ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Aplicación de un visor de realidad virtual a juegos serios para rehabilitación

AUTOR: José Zarco Torres

TUTOR (o Director en su caso): Martina Eckert

TITULACIÓN: Grado en Ingeniería de Sonido e Imagen

DEPARTAMENTO: Teoría de la Señal y Comunicaciones

V°B°

Miembros del Tribunal Calificador:

PRESIDENTE: María Luisa Martín Ruíz

VOCAL: Martina Eckert

SECRETARIO: Enrique Rendón Angulo

Fecha de lectura: 14 de Julio de 2016

Calificación:

El Secretario,

Porque ciertamente hay fin,
Y tu esperanza no será cortada.
Oye, hijo mío, y sé sabio,
Y endereza tu corazón al camino.
Proverbios 23:18-19

Agradecimientos

Este proyecto pone punto y final a cinco años de carrera, y no puedo cerrar esta etapa sin dar gracias.

Primeramente debo dar gracias a mis padres. Han sido y son para mí un ejemplo de superación y sacrificio. Han sabido retarme a ser mejor persona cada día y a esforzarme por lo que quiero.

Gracias a Sara, que me soporta cuando nadie más puede y alegra cada día de mi vida.

Gracias al Búnker, por ser mis amigos.

Gracias a todos los compañeros con los que he compartido estos años y me han ayudado tanto.

Gracias a los buenos profesores que han sido para mí una inspiración, y a los malos, que me han obligado a superarme para conseguir perderlos de vista.

Gracias a mi tutora, Martina Eckert, que siempre intenta sacar todo lo que sus alumnos pueden dar y a mis compañeros de proyecto.

Gracias a los voluntarios que dedicaron su tiempo a llevar a cabo nuestras pruebas.

Gracias a Alisha, por corregirme el abstract.

Y sobre todo, gracias a Dios. “Porque Jehová da la sabiduría, y de su boca viene el conocimiento y la inteligencia.” Proverbios 2:6.

Resumen

En este proyecto de fin de grado se pretende añadir la funcionalidad que ofrece un visor de realidad virtual a un proyecto de juegos serios orientados a la rehabilitación de personas con movilidad reducida. Este otro proyecto sobre el cual se ha trabajado tenía desarrollado un *middleware* que sirve de pasarela de comunicación entre el sensor Kinect y Blender Game Engine, el cual permite la creación de videojuegos controlados por gestos y movimientos corporales. Primeramente, se ha analizado la tecnología de realidad virtual y su evolución para después definir su situación actual y el valor que esta tecnología puede aportar en el área de los *exergames*. Después se ha analizado en profundidad el mercado de dispositivos de realidad virtual, dividiéndolo en dos familias: los que hacen uso de un *smartphone* y los que no, y se ha seleccionado el más adecuado para empezar el desarrollo del