diseño del sistema de nomenclatura de las imágenes generadas.

  2.4. Implementación del diseño en un ejecutable en C++ con la librería OpenCV0

  2.5. Cambios a la aplicación para generar varias imágenes con cada uso siguiendo el sistema de nomenclatura.

- **<u>Bloque 3:</u> Preprocesado de imágenes:**
  - **3.1.** Investigación y elección de las imágenes a utilizar.
  - **3.2.** Edición de la imagen vectorial original para producir una imagen objetivo más adecuada.
  - **3.3.** Procesado de varias resoluciones de la imagen seleccionada y elección de la versión más adecuada.
  - **3.4.** Integración de las imágenes en el proyecto en Unity.

- **<u>Bloque 4:</u> Diseño de la aplicación:**
  - **4.1.** Creación de un primer proyecto en Unity con controladores de VR.
  - **4.2.** Desarrollo de una escena básica con acceso a imágenes por explorador de archivos
  - **4.3.** Descarte de integración de OpenCV en Unity en favor del preprocesado de imágenes estáticas.

- **4.4.** Desarrollo de las pantallas Ver, Distancia y Adivinar.
- **4.5.** Desarrollo de la pantalla Adivinar Distancia.
- **4.6.** Cambio de propósito de la pantalla Adivinar Distancia, conversión a Prueba Miopía.
- **4.7.** Desarrollo del método para comprobar la existencia de ametropías.
- **4.8.** Diseño del menú lateral para cambiar entre escenas.
- **4.9.** Desarrollo de un decorado a escala simulando una habitación con proporciones realistas.
- **4.10.** Inclusión de modelos de manos para representar los controladores en la aplicación.
- **4.11.** Desestimación de pruebas que combinaran pruebas con cambio de distancia del display y de aplicación de algoritmo por redundancia.
- **4.12.** Reciclaje de estas pruebas en las pantallas finales: Ver, Adivinar, Acercar, Distancia y Test.
- **4.13.** Inclusión de elementos de feedback en el menú lateral para indicar la escena actual.

- **<u>Bloque 5:</u> Despliegue en el dispositivo Oculus Rift S y pruebas con usuario:**
  - **5.1.** Adquisición de un ordenador con tarjeta gráfica suficiente para operar las Oculus Rift S.
  - **5.2.** Instalación y configuración de las Oculus Rift S.
  - **5.3.** Resolver problemas de compatibilidad con el editor de Unity para el testeo de la aplicación.
  - **5.4.** Inicio del uso de las Oculus Rift S como dispositivo de testeo y calibrado de la aplicación.

- **5.5.** Generación de un ejecutable que puede ser distribuido a otros dispositivos VR.
- **5.6.** Definición de objetivos futuros: Refinando la aplicación haciendo pruebas con usuarios.

- **<u>Bloque 6:</u> Documentación y memoria del proyecto**
  - **6.1.** Generar las pantallas de ayuda de los distintos escenarios.
  - **6.2.** Documentar gráficamente la aplicación y grabación de la demo.
  - **6.3.** Escribir la memoria del TFG.

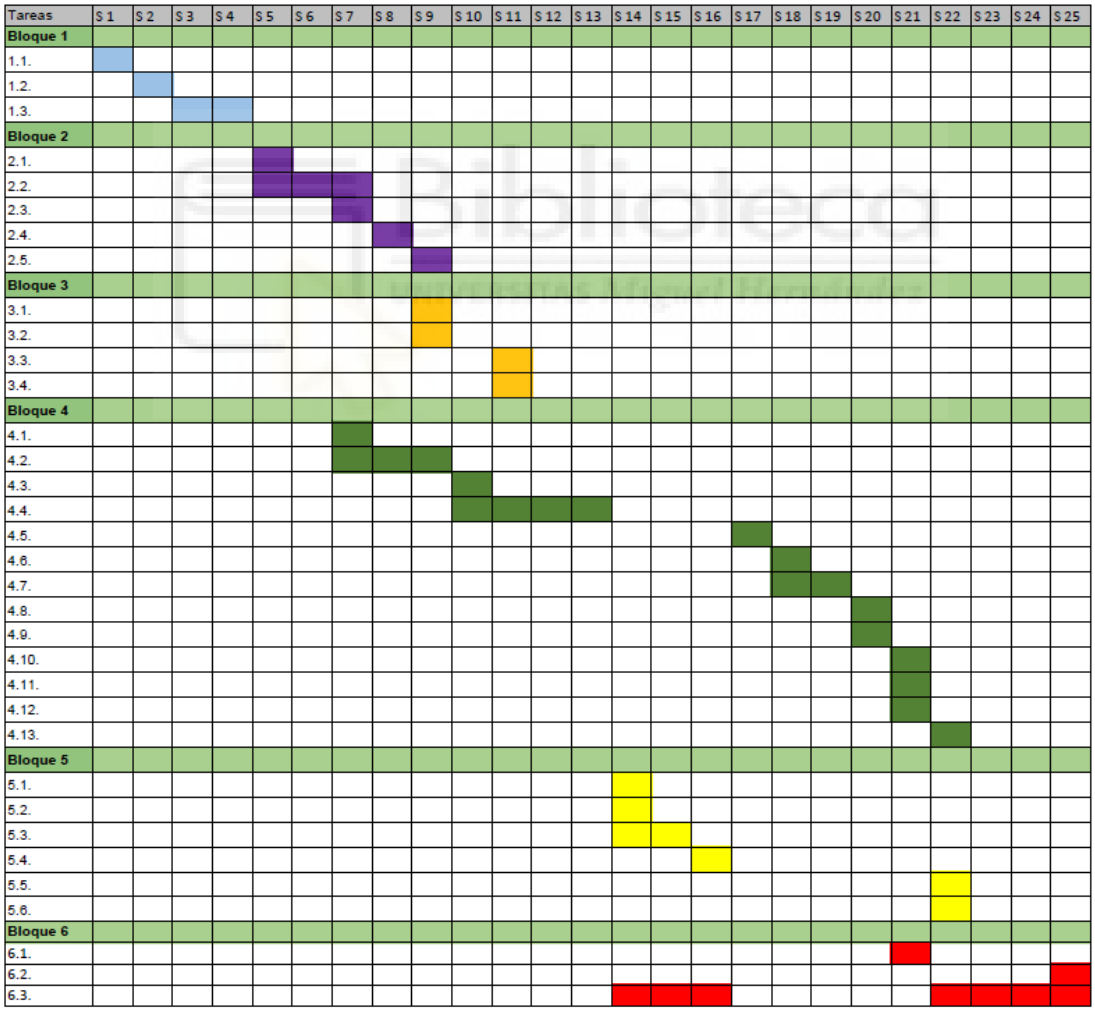


Figura 14 Cronograma de las distintas fases desarrolladas.

3.2 API C++ PARA LA SIMULACIÓN DE AMETROPÍAS CON OPENCV

El programa de procesado es un ejecutable simple que, a partir de una imagen, la pre-procesa y genera imágenes distorsionadas simulando ametropías que serán las que usaremos posteriormente en el juego en Unity.

3.2.1. CASOS DE USO

El actor de este programa se denomina Desarrollador, y su función es crear las imágenes distorsionadas que serán usadas al desarrollar la aplicación principal (Figura 15). En el Anexo I están descritos sus casos de uso.

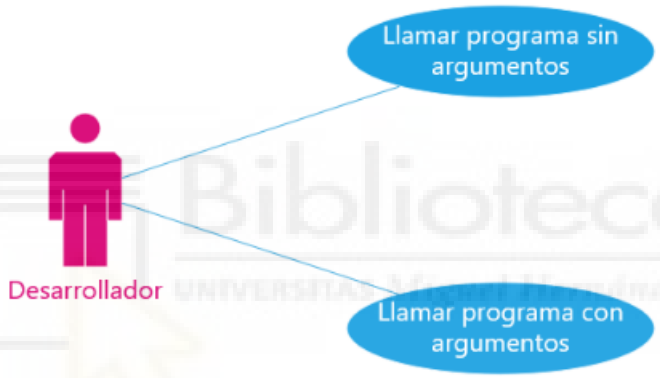


Figura 15 Diagrama de casos de uso de Desarrollador

3.2.2. IMPLEMENTACIÓN DE AMETROPÍAS CON LA FUNCIÓN BLUR

Para poder simular el efecto de las ametropías, se han asumido varias simplificaciones:

- Se supone que la imagen está en 2 dimensiones, y no se considera la variación de profundidad entre puntos, toda la imagen se considera a la misma distancia.
- Se asume que las imágenes solo se ven afectadas por la propia imagen, por lo que los bordes no se ven tan afectados como los puntos centrales.

Decidimos tener en cuenta tres variables para la simulación de la ametropía: (1) las dioptrías a representar, (2) la ruta a la propia imagen y (3) una última variable para representar el tipo de ametropía.

La función de '`blur()`' es suavizar los píxeles de una imagen haciendo una media con los valores de los píxeles laterales. Utilizaremos esa función para simular la interferencia de otros círculos de difusión en la imagen. En la Figura 16 se muestra un extracto del código que realiza esta labor.

```
111     //Hacemos el proceso para las 10 imagenes, desde 0 hasta 100 metros
112     //Miopía, mas kernel cuantos mas metros
113     if (ametropia == 1) {
114         for (i = 0; i < 11; i++) {
115             imagenametrope = image.clone();
116             kernel = 2 * i + 1;
117             for (j = 0; j < dioptrias; j++) {
118                 blur(imagenametrope, imagenametrope, Size(kernel, kernel), Point(-1, -1), BORDER_DEFAULT);
119             }
120             if (i != 0)
121                 guardarImagen = "imagenes\\" + to_string(dioptrias) + "M" + to_string(i) + "0" +  nombrearchivo;
122             else {
123                 guardarImagen = "imagenes\\";
124                 guardarImagen += to_string(dioptrias);
125                 guardarImagen += "M";
126                 guardarImagen += "0";
127                 guardarImagen += nombrearchivo;
128             }
129             //cout << guardarImagen << std::endl;
130             imwrite(guardarImagen, imagenametrope);
131         }
132     }
```

Figura 16 Código de la aplicación de preprocesado. Este bloque se encarga de generar emborronados de Miopía y guardarlos en ficheros con los nombres adecuados.

El primer bloque `for` se encarga de realizar todo lo que incluye 11 veces, una para cada distancia simulada. El bucle `for` interior, en el siguiente nivel de indentación, aplica la función de emborronado varias veces según el número de dioptrías, antes de dejar que se guarde el resultado en el fichero con el nombre correspondiente.

El tamaño de los círculos de difusión usados en la distorsión se determina en función de la distancia a la que se encontrará la imagen, ya que `blur()` necesita un parámetro

‘`ksize`’ que indique la cantidad de píxeles alrededor del píxel tratado que serán considerados, y esto nos permite emular el mayor efecto de una ametropía dependiendo de la distancia, y además definir una distancia límite donde la imagen se verá correctamente, siendo esta 0 metros para la miopía y 100 metros para la hipermetropía.

En cuanto a la ametropía elegida, esta decide si los píxeles tratados aumentan con la distancia, o si disminuye. El número de dioptrías introducido indica el número de veces consecutivas que se aplicará el efecto de ‘`blur()`’, para aumentar la gravedad de la condición conforme aumentan las dioptrías.

3.2.3. EJEMPLO DE USO DE LA LIBRERÍA

En la Figura 17 se muestra un ejemplo de uso de la función de pre-procesado. Si se invoca el ejecutable sin los parámetros necesarios, este mostrará como ejecutarlo correctamente. Si se ejecuta con parámetros válidos y encuentra la imagen, generará las imágenes procesadas en una carpeta imágenes situada en el mismo directorio que el ejecutable.

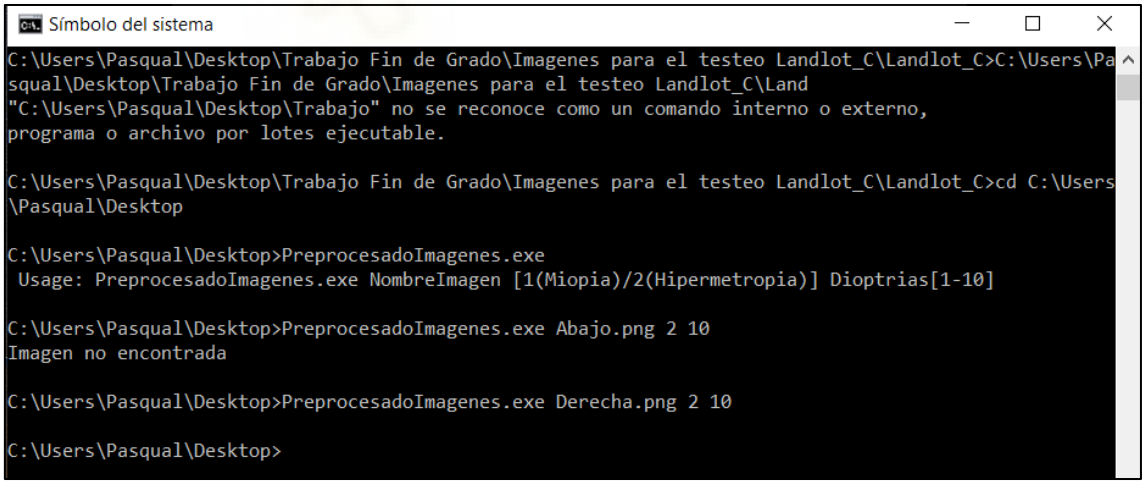


Figura 17 Ejemplo de uso en CMD de la aplicación de preprocesado de imágenes.

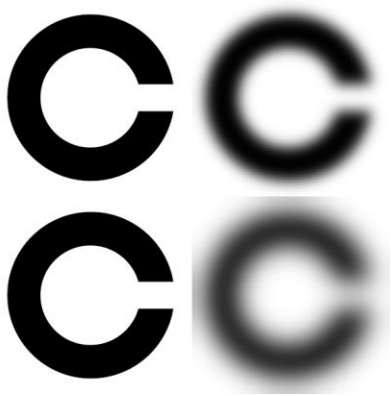


Figura 18 Resultado del pre-procesado con hipermetropía y dioptrías máximas de varias imágenes. Arriba izquierda: Imagen original; Arriba derecha: Procesado a 50 metros; Abajo izquierda: pre-procesado a 100 metros; Abajo derecha: pre-procesado a 0 metros.

El resultado de la transformación puede observarse en Figura 18, donde se han aplicado los parámetros máximos a una imagen de prueba. Pese a ser parecida a las imágenes usadas en el Juego Serio final, estas imágenes no son las que aparecen en el juego ya que su alta resolución hace que la distorsión de la aplicación no sea suficiente.

Merece la pena destacar que, las imágenes transformadas deberán incluirse como recursos en el Juego Serio de RV, por lo que deberemos generar tantas imágenes distorsionadas como opciones de configuración le ofrezcamos a los usuarios. Por tanto, deberemos guardar alrededor de 221 imágenes modificadas por cada imagen base que elijamos.

3.2.4. SELECCIÓN DE IMÁGENES A USAR EN EL JUEGO SERIO

La selección de las imágenes finales a implementar se hizo siguiendo varias directrices. Como se trata de una aplicación centrada en la divulgación, se han aplicado principios heurísticos de usabilidad en la elección de imágenes. Para ello, se han elegido imágenes originales con mucho contraste y sin colores.

Como se trata de realizar pruebas de visión, se han investigado distintos test visuales a aplicar, entre ellos:

- Los test de Snell (Figura 19 izquierda).
- Los test de Lea (Figura 19 derecha).
- Las Cs de Landolt (Figura 20 izquierda)
- Las Es de Snell (Figura 20 derecha).

Dado el alto número de imágenes que debemos generar para ser incluidas en la aplicación de RV, hemos descartado los test de Snell clásicos ya que generar imágenes para un alfabeto entero sería demasiado.

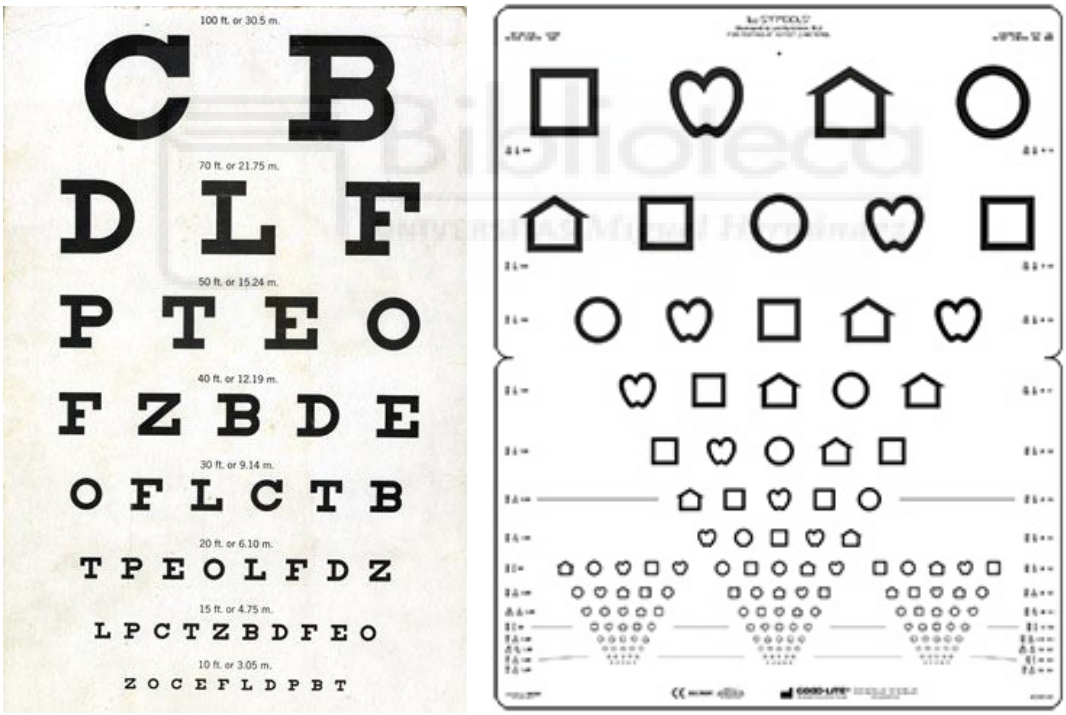


Figura 19 A la izquierda el Test de Snell clásico, extraído de (*All About the Eye Chart - American Academy of Ophthalmology*, 2022). A la derecha el Test de Lea, extraído de (*LEA SYMBOLS® Distance Visual Acuity Chart – Lea-Test Ltd.*, 2022).

Figura 20 Izquierda: Landolt C; Derecha: E de Snell, extraídas de (Danilova & Bondarko, 2007).

Los test de Lea son buenos para hacer la prueba a gente analfabeta o niños que aún no saben leer al estar representada por dibujos, pero sigue necesitando una gran cantidad de símbolos. Además, este efecto también se puede conseguir con las Es de Snell y las Cs de Landolt, ya que incluso personas analfabetas y niños pueden reconocer una apertura y la dirección a la que apunta, y ambas opciones sólo necesitan cuatro imágenes base.

Entre las Es y las Cs, elegimos las Cs debido a que como el ejecutable usa matrices cuadradas como zonas para generar la distorsión, es posible que muchos de los píxeles de la imagen acaben siendo alterados de la misma forma que sus vecinos cuando las áreas afectadas coincidan con el trazado de las Es. Como con las Cs son redondeadas, este efecto no se producirá, por lo que son las imágenes elegidas para ser procesadas.

3.3 ANÁLISIS Y DISEÑO DEL SOFTWARE

3.3.1. ESCENARIOS DEL JUEGO

Las pantallas diseñadas fueron variando con el tiempo, pero los diseños finales son cinco, y se dividen en dos grupos, las diseñadas para ser usadas con lentes y las que no:

- **Pantallas de concienciación (con lentes):** el objetivo del primer grupo es que sea usadas para concienciar a un usuario que ve bien (ya sea de forma natural o

corregida con gafas o lentillas) de como otros usuarios con deficiencias percibirían la realidad. Por ejemplo, estas pantallas permiten mostrar a gente con buena visión y a quien ya corrige su visión cómo es ver imágenes con distintas severidades de ametropías, para que puedan poner en perspectiva su visión si en el futuro sus dioptrías cambian.

- **Pantallas de testeo de agudeza visual (sin lentes):** el objetivo del segundo grupo es probar la agudeza visual de un usuario a través de una serie de pruebas o escenarios que simulan un entorno real en el que se manipularán las distancias y propiedades de los objetos que compone Cs son redondeadas del test elegido en el punto anterior. Estas escenas sirven, por ejemplo, para testear las capacidades visuales del jugador y comparar con otras personas al pasarles las Oculus.

Desde el punto de vista técnico, las pantallas de concienciación aplican las distorsiones del programa de preprocesado a una distancia fija:

- **Pantalla 1, Escenario Ver:** Muestra los efectos de una deficiencia visual sobre la misma imagen a distintas distancias. El usuario puede elegir que imagen y dioptrías serán las aplicadas a la imagen.
- **Pantalla 2, Escenario Adivinar:** Muestra una imagen aleatoria y muestra los parámetros de su distorsión, y el usuario deberá reconocer cuál de las cuatro imágenes posibles. Se acumulan los aciertos y fallos de la sesión en un marcador para su consulta.
- **Pantalla 3, Escenario Acercar:** La única escena de este grupo que cambia la distancia del panel, que muestra una imagen aleatoria distorsionada según los parámetros elegidos por los usuarios y la distancia del panel. El usuario puede intentar reconocer cuál de las cuatro imágenes es la mostrada en pantalla.

Las pantallas de testeo de agudeza visual no aplican distorsiones, y se basan en el uso de la distancia del display para que se apliquen de forma natural las distorsiones a las imágenes según los defectos visuales del usuario. En estas escenas encontramos:

- **Pantalla 4, Escenario Distancia:** Muestra una imagen sin distorsión según el valor del único parámetro. El usuario podrá mover el display de la imagen para ver cuando reconoce una imagen a medida que se vuelve más pequeña o grande, dependiendo de la ametropía que sufra el usuario.
- **Pantalla 5, Escenario Test:** Muestra imágenes sin distorsión a distancia variable para que el jugador las reconozca un número específico de veces, tras lo cual se emitirá una recomendación basada en sus respuestas.

3.3.2. CASOS DE USO

En la Figura 21 se muestra el diagrama de casos de usos del juego (desarrollados en el Anexo I). El juego está diseñado para ser usado por un Jugador, cuyo propósito es experimentar sobre deficiencias visuales a través de la simulación de sus efectos.

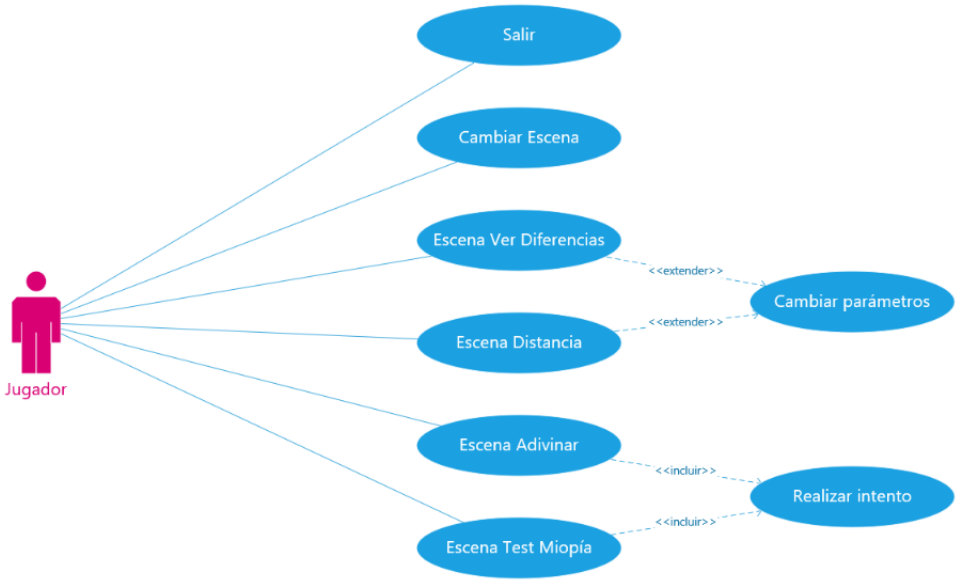


Figura 21 Diagrama de casos de uso de Jugador.

3.3.3. DIAGRAMA DE FLUJO DE NAVEGACIÓN DEL JUEGO EN UNITY

En la Figura 22 mostramos el diagrama de flujo de navegación del Juego Serio desarrollado. Tras iniciar la aplicación, el jugador puede elegir interactuar con el menú de navegación o no hacerlo. Si decide no hacerlo, su única opción es interactuar con la pantalla actual hasta que decide volver a interactuar con el menú. Si lo hace, tendrá que decidir si ir a otra escena o no hacerlo. En caso de hacerlo, la escena actual cambiará y volverá a un estado original (elegir usar el menú o no), en caso contrario podrá pulsar el botón de salir para cerrar la aplicación, y de no hacerlo volver al estado original de navegación.

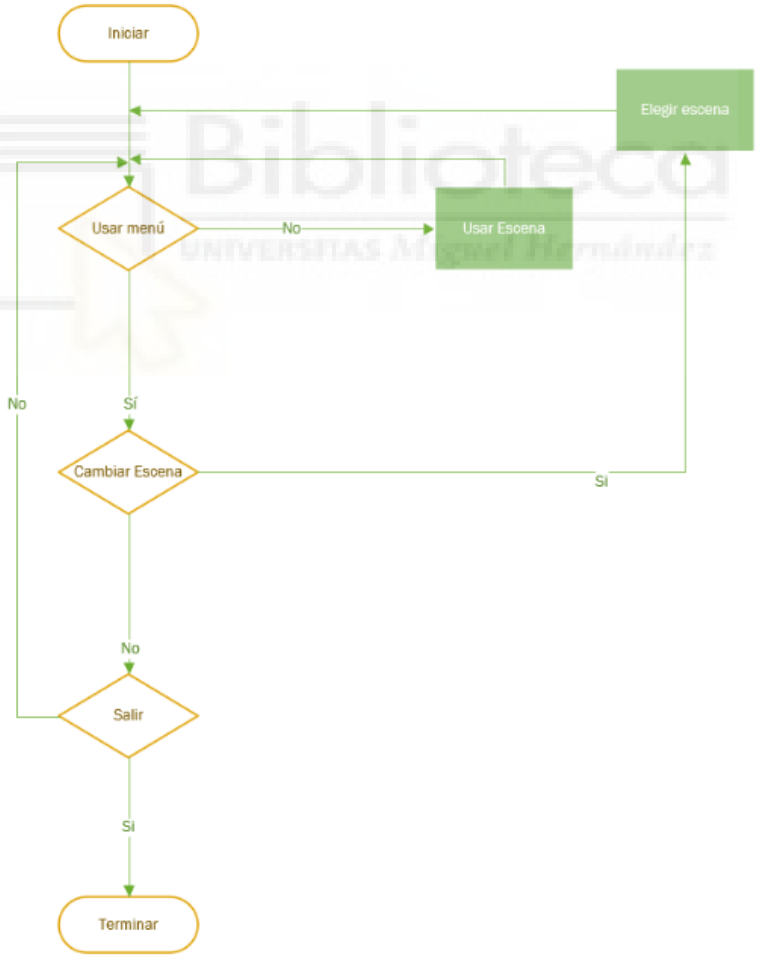


Figura 22 Diagrama de flujo del Juego Serio.

Cabe destacar que interactuar con el botón de ayuda no se contempla dentro de la interacción con el menú, pese a estar ubicado dentro de este. Esto se debe a que el botón de ayuda interactúa con la escena, haciendo visible la ayuda. Debido a esto, he decidido que la interacción con ese botón se engloba en el uso de la escena actual.

3.3.4. DISEÑO 3D

El diseño e implementación de la aplicación 3D se puede dividir en 2 apartados bien diferenciados:

- **Diseño de las escenas:** Se centra en el diseño de interfaces y componentes con los que se va a interactuar, principalmente elementos Canvas, RawImage y elementos de interacción como Slider, Toggle, Dropdown y otros elementos de interacción típicos de las interfaces. Estos componentes serán luego vinculados a Scripts escritos en C#, que se incluyen en los propios canvas para dar funcionalidad a los elementos.

- **Diseño de entornos:** Se centra en el diseño del decorado, incluyendo objetos en tres dimensiones descargados de la tienda de assets de Unity (*Unity Asset Store - The Best Assets for Game Making*, 2022). Estos assets deberán ser adaptados en color y tamaño. En color, porque a menudo aparecen con errores en las texturas y no parecen realistas, y en tamaño para que tengan un aspecto normal cuando se use la aplicación con dispositivos RV. Esto hace que para su ajuste es imprescindible la pruebas con el hardware adecuado, además de distintos usuarios para llegar a unos tamaños más realistas.

3.4 ASPECTOS RELEVANTES DE LA IMPLEMENTACIÓN CON RV

3.4.1. GESTIÓN DE ESCENAS

La interfaz de la aplicación se ha realizado en Unity, donde se han generado una ventana de inicio y varios escenarios, cada una conteniendo una prueba sobre la visión. En las escenas diseñadas para usarse con corrección se usan las imágenes procesadas y no suelen mover el panel donde aparecen las imágenes. La excepción a esto es la escena Acercar, que muestra imágenes modificadas a la distancia que les corresponde, con el fin de demostrar que la hipermetropía es más difícil de detectar, ya que pese a estar distorsionada, una imagen más grande (por estar más cerca) es más fácil de reconocer.

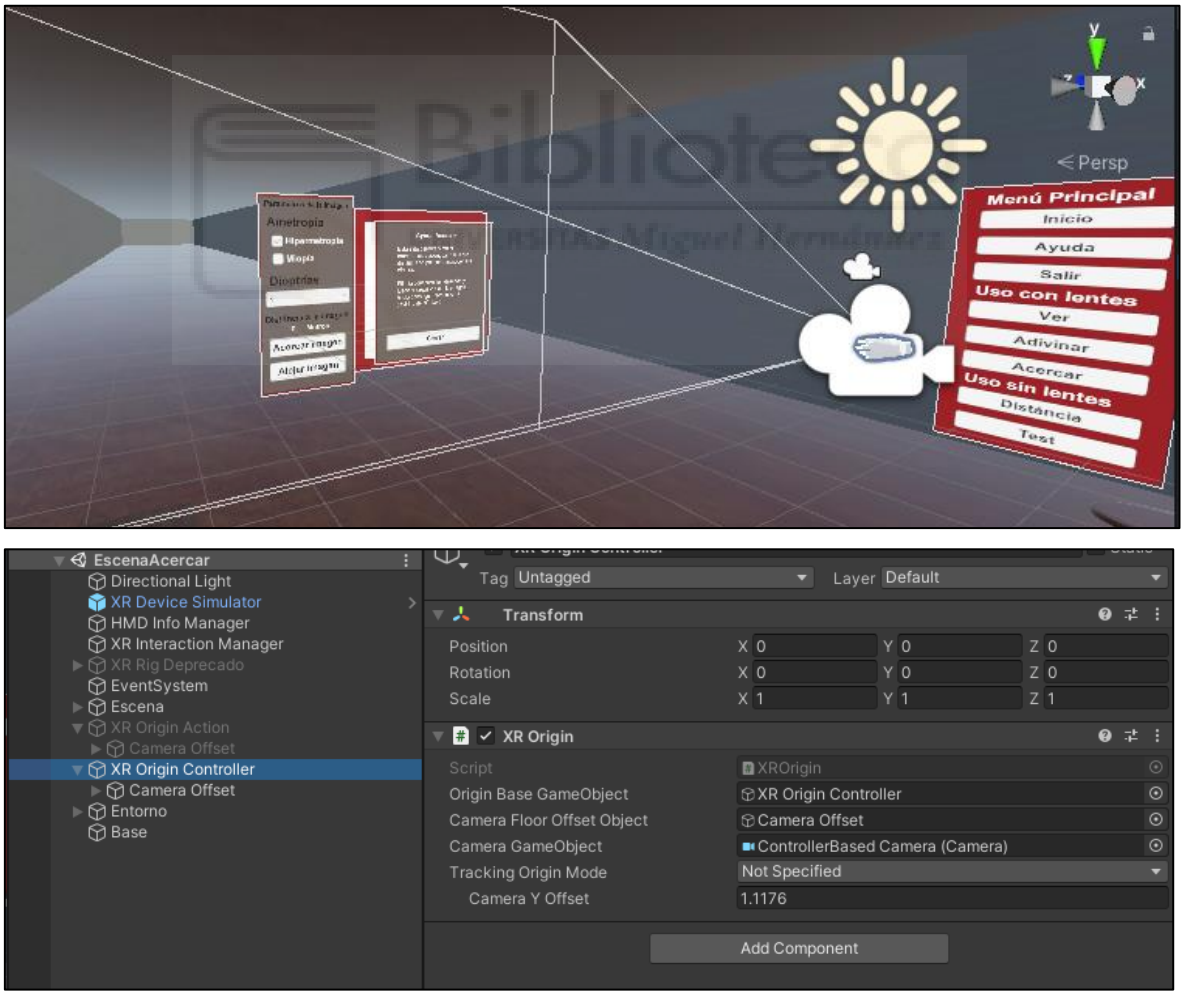


Figura 23 Controladores XR en el proyecto.

3.4.2. GESTIÓN DE CÁMARAS

Debido al desarrollo en dos dispositivos, uno de ellos emulado, han sido necesario usar dos componentes distintos para los controladores, "`ControllerBased Camera`" y "`ActionBased Camera`", ya que el dispositivo que simula el dispositivo de RV, "`XR Device Simulator`", solo funciona con `ActionBased`, pero al usar ese controlador con el dispositivo Oculus, ni los mandos ni el HMD detectaban la rotación, por lo que se ha tenido que optar por tener ambos componentes presentes y deshabilitar el que no se vaya a usar, como se puede observar en Figura 23.

3.4.3. FUNCIONALIDAD DE LOS SCRIPTS C# E INTERACCIONES

Las escenas están diseñadas en Unity, pero la funcionalidad la dan los scripts escritos en C# que hay asociados a los canvas. En cada escena, el script está asociado a un elemento canvas de la escena. No todos los datos que manipula el script deben estar dentro del mismo componente. Por ejemplo, en Figura 24, los contadores en blanco de los canvas laterales están asociados al script para que este pueda manipular sus valores.

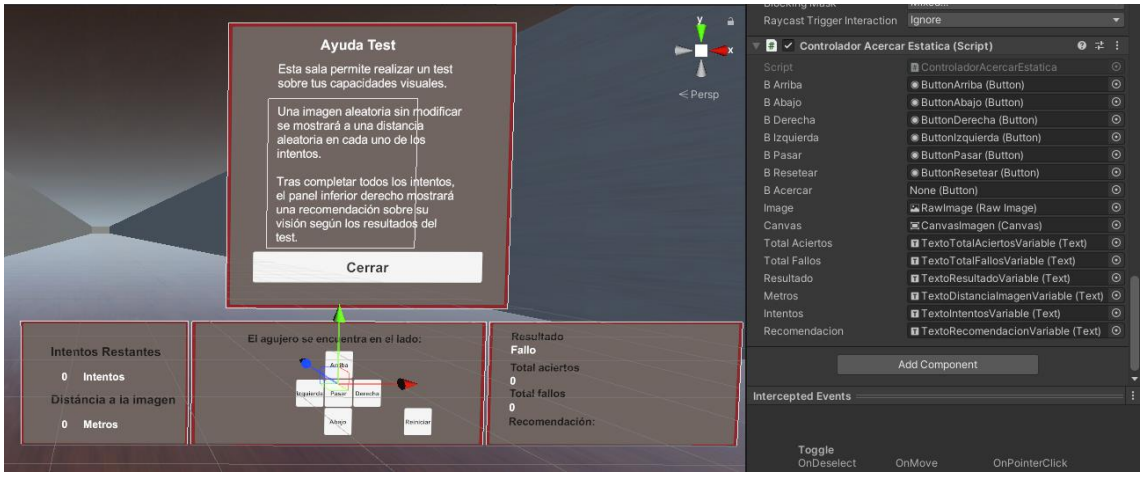


Figura 24 Script en escena Test.

Otro ejemplo de este punto es el script que controla el “Menú Principal”, que es el mismo en todas las escenas, y controla el panel de cerrar ayuda, que en realidad desplaza el panel a una altura superior al techo. Incluso en la escena principal, donde no se puede cerrar el menú de ayuda, existe un botón (con anchura 0 para no ser detectable) que está conectado al script, porque de lo contrario el script no funciona.

El cambio entre escenas del menú se basa en la configuración de `build` de Unity (ver Figura 25), donde se pueden asignar distintas escenas para incluirlas al compilar el juego. Al incluirlas se les asigna un identificador único, que es el que usa el Menú Principal para cargar la escena.

Figura 25 Ajustes de build del proyecto.

Además, en varias escenas y componentes del proyecto se ensombrecen botones para indicar que están deshabilitados. Esto se hace porque hay funciones condicionales que evitan que los botones apliquen el código si no se cumplen ciertas condiciones. El color se usa simplemente como indicador. Dicho esto, hay un caso especial. Los botones de selección de la escena `Test` pierden los `onClickListeners` al volverse grises. Se les borran para evitar que pulsarlos altere el resultado ya obtenido. Al pulsar Reiniciar, se vuelven a generar y asociar los listeners.

```
2 referencias | Pasqual Martínez Botella, Hace 57 días | 1 autor, 1 cambio
void DeshabilitarJuego() {
    BArriba.onClick.RemoveListener(ClickArriba);
    BAbajo.onClick.RemoveListener(ClickAbajo);
    BDerecha.onClick.RemoveListener(ClickDerecha);
    BIzquierda.onClick.RemoveListener(ClickIzquierda);
    BPasar.onClick.RemoveListener(ClickPasar);

    BArriba.GetComponent<Image>().color = Color.gray;
    BAbajo.GetComponent<Image>().color = Color.gray;
    BDerecha.GetComponent<Image>().color = Color.gray;
    BIzquierda.GetComponent<Image>().color = Color.gray;
    BPasar.GetComponent<Image>().color = Color.gray;

}

1 referencia | Pasqual Martínez Botella, Hace 57 días | 1 autor, 1 cambio
void ResetearBotones() {
    DeshabilitarJuego();
    BArriba.onClick.AddListener(ClickArriba);
    BAbajo.onClick.AddListener(ClickAbajo);
    BDerecha.onClick.AddListener(ClickDerecha);
    BIzquierda.onClick.AddListener(ClickIzquierda);
    BPasar.onClick.AddListener(ClickPasar);
    BArriba.GetComponent<Image>().color = Color.white;
    BAbajo.GetComponent<Image>().color = Color.white;
    BDerecha.GetComponent<Image>().color = Color.white;
    BIzquierda.GetComponent<Image>().color = Color.white;
    BPasar.GetComponent<Image>().color = Color.white;
}
```

Figura 26 Funciones que se encargan de controlar los botones de respuesta de la escena Test.

3.4.4. ALGORITMO DE PRUEBA DE VISIÓN EN LA ESCENA TEST

Existen escenas dinámicas en las que el usuario debe adivinar la posición de un agujero. Estas escenas son `Adivinar` y `Test`: La escena `Adivinar` se trata de un juego para el que es necesario el *feedback* de si se ha acertado o no.

Por ello, la escena `Test` intenta ser una prueba para determinar si es posible que una persona sufra deficiencias visuales de forma cuantitativa. Para implementar dicha prueba de visión se han definido variables y parámetros con los que se calcula un resultado, según el cual se hará una recomendación al usuario. Algunos aspectos relevantes de este algoritmo son:

- Se han definido tres umbrales de distancia en los que puede aparecer una imagen, cada uno representando distancias cortas (inferiores a 10 metros), medias (entre

10 y 100 metros) y largas (superiores a 100 metros). Cada umbral puede generar distancias dentro de sus propios límites.

- Hay variables que definen la distribución de los intentos máximos basadas en porcentajes, independientemente de cuantos se programe. A cada umbral se le asigna una cantidad de intentos acorde a estos porcentajes.

- A cada umbral también se le asocia un peso, que se usa para determinar una puntuación total mediante un promedio ponderado.

Se han definido recomendaciones para cuando este promedio supera cierto grado de errores. Esta forma de generar la recomendación se diseñó pensando en que fuera fácil su calibrado mediante las constantes del programa, y que se pudiera generar fácilmente un segundo test para medir la ametropía contraria, en este caso la hipermetropía. En este caso, también se contabilizan las veces que se pulsa pasar como errores (ya que el jugador no ha reconocido el símbolo) aunque no lo muestre por pantalla.

3.5 DESPLIEGUE Y PRUEBAS CON USUARIOS CONTROLADOS

Las primeras pruebas de despliegue en el dispositivo (ver Figura 27) hicieron evidentes varios fallos de diseño. Por ejemplo, los controladores no funcionaban al usarse con un dispositivo físico y hubo que cambiarlos. Otro ejemplo, se tuvo que ajustar alturas y distancias de los componentes de todas las escenas, ya que los paneles aparecían más altos y lejanos de lo que se pretendía durante su desarrollo. Además, hubo que asegurarse de que las escenas y el entorno se veían bien, tanto de pie como sentado (ver Figura 28).

Figura 27 Oculus Rift S en las que se ha desplegado el proyecto.

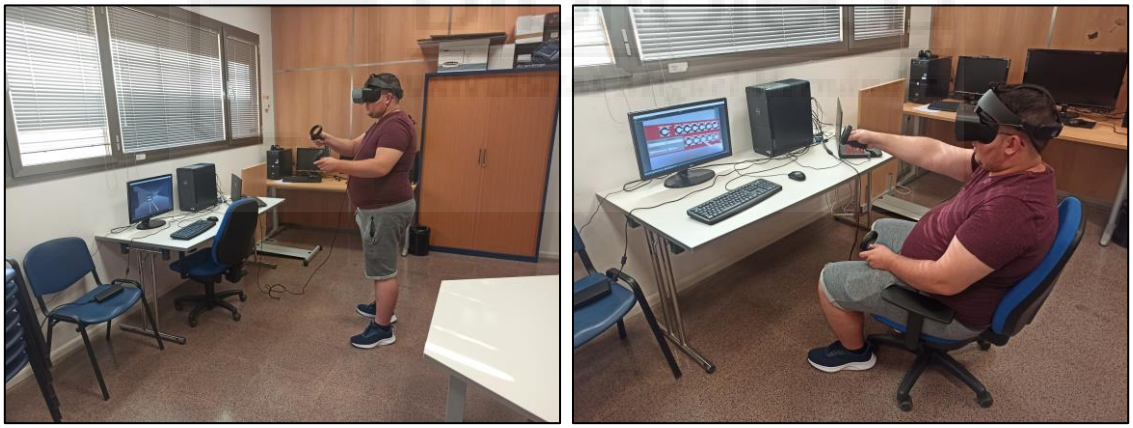


Figura 28 Pruebas de desarrollo. De pie y sentado.

Una consecuencia inesperada en un primer momento fue que el ojo altera las imágenes que ve a través de las Oculus como si se encontraran a la distancia a la que las percibe. Debido a esto, varias escenas que combinaban distancia y aplicación de ametropías con el pre-procesado tuvieron que ser adaptados o borrados, debido a la redundancia del efecto.

3.6 EJEMPLO DE USO DEL JUEGO SERIO

En este apartado exploraremos la aplicación como lo haría un jugador. Al iniciar la aplicación, apareceremos en la escena de inicio frente a un panel que explica el objetivo de este juego e indica la posición del menú (Figura 29). Si observamos el entorno podemos se está simulando una sala de una oficina genérica (Figura 30).

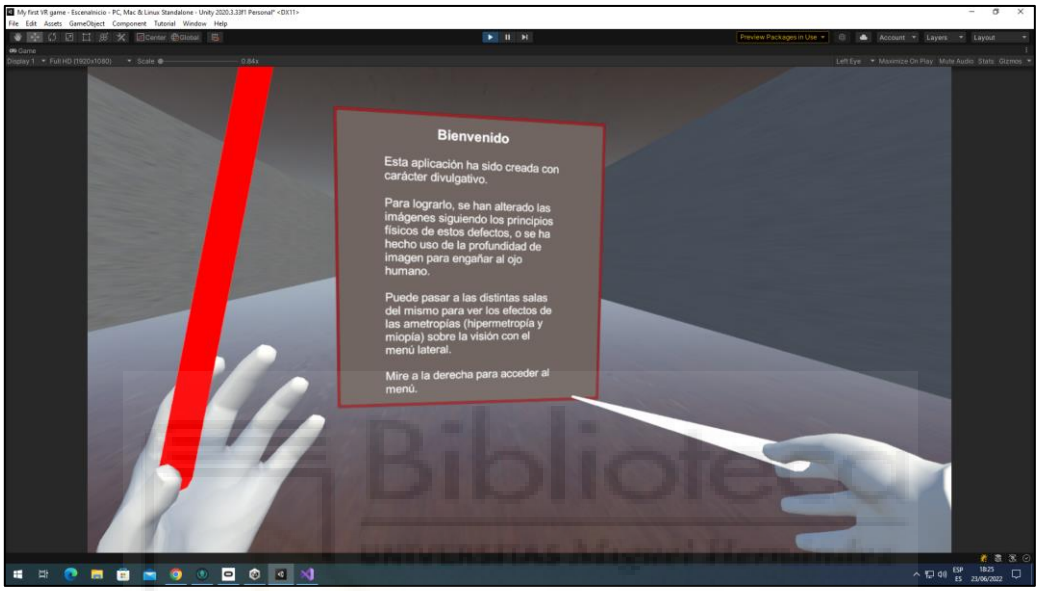


Figura 29 Escena inicial del Juego Serio.

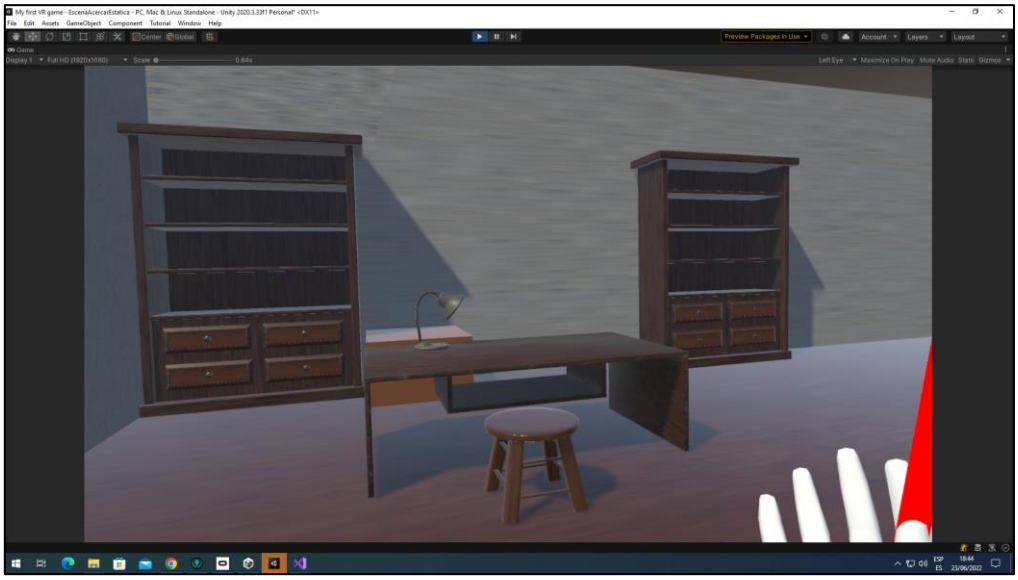


Figura 30 Entorno del Juego Serio.

También encontraremos el menú de navegación, en el que podemos ver que hay un botón ensombrecido (Figura 31). Al pulsarlo no hará nada, pero si pulsamos cualquier otro hará que se sombree y el botón de Inicio se vuelva blanco (Figura 32).

Figura 31 Menú en estado inicial.

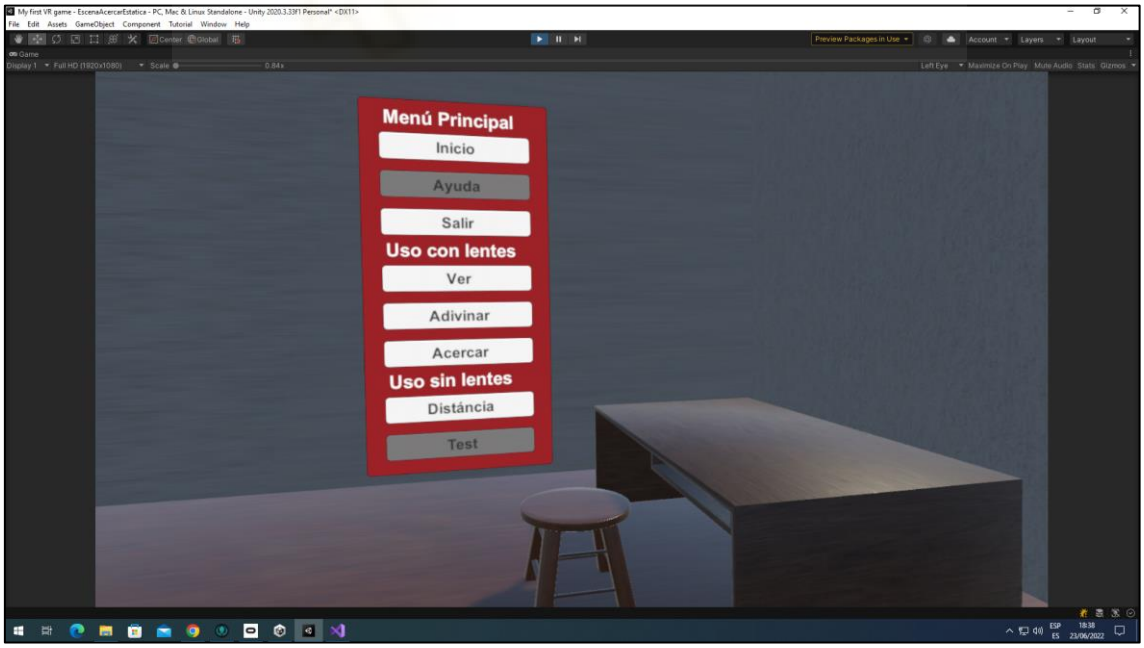


Figura 32 Menú al pulsar en otra escena.

Si volvemos a mirar al frente, dicho de otro modo, la dirección a la que estábamos mirando al iniciar el juego, podremos ver que el panel informativo ya no está, pero encontramos varios paneles, con el más cercano explicando cómo funciona la escena.

A continuación, vamos a revisar las escenas por el orden en que aparecen por pantalla.

3.7.1. DESCRIPCCIÓN DE LA ESCENA “VER”

Tal y como se informa al usuario en el panel de ayuda, el objetivo de esta sala es:

> *“Esta sala muestra las diferencias visuales de una misma imagen a distancias regulares. Se pueden cambiar los parámetros de la distorsión con el menú en la parte inferior izquierda del panel.”*

Lo primero que veremos al volver a mirar al frente será parecida a la Figura 33. Si consultamos el menú lateral, ahora el botón de Ayuda aparecerá en blanco. Si lo pulsamos, se ensombrece y el panel de ayuda vuelve a aparecer delante. Tras leer la ayuda, pulsamos en cerrar y encontramos con un escenario parecido al de la Figura 34.

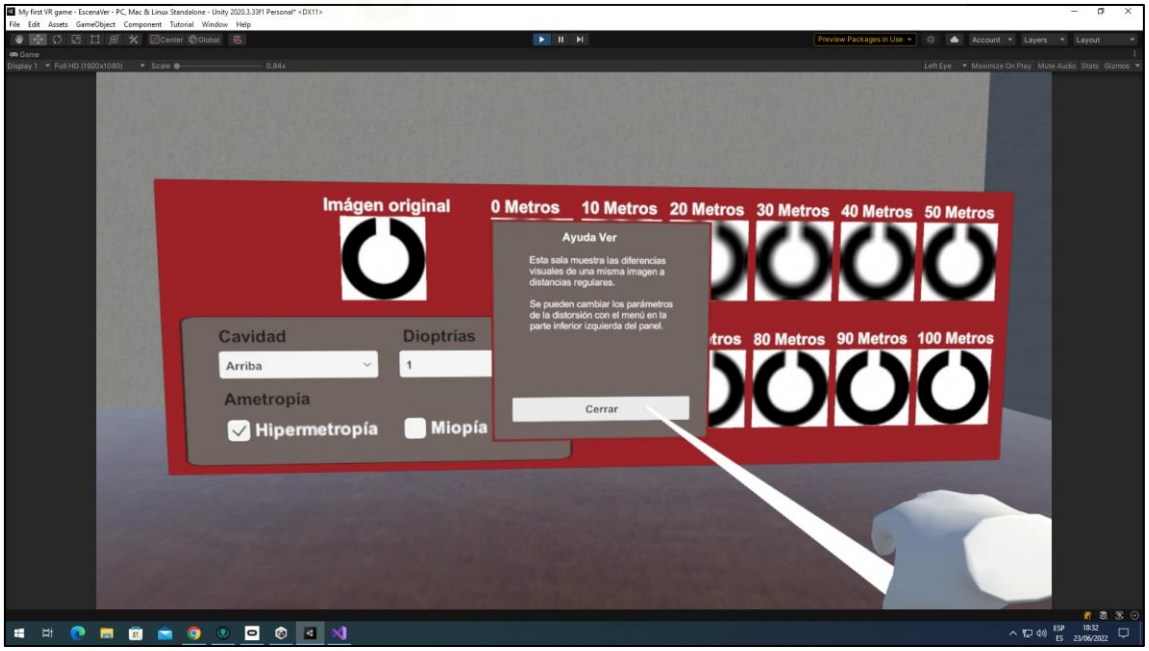


Figura 33 Escena ver con Ayuda abierta.

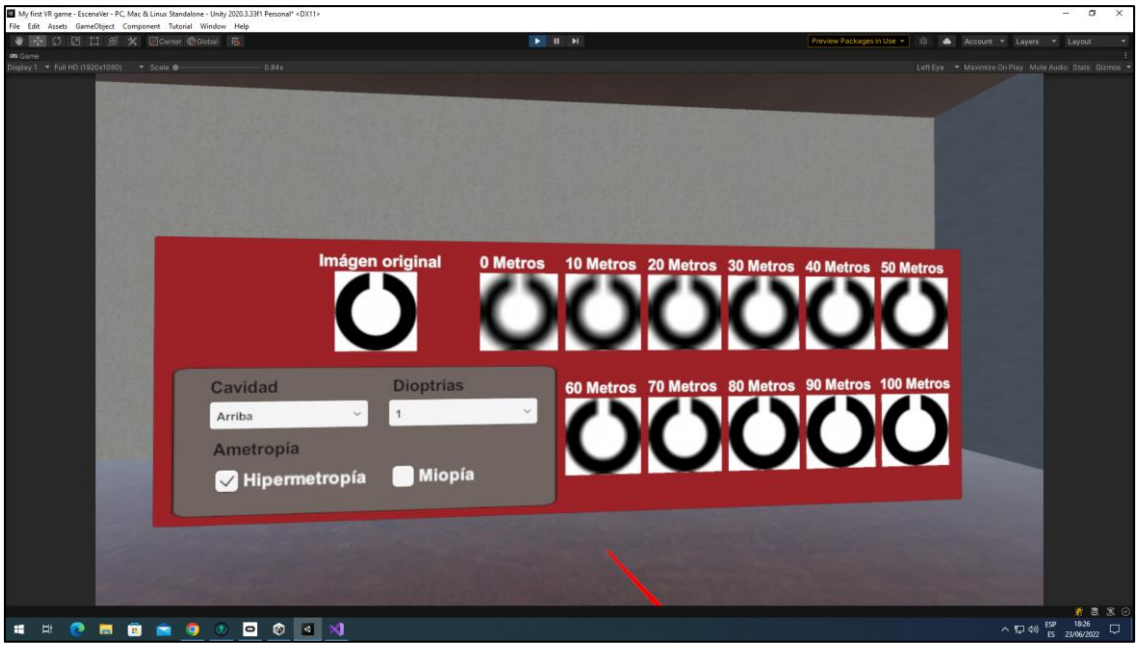


Figura 34 Escena Ver sin ayuda.

En esta escena, "Ver", podemos usar el menú a la izquierda para alterar los parámetros de distorsión que se le aplican a las imágenes, que aparecen unas junto a las otras con letreros para que se pueda apreciar la diferencia entre ellas. En la Figura 35 podemos ver como las imágenes mostradas cambian y son distintas a las mostradas en la Figura 34.

Figura 35 Escena ver, parámetros alterados.

3.7.2. DESCRIPCCIÓN DE LA ESCENA “ADIVINAR”

La siguiente escena es “Adivinar”, y su explicación aparece en un panel similar al que ya hemos visto en la escena anterior. Su objetivo es el siguiente:

> *“Esta sala muestra imágenes alteradas de forma aleatoria. Los parámetros de la alteración se muestran en el panel izquierdo. El panel inferior registra las respuestas. Tanto el resultado actual como el acumulado se mostrarán con más detalle en el panel derecho.”*

En esta escena (Figura 36), tenemos tres paneles auxiliares: El izquierdo muestra los parámetros de los cambios que se le están aplicando a la imagen mostrada, el inferior permite elegir qué imagen es la correcta, y el derecho muestra los resultados de la sesión, tanto en total como en concreto el último intento.

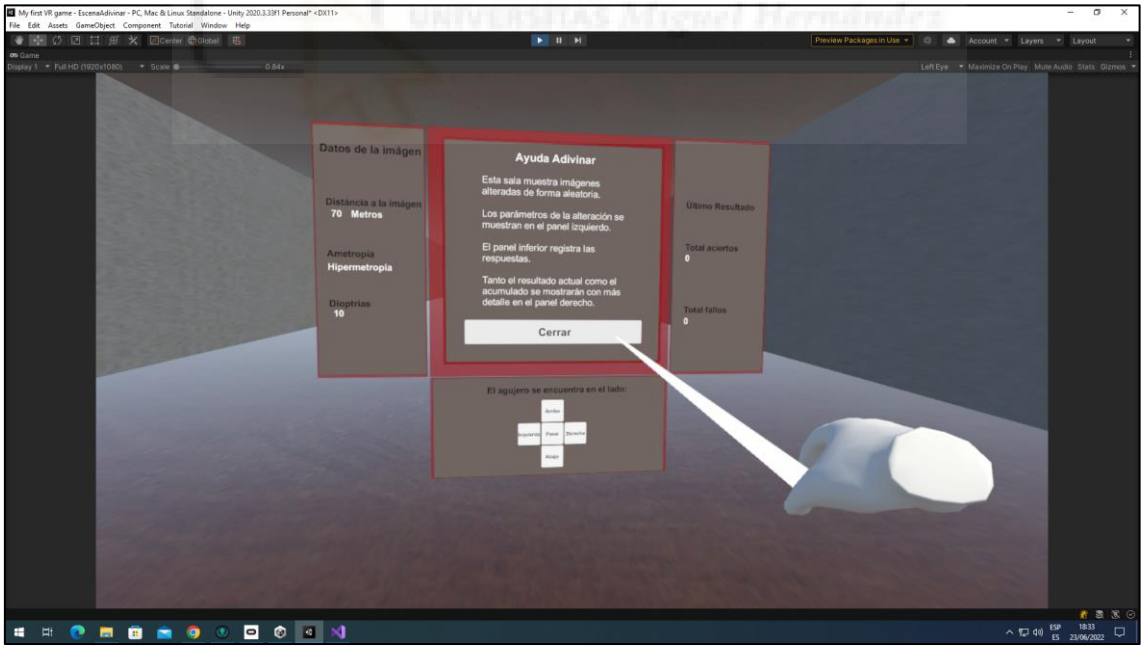


Figura 36 Escena Adivinar con Ayuda abierta.

En las Figura 37 y Figura 38 podemos observar las diferencias según el progreso de la escena, se ha hecho zoom sobre las capturas originales para que se lean mejor los textos.

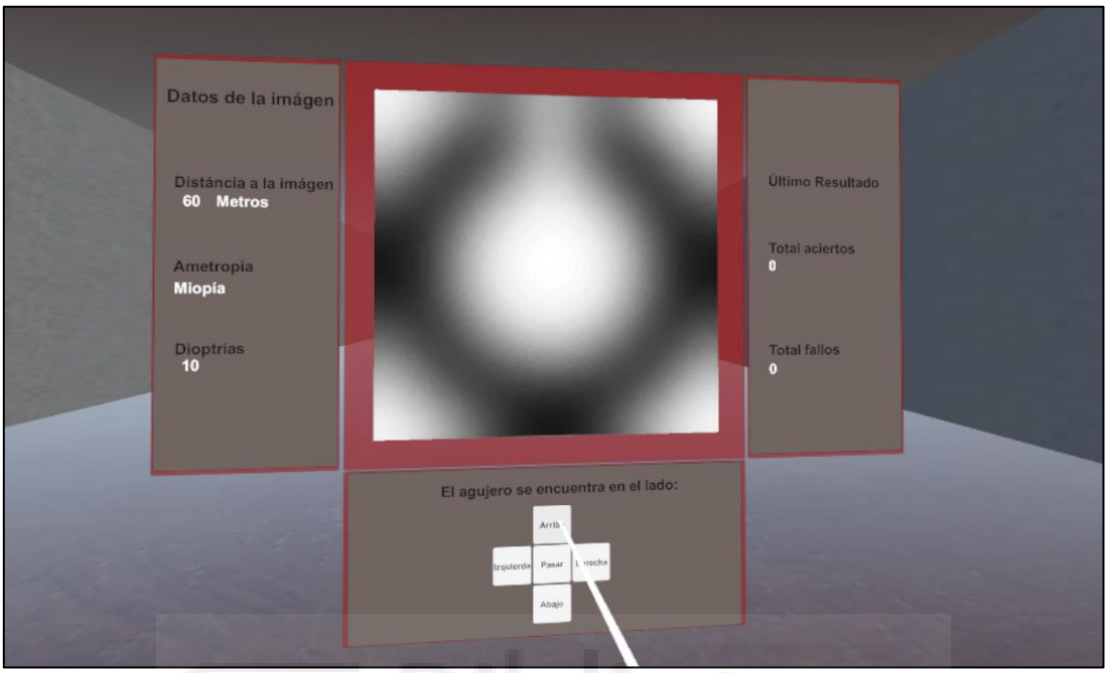


Figura 37 Escena Adivinar, Sin contestar ningún intento.

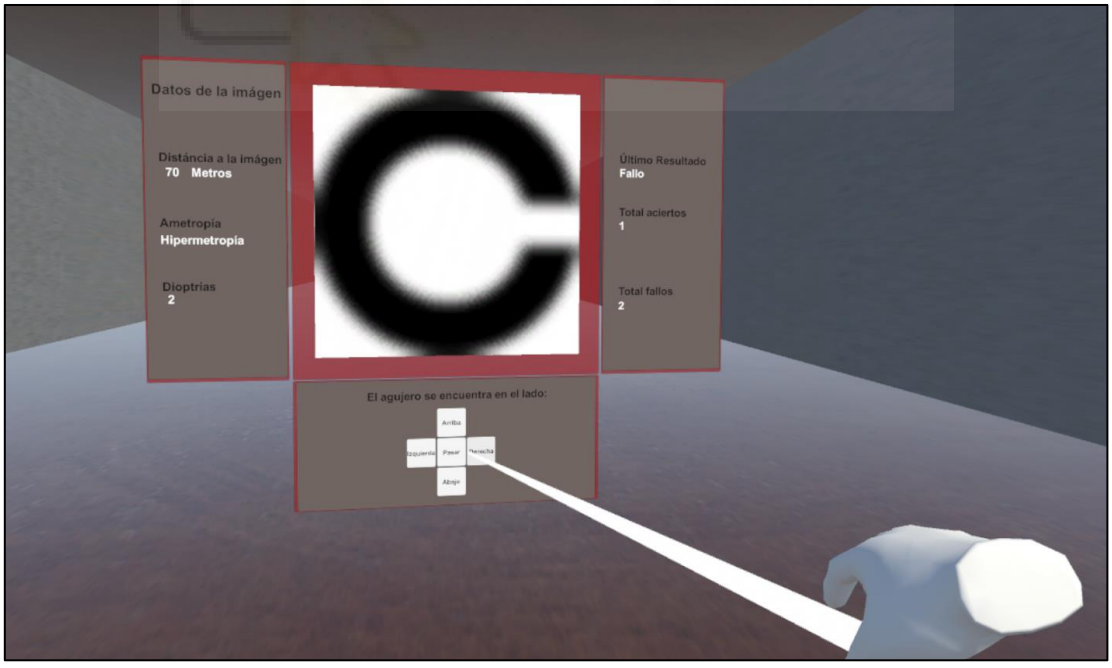


Figura 38 Escena adivinar, tras contestar varios intentos. En el menú derecho se puede ver el resultado del último intento, en el primer texto blanco.

3.7.3. DESCRIPCCIÓN DE LA ESCENA “ACERCAR”

La siguiente escena es “Acercar”, y como explica el mensaje de ayuda de la Figura 39, su objetivo es el siguiente:

> *“Esta sala permite simular ametropías y averiguar hasta que distancia se pueden reconocer sus efectos. Elija la configuración deseada y luego acerque o aleje la imagen hasta conseguir reconocer la cavidad del símbolo.”*

Como podemos ver en la figura seguimos pudiendo cambiar algunos parámetros. Estos son la ametropía que simularemos y su intensidad. Una miopía será más fácil de detectar a menor distancia, y una hipermetropía cuanto más lejos esté la imagen, ya que las distorsiones serán menores.

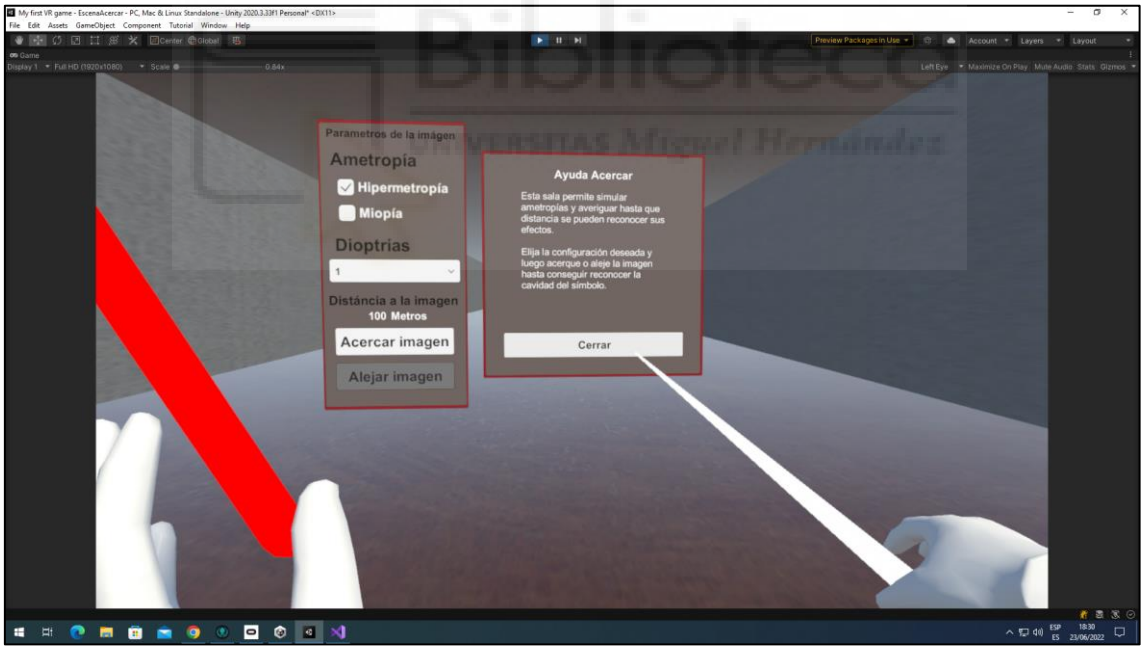


Figura 39 Escena Acercar con ayuda.

Aun así, la distancia también es un factor a tener en cuenta al reconocer una imagen, como podemos ver en la Figura 40 y Figura 41. Pese a estar menos distorsionada, la mayor distancia afecta a nuestra capacidad de reconocer el agujero en la imagen.

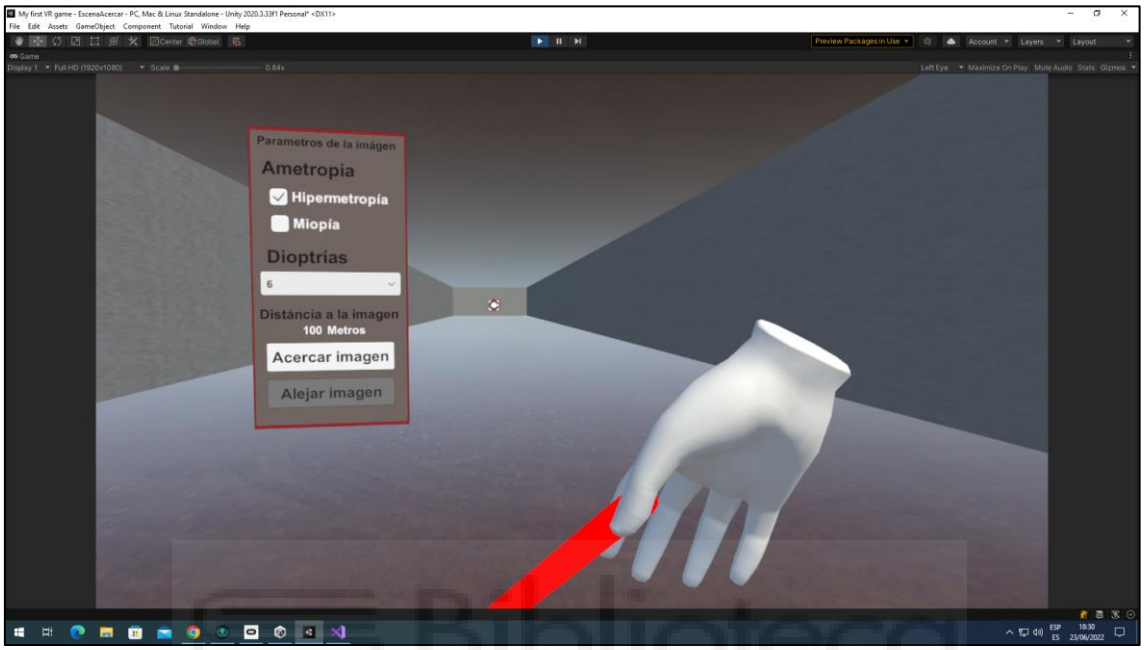


Figura 40 Escena Acercar, Imagen vista con hipermetropía de lejos.

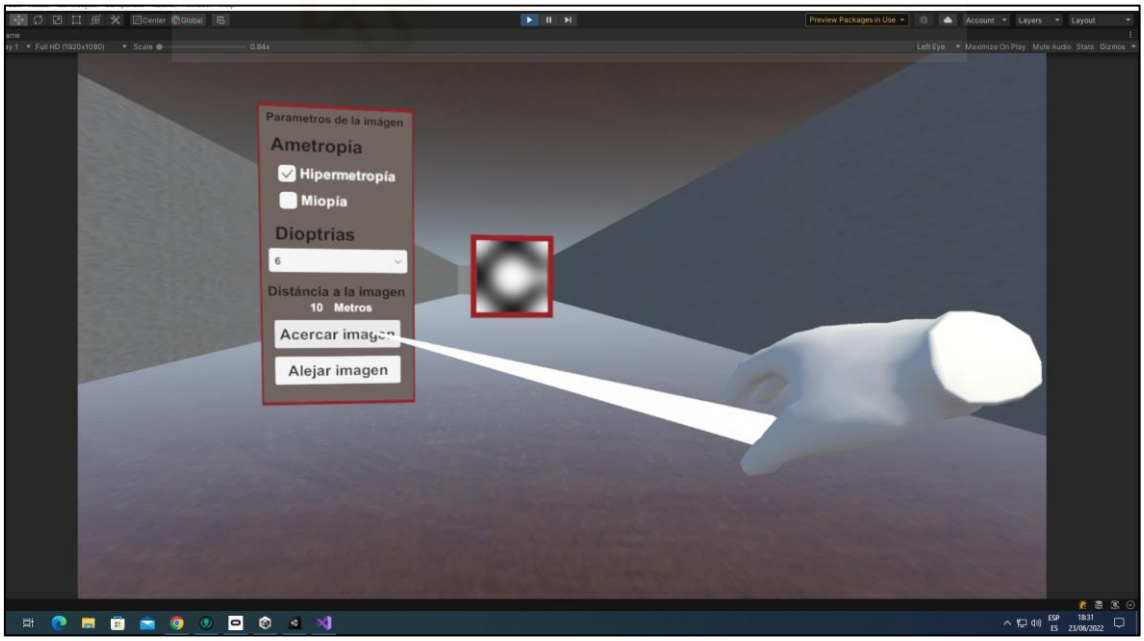


Figura 41 Escena Acercar, Imagen vista con miopía de cerca.

3.7.4. DESCRIPCCIÓN DE LA ESCENA “DISTANCIA”

Como se ha comentado anteriormente, las últimas dos escenas están diseñadas para realizarse sin gafas, y esto se puede leer en el menú de navegación. El objetivo de esta escena es:

> *“Esta sala permite experimentar como las ametropías sufridas se aplican a imágenes mostradas en realidad virtual. Desplazando la Distancia de la pantalla se puede comparar como los efectos de las ametropías se aplican incluso en este entorno virtual.”*

Es decir, esta escena (Figura 42) nos permite mover una imagen sin modificar y cambiar la dirección de la cavidad (Figura 43). A pesar de que en las siguientes imágenes se vean con la misma claridad (Figura 43, Figura 44 y Figura 45), los defectos visuales sin corregir de los jugadores harán la imagen difícil de reconocer.

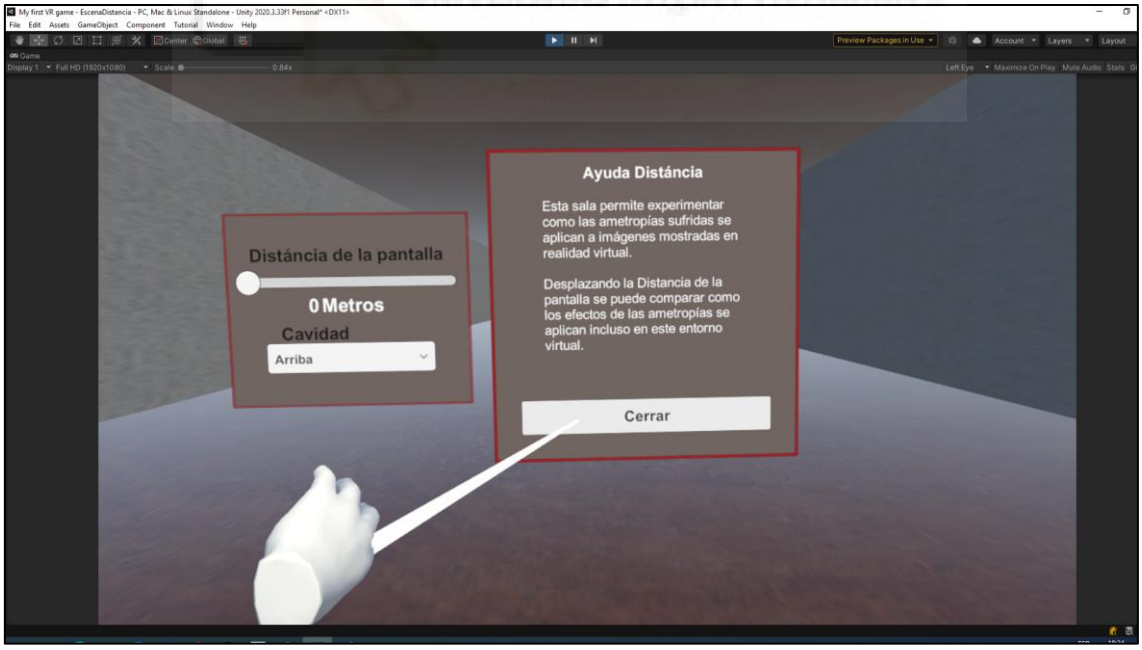


Figura 42 Escena Distancia con Ayuda.

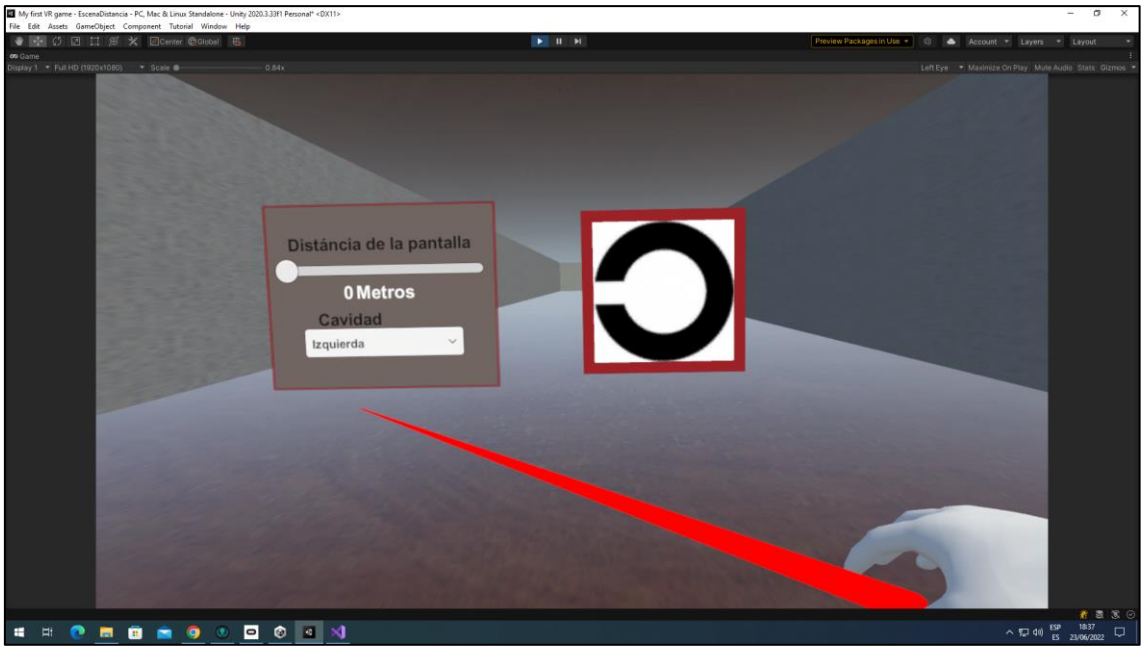


Figura 43 Escena Distancia, 0 metros.

Una actividad propuesta es comparar las distancias a las que se es capaz de reconocer cada persona las imágenes, y hacer comparaciones entre distintos jugadores.

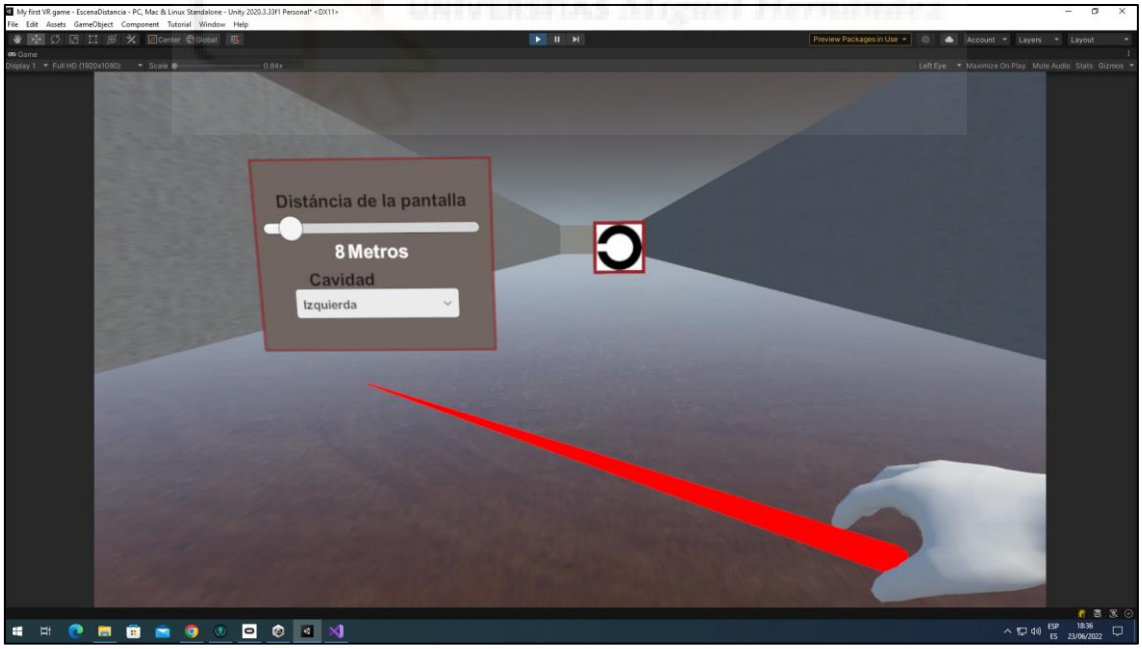


Figura 44 Escena Distancia, 8 metros.

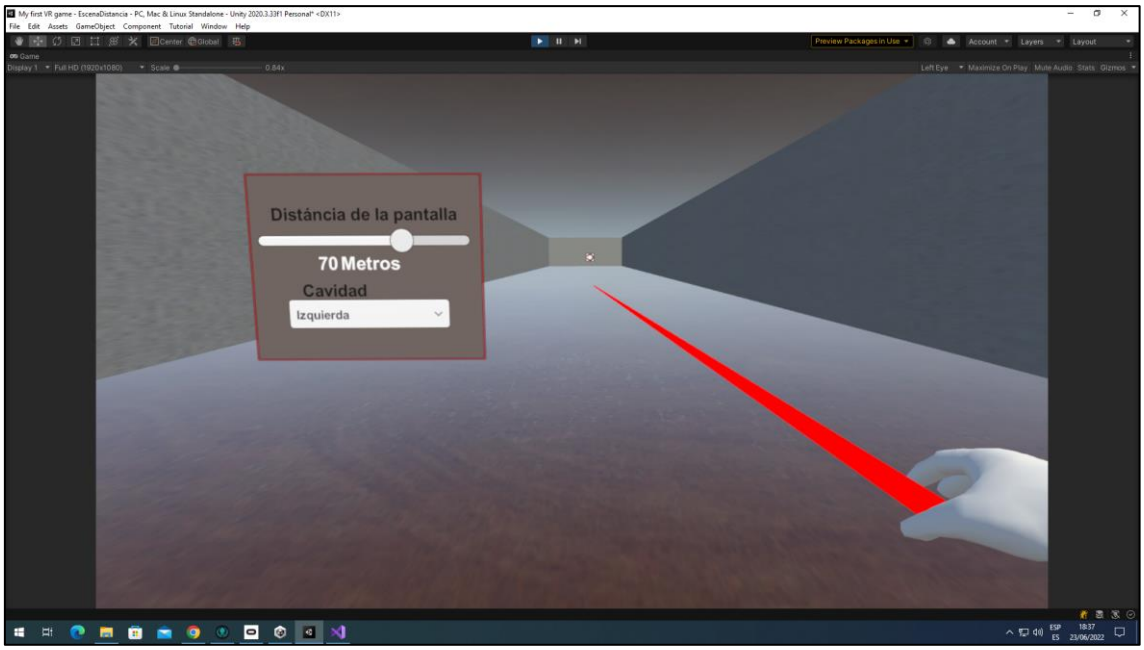


Figura 45 Escena Distancia, 70 metros

3.7.5. DESCRIPCCIÓN DE LA ESCENA “TEST”

El objetivo de esta escena es:

> *“Esta sala permite realizar un test sobre tus capacidades visuales. Una imagen aleatoria sin modificar se mostrará a una distancia aleatoria en cada uno de los intentos. Tras completar todos los intentos, el panel inferior derecho mostrará una recomendación sobre su visión según los resultados del test.”*

Es decir, esta última escena (Figura 46) permite hacer una prueba intentando reconocer 20 imágenes sin modificar a distintas distancias. La distribución de esta prueba se parece a la escena “Adivinar”, pero los paneles están todos al nivel del inferior para no impedir la visión de la imagen, que se desplaza desde los paneles hasta el fondo de la sala.

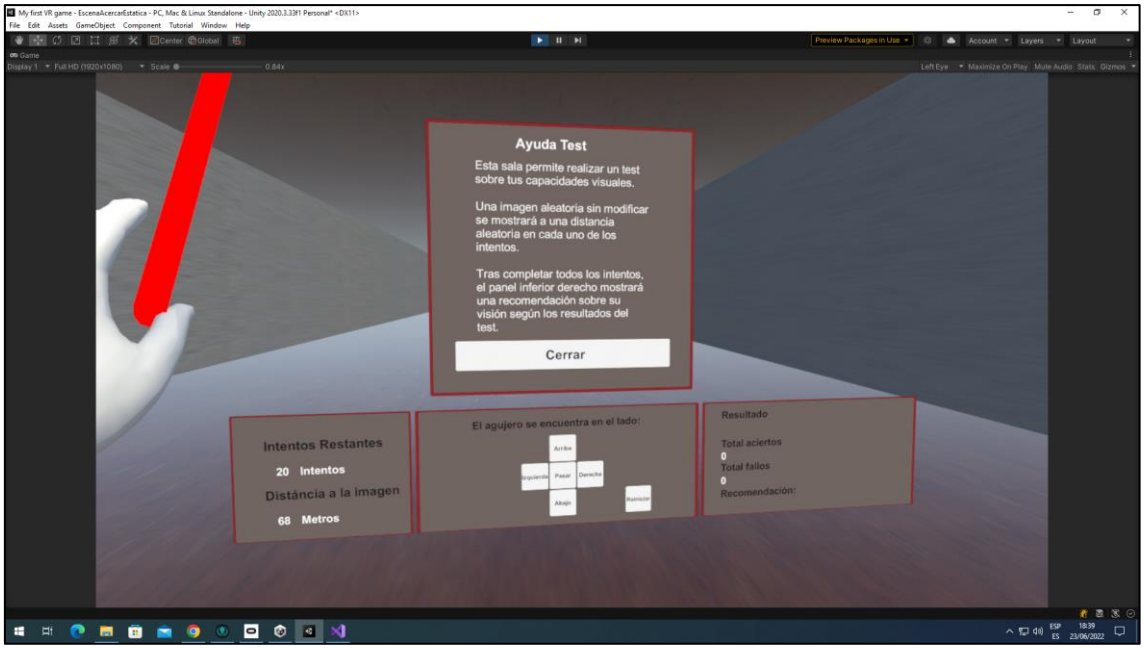


Figura 46 Escena Test con ayuda.

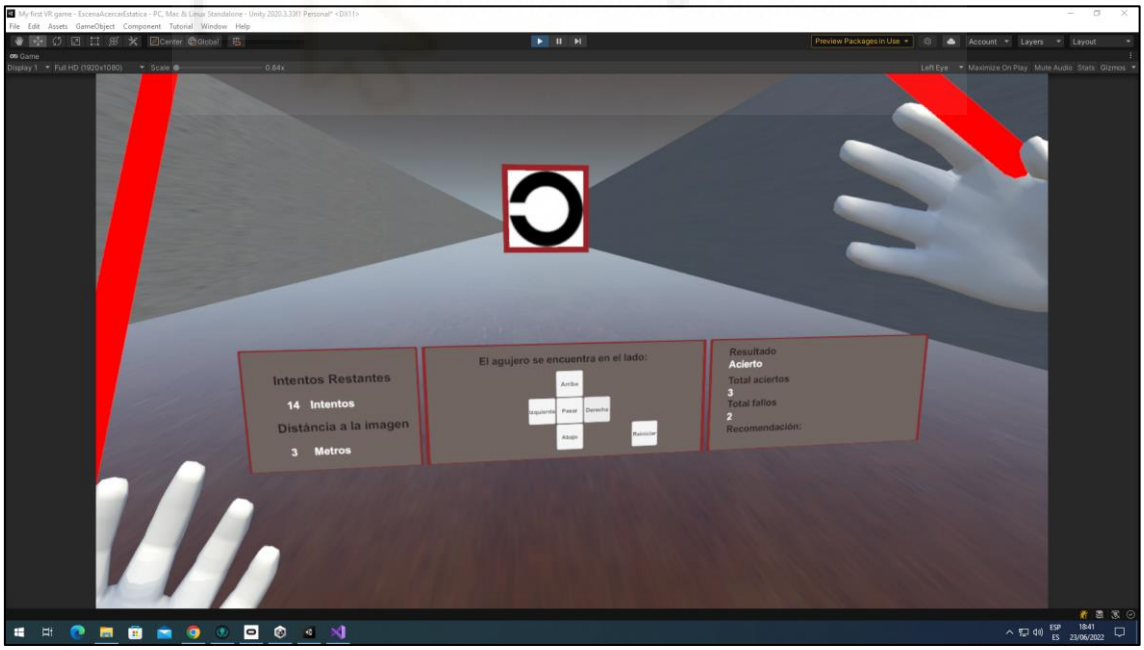


Figura 47 Escena Test, realizando la prueba.

En la Figura 47 podemos ver como la imagen no está modificada, pero si aparece a mayor distancia de lo que lo hacía en la escena “Adivinar”. Pulsar por error un botón que no queríamos pulsar podría cambiar el resultado del test. Para evitar esto, podemos pulsar el botón “Reiniciar”, situado abajo a la derecha de la cruz destinada a responder. Hacer esto devolverá los intentos al máximo y reiniciará los contadores que miden las respuestas, para poder volver a hacer el test con la mayor precisión posible.

Finalmente, como podemos ver en la Figura 48, al acabarse los intentos se bloquean los botones de respuesta, y aparece una recomendación en el panel de resultados, al que las manos están apuntando. Como podemos ver en la Figura 47, este espacio estaba vacío mientras se realizaba el test.

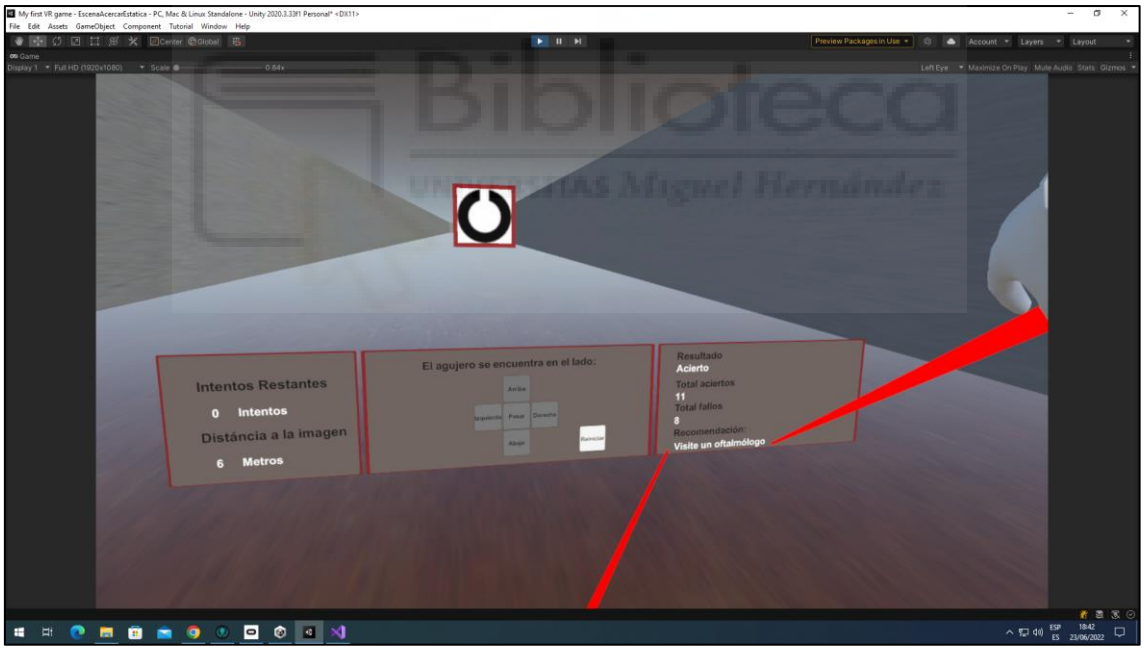


Figura 48 Escena Test, resultado final

Con esto, hemos terminado el ejemplo de uso del Juego Serio, ya que hemos cubierto todos los apartados de la aplicación.

4. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se aborda la construcción de una aplicación de Realidad Virtual (RV) para la simulación de ametropías (miopía e hipermetropía), siguiendo una metodología basada en los principios del Manifiesto Ágil.

En línea con los objetivos planteados inicialmente, podemos concluir que se han cumplido de forma satisfactoria, logrando la mayoría de los objetivos planteados durante las reuniones semanales de la planificación. El objetivo general se ha logrado, ya que hemos sido capaces de desarrollar una aplicación de RV y desplegarla en un dispositivo Oculus Rift S. Esta aplicación ofrece a los usuarios, a través de sus cinco escenarios, ser parte de un Juego Serio con actividades que van desde la concienciación hasta la detección de las dos ametropías mencionadas.

En cuanto a los objetivos secundarios, el estudio de APIs y frameworks para dearrollar aplicaciones de RV desplegables en dispositivos Oculus nos llevó a elegir el entorno de desarrollo Unity usando las librerías de desarrollo VR multiplataforma, que si bien no nos han permitido usar los controles de RV propios del dispositivo físico que estamos utilizando a máxima potencia, hacen que el juego se pueda exportar a dispositivos compatibles con Unity, aunque no hemos podido probarlo al no tener el hardware necesario. Además, durante esta búsqueda se identificó que parte del desarrollo debería centrarse en diseñar el 3D del mundo virtual para lo que se decidió utilizar los elementos básicos de Unity combinados con *assets* 3D descargados de bibliotecas externas. Aunque en un principio se quiso utilizar la manipulación de las cámaras de Unity para simular las deficiencias, el análisis de los fundamentos detrás de las ametropías nos llevó a redirigir esta búsqueda a otros entornos, lo que finalmente se resolvió implementando el ejecutable que hace uso de OpenCV para simular las deficiencias sobre un conjunto de imágenes

originales, para las que decidimos utilizar las 4 Cs de Landolt. Este marco de trabajo ha sido adecuado ya que nos ha permitido desarrollar la aplicación anteriormente mencionada que hemos podido utilizar para realizar unas pruebas preliminares con usuarios.

4.1 TRABAJO FUTURO

A partir de las pruebas preliminares realizadas y la experiencia que hemos adquirido a lo largo del proyecto se han detectado una serie de limitaciones que se proponen como trabajo futuro:

- **Mejora del algoritmo para alterar de forma adecuada los bordes de las imágenes:** Una limitación que hemos podido observar en el ejecutable es que la distorsión de las imágenes causa que cuando las partes la C de Landolt cercanas a los bordes se deformen de forma extraña y sean fáciles de reconocer la cavidad debido a una acumulación excesiva de píxeles oscuros. Este problema desaparecía al añadir espacio en blanco en todas las direcciones, lo que hacía que las imágenes se distorsionaran de forma adecuada, pero el mayor número de píxeles por imagen reducía el efecto de las deformaciones. Se propone alterar el ejecutable y las imágenes originales hasta encontrar un equilibrio que, mientras siga siendo adecuado al efecto de las ametropías, no provoque este efecto no deseado.

- **Integrar el ejecutable basado en OpenCV dentro de Unity:** en la versión actual, el uso de las imágenes manipuladas con las Cs de Landolt se incluye estáticamente como recursos del proyecto de RV. Esta decisión se debió a que se encontraron problemas al integrar la librería C++ de simulación en el proyecto de Unity ya que la dependencia con OpenCV generaba problemas. Como trabajo

futuro creemos que se podría encapsular el ejecutable como un servicio web que permita realizar las transformaciones dinámicas directamente desde las gafas.

- **Pruebas más allá de las Cs de Landolt y ofrecer personalización:** si se consigue la mejora anterior, se podría valorar ampliar la funcionalidad de la aplicación anterior ofreciendo escenarios que permitan trabajar con imágenes dinámicas elegidas por el usuario. Además, sería interesante incorporar la personalización de los objetos de la sala, ya que permitiría ofrecer la aplicación a entidades interesadas que podría recrear sus propios espacios en un entorno virtual.

- **Desplegar el proyecto en otros dispositivos:** actualmente se han desarrollado ejecutables compatibles con otros dispositivos de RV, sin embargo, no hemos podido probar si funciona correctamente ya que no disponemos de dicho hardware.

- **Realizar un piloto con usuarios:** nos gustaría poner en práctica la aplicación en un entorno real con usuarios. Esto nos permitiría evaluar la usabilidad de la aplicación desarrollada y también permitiría ajustar mejor los umbrales definidos en el test de visión implementado en uno de los escenarios.

- **Aumentar la funcionalidad y el número de escenarios:** los escenarios actuales han sido elegidos a partir del análisis de aplicaciones relacionadas y la experiencia que íbamos adquiriendo a lo largo del proyecto. No obstante, se podrían analizar entornos reales y recrear las pruebas realizadas allí dentro de la aplicación.

En conclusión, creemos que hemos desarrollado una aplicación novedosa que hace uso de una tecnología emergente como la RV en la simulación de ametropías.

5. BILIOGRAFÍA

AForge.NET. (2022). *Biblioteca de visión artificial e inteligencia artificial*. http://aforgenet.com/

All About the Eye Chart - American Academy of Ophthalmology. (2022). https://www.aao.org/eye-health/tips-prevention/eye-chart-facts-history

Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2000). *Manifesto for Agile Software Development*. http://agilemanifesto.org/principles.html

Danilova, M. V, & Bondarko, V. M. (2007). Foveal contour interactions and crowding effects at the resolution limit of the visual system. *Journal of Vision*, *7*(2), 25.1-18. https://doi.org/10.1167/7.2.25

DevIL. (2017). *DevIL - A full featured cross-platform Image Library*. http://openil.sourceforge.net/

ElTerritorio. (2018). *Realidad virtual para cirujanos y estudiantes de medicina*. https://www.elterritorio.com.ar/noticias/2018/08/28/592953-realidad-virtual-para-cirujanos-y-estudiantes-de-medicina

Epic Games Inc. (2022). *Unreal Engine for extended reality (XR): AR, VR & MR - Unreal Engine*. https://www.unrealengine.com/en-US/xr

Foerster, R. M., Poth, C. H., Behler, C., Botsch, M., & Schneider, W. X. (2016). Using the virtual reality device Oculus Rift for neuropsychological assessment of visual processing capabilities. *Scientific Reports*, *6*(1), 37016.

https://doi.org/10.1038/srep37016

Gómez Ulla de Irazazábal, F., & Ondategui-Parra, S. (2012). Informe sobre la ceguera en España. *Ernest & Young*, *0814*, 125. http://www.seeof.es/archivos/articulos/adjunto_20_1.pdf

Gunjan04, & CHPedersen. (2014). *How to Load Images Dynamically From A Particular Folder In unity C#*. Unity Forums. https://answers.unity.com/questions/854878/how-to-load-images-dynamically-from-a-particular-f.html

Häkkilä, J., Colley, A., Väyrynen, J., & Yliharju, A.-J. (2018). Introducing Virtual Reality Technologies to Design Education. *Seminar.Net*, *14*(1), 1–12. https://doi.org/10.7577/seminar.2584

Hayden, S. (2019). *Oculus Job Listing Points to Eye-tracking in "next gen AR/VR products" - Road to VR*. https://www.roadtovr.com/oculus-job-listing-eye-tracking-ar-vr/

INE. (2008). Encuesta de Discapacidad, Autonomía personal y situaciones de Dependencia (EDAD). *Instituto Nacional de Estadística*, 1–12. http://cantabria.fspugt.es/uploads/documentos/documentos_Comunicado_INE_15-feb-2008__1_809cbef7.pdf

Jones, P. R., & Ometto, G. (2018). Degraded Reality: Using VR/AR to simulate visual impairments. *2018 IEEE Workshop on Augmented and Virtual Realities for Good (VAR4Good)*, 1–4. https://doi.org/10.1109/VAR4GOOD.2018.8576885

LEA SYMBOLS® Distance Visual Acuity Chart – Lea-Test Ltd. (2022). https://leatest.com/vision-test-system/lea-symbols-distance-visual-acuity-chart/

Li, L., Yu, F., Shi, D., Shi, J., Tian, Z., Yang, J., Wang, X., & Jiang, Q. (2017). Application of virtual reality technology in clinical medicine. *American Journal of Translational Research*, *9*(9), 3867–3880. http://www.ncbi.nlm.nih.gov/pubmed/28979666

Lowood, H. E. (2013). *Virtual reality*. 12 Mar. 2013. https://www.britannica.com/technology/virtual-reality

Meta. (2021). *Introducing Meta*. https://www.youtube.com/watch?v=pjNI9K1D_xo

Meta. (2022a). *Especificaciones del sistema y requisitos mínimos de Rift S y Rift*. https://support.oculus.com/articles/getting-started/getting-started-with-rift/rift-s-minimum-requirements#specs

Meta. (2022b). *Productos de Facebook, herramientas para desarrolladores - AR/VR*. https://developers.facebook.com/products/#filter-id=ar-vr

MVTec Software Gmbh. (2022). *HALCON - The power of machine vision: MVTec Software*. https://www.mvtec.com/products/halcon

Naciones Unidas. (2022). *Salud - Desarrollo Sostenible*. https://www.un.org/sustainabledevelopment/es/health/

NASA Advanced Supercomputing Division. (2013). *NASA: Virtual Reality*. https://www.nas.nasa.gov/Software/VWT/vr.html

Navia Ávila, M. (2015). *Desarrollo de prototipo de juego serio para ejercitación ocular basado en tracking de mirada*. https://repository.unimilitar.edu.co/handle/10654/13798

Oculus (Meta). (2022a). *Oculus Developers, Get Started Developing in Oculus Games*

+ *Apps*. https://developer.oculus.com/get-started-platform/

Oculus (Meta). (2022b). *Oculus Developers, Introduction to Oculus Browser*. https://developer.oculus.com/documentation/web/browser-intro/

Oculus (Meta). (2022c). *Oculus Developers, Native Development Overview*. https://developer.oculus.com/documentation/native/native-overview/

Oculus (Meta). (2022d). *Oculus Developers, Oculus App Development in Unity*. https://developer.oculus.com/documentation/unity/unity-overview/

Oculus (Meta). (2022e). *Oculus Developers, Unreal Engine*. https://developer.oculus.com/documentation/unreal/unreal-engine/

OpenCV. (2022a). *OpenCV: Additional photo processing algorithms*. https://docs.opencv.org/4.6.0/de/daa/group__xphoto.html

OpenCV. (2022b). *OpenCV: Biologically inspired vision models and derivated tools*. https://docs.opencv.org/4.6.0/dd/deb/group__bioinspired.html

OpenCV. (2022c). *OpenCV: Computational Photography*. https://docs.opencv.org/4.6.0/d1/d0d/group__photo.html

OpenCV. (2022d). *OpenCV: Extended Image Processing*. https://docs.opencv.org/4.6.0/df/d2d/group__ximgproc.html#ga37002c6ca80c978edb6ead5d6b39740c

OpenCV. (2022e). *OpenCV: Image Filtering*. https://docs.opencv.org/4.6.0/d4/d86/group__imgproc__filter.html

OpenCV. (2022f). *OpenCV: Image Processing*. https://docs.opencv.org/4.6.0/d7/dbd/group__imgproc.html

OpenCV. (2022g). *OpenCV: OpenCV modules*. https://docs.opencv.org/4.x/

Organización Mundial de la Salud. (2021). *Ceguera y discapacidad visual*. Nota Descriptiva N° 282. http://www.who.int/mediacentre/factsheets/fs282/es/

Puell, M. (2006). *Óptica Fisiológica: el sistema óptico del ojo y la visión binocular*. http://www.worldcat.org/title/optica-fisiologica-el-sistema-optico-del-ojo-y-la-vision-binocular/oclc/795294382&referer=brief_results

Roberto, P. (2018). *Un hospital francés utiliza la Realidad Virtual como sustituto de la anestesia*. El Chapuzas Informático. https://elchapuzasinformatico.com/2018/06/un-hospital-frances-utiliza-la-realidad-virtual-como-sustituto-a-la-anestesia/

Rodriguez, N. (2018). Identifying Accessibility Conditions for Children with Multiple Disabilities: A Virtual Reality Wheelchair Simulator. *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 370–372. https://doi.org/10.1109/ISMAR-Adjunct.2018.00107

Saraswat, P. (2021). A review paper on virtual reality. *Asian Journal of Multidimensional Research*, *10*(10), 705–711. https://doi.org/10.5958/2278-4853.2021.00872.7

Schwaber, K. (1997). SCRUM Development Process. *Business Object Design and Implementation, April 1987*, 117–134. https://doi.org/10.1007/978-1-4471-0947-1_11

ScienceDirect. (2022). *Image Normalization - an overview*. https://www.sciencedirect.com/topics/engineering/image-normalization

Spiegel, S., & Hoinkes, R. (2009). Immersive serious games for large scale multiplayer dialogue and cocreation. *Serious Games: Mechanisms and Effects*, *2005*, 469–485. https://doi.org/10.4324/9780203891650

Tobii Dynavox AB. (2022). *What is eye tracking?* https://us.tobiidynavox.com/pages/what-is-eye-tracking#:~:text=The eye tracker sends out,knows where you are looking.

Triviño Merida, D. (2014). Experiencia en realidad virtual para concienciar sobre el cuidado de la vista. In *English Language Teaching* (Vol. 39, Issue 1). https://upcommons.upc.edu/handle/2117/173959

Tschumperlé, D. (2004). *The CImg Library - C++ Template Image Processing Toolkit*. https://cimg.eu/index.html

Unity Asset Store - The Best Assets for Game Making. (2022). https://assetstore.unity.com/

Unity Technologies. (2021). *Unity Manual: Camera*. https://docs.unity3d.com/2021.1/Documentation/Manual/class-Camera.html

Unity Technologies. (2022a). GitHub - Unity-Technologies/Physical-Camera. *GitHub*. https://github.com/Unity-Technologies/Physical-Camera

Unity Technologies. (2022b). *Unity, Suported Partners Ecosystem*. https://unity.com/partners

Unity Technologies. (2022c). *Wondering what Unity is?* https://unity.com/our-company

Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE Software*, *17*(4), 19–25.

https://doi.org/10.1109/52.854064

World Health Organisation. (2019). Informe mundial sobre la visión. In *World health Organisation* (Vol. 214, Issue 14). https://apps.who.int/iris/bitstream/handle/10665/331423/9789240000346-spa.pdf

Yu Yuan. (2017). *IEEE P2048 STANDARDS PAVING THE ROAD FOR VIRTUAL REALITY AND AUGMENTED REALITY*. https://www.standardsuniversity.org/e-magazine/june-2017/ieee-p2048-standards-paving-road-virtual-reality-augmented-reality/

Zebra. (2022). *AVL Machine Vision Library for C++ and .NET - Adaptative Vision*. https://www.adaptive-vision.com/en/software/avl_lite/

Žiak, P., Holm, A., Halička, J., Mojžiš, P., & Piñero, D. P. (2017). Amblyopia treatment of adults with dichoptic training using the virtual reality oculus rift head mounted display: preliminary results. *BMC Ophthalmology*, *17*(1), 105. https://doi.org/10.1186/s12886-017-0501-8

ANEXO I: CASOS DE USO

En este anexo se desarrollan los distintos casos de uso definidos en la fase de extracción de requisitos.

Casos de uso del ejecutable simulador de ametropías:

- C. U. 1. Llamar programa sin argumentos
- C. U. 2. Llamar programa con argumentos

Casos de uso del Juego Serio:

- C. U. 3. Salir
- C. U. 4. Cambiar Escena
- C. U. 5. Cambiar parámetros
- C. U. 6. Realizar intento
- C. U. 7. Jugar escena Ver
- C. U. 8. Jugar escena Adivinar
- C. U. 9. Jugar escena Acercar
- C. U. 10. Jugar escena Distancia
- C. U. 11. Jugar escena Test
- C. U. 12. Resetear prueba
- C. U. 13. Abrir Ayuda
- C. U. 14. Cerrar Ayuda

C. U. 1	**Llamar programa sin argumentos**
Actores	Desarrollador
Descripción	El desarrollador ejecuta el programa por línea de comandos sin argumentos

Dependencias	
Precondición	
Secuencia normal	1- El desarrollador abre un terminal y accede a la carpeta donde está el programa 2- El desarrollador ejecuta el programa 3- El programa muestra por línea de texto el funcionamiento del programa
Postcondición	
Excepciones	

C. U. 2	**Llamar programa con argumentos**
Actores	Desarrollador
Descripción	El desarrollador ejecuta el programa por línea de comandos con argumentos para generar imágenes distorsionada
Dependencias	
Precondición	
Secuencia normal	1- El desarrollador abre un terminal y accede a la carpeta donde está el programa 2- El desarrollador ejecuta el programa con sus argumentos 3- El programa genera las imágenes
Postcondición	Una carpeta contiene las imágenes en el directorio donde se ejecuta el programa
Excepciones	Si los argumentos son erróneos, se indicará por pantalla y se cerrará el programa Si no existe la imagen especificada o hay un error al leerla, se indicará por pantalla y se cerrará el programa

C. U. 3	**Salir**
Actores	Jugador
Descripción	El jugador cierra el juego
Dependencias	
Precondición	
Secuencia normal	1- El jugador pulsa el botón “Salir” del menú principal 2- El juego se cierra
Postcondición	La aplicación se cierra
Excepciones	

C. U. 4	**Cambiar Escena**
Actores	Jugador
Descripción	El jugador elige una de las opciones para cambiar la escena
Dependencias	
Precondición	

Secuencia normal	1- El jugador elige una de las opciones del menú y pulsa su botón 2- La escena cambia a la elegida
Postcondición	La escena cambia a la elegida de la lista
Excepciones	

C. U. 5	**Cambiar parámetros**
Actores	Jugador
Descripción	El jugador cambia los parámetros de la escena
Dependencias	
Precondición	El jugador debe estar en una de las escenas que permite modificar parámetros
Secuencia normal	1- El jugador cambia los datos en el panel designado 2- El jugador pulsa el botón aplicar 3- Los cambios se aplican en la/s imagen/es
Postcondición	Se cambian las imágenes mostradas por las apropiadas
Excepciones	

C. U. 6	**Realizar intento**
Actores	Jugador
Descripción	El jugador pulsa el botón que cree correcto para adivinar la cavidad de la imagen
Dependencias	
Precondición	El jugador debe de estar en una escena con los botones de dirección
Secuencia normal	1- El jugador observa la imagen e intenta reconocer el agujero en el círculo 2- El jugador pulsa el botón que cree apropiado entre los direccionales y el botón de pasar 3- El programa muestra el resultado y lo añade al total
Postcondición	El programa informa del acierto o error y lo suma al total
Excepciones	

C. U. 7	**Jugar escena Ver**
Actores	Jugador
Descripción	El jugador puede observar los efectos de la ametropía en la visión, con la posibilidad de emular la intensidad con los parámetros disponibles
Dependencias	C. U. 5
Precondición	
Secuencia normal	1- El jugador cambia los parámetros y aplica los cambios 2- El jugador observa las diferencias en las imágenes conforme están cada vez más lejanas

	3- El jugador puede volver al paso 1 o estar satisfecho y cambiar de escena
Postcondición	
Excepciones	

C. U. 8	**Jugar escena Adivinar**
Actores	Jugador
Descripción	El jugador puede intentar adivinar donde está la cavidad de una imagen alterada
Dependencias	C. U. 5
Precondición	
Secuencia normal	1- El jugador observa los parámetros de la distorsión y la imagen 2- El jugador decide y pulsa la opción que le parece más razonable 3- El resultado se muestra en la pantalla lateral derecha y se genera otra imagen distorsionada 4- El jugador puede volver al paso 1 o estar satisfecho y cambiar de escena
Postcondición	
Excepciones	

C. U. 9	**Jugar escena Acercar**
Actores	Jugador
Descripción	El jugador observa los cambios sobre una imagen a la que pueden cambiar los parámetros, esta vez siendo capaces de variar la distancia del lienzo para verla en diferentes distancias, que se aplican a la distorsión
Dependencias	C. U. 5
Precondición	
Secuencia normal	1- El jugador cambia los parámetros y se aplican automaticamente los cambios 2- El jugador puede acercar y alejar la imagen para ver como la imagen distorsionada cambia con la distáncia 3- El jugador puede volver al paso 1 o estar satisfecho y cambiar de escena
Postcondición	
Excepciones	

C. U. 10	**Jugar escena Distancia**
Actores	Jugador
Descripción	El jugador puede comprobar a que distancia es capaz de reconocer los agujeros en la imagen sin distorsionar a distancia variable
Dependencias	C. U. 5
Precondición	
Secuencia normal	1- El juego puede elegir que imagen aparece en el display 2- El jugador mueve el slider para alejar/acercar la imagen hasta que deja de poder reconocer el agujero 1- El jugador puede pasarle las gafas a otro jugador, que volvería al paso 1, o cambiar de escena
Postcondición	
Excepciones	

C. U. 11	**Jugar escena Test**
Actores	Jugador
Descripción	El jugador puede comprobar si se le detecta una discapacidad visual realizando un test en el que se comprueban los resultados de los intentos establecidos
Dependencias	C. U. 6
Precondición	
Secuencia normal	1- El jugador ve una imagen colocada a cierta distancia 2- El jugador pulsa el botón que considera acertado 3- Se muestra el resultado por pantalla y se vuelve al paso 1 hasta completar los intentos establecidos 1- Se deshabilitan los botones del juego y se muestra una recomendación
Postcondición	El juego calcula un resultado según las respuestas dadas para emitir una recomendación sobre la visión del jugador
Excepciones	Si el jugador se equivoca, tiene la capacidad de resetear la prueba

C. U. 12	**Resetear prueba**
Actores	Jugador
Descripción	El jugador resetea la prueba actual
Dependencias	C. U. 10
Precondición	
Secuencia normal	2- El jugador pulsa el botón de reset 3- La prueba se resetea, los contadores vuelven a 0, los intentos vuelven a su valor original y los botones bloqueados se reestablecen

Postcondición	El juego calcula un resultado según las respuestas dadas para emitir una recomendación sobre la visión del jugador
Excepciones	Si el jugador se equivoca, tiene la capacidad de resetear la prueba

C. U. 13	**Abrir Ayuda**
Actores	Jugador
Descripción	El jugador abre la ayuda de la prueba actual
Dependencias	
Precondición	
Secuencia normal	1- El jugador pulsa el botón de Ayuda en el Menú de navegación 2- El panel de ayuda aparece frente a la escena actual
Postcondición	Aparece un panel de ayuda que explica el funcionamiento de la escena actual en la pantalla
Excepciones	

C. U. 14	**Cerrar Ayuda**
Actores	Jugador
Descripción	El jugador cierra la ventana de ayuda
Dependencias	C. U. 10
Precondición	
Secuencia normal	1- El jugador pulsa el botón de cerrar que aparece en la ayuda 2- La ventana de ayuda se cierra
Postcondición	La ventana de ayuda se cierra, dejando el camino libre para interactuar con la escena actual
Excepciones	En la escena de Inicio no hay botón de cerrar en la ventana de Ayuda, por lo que esta no se puede cerrar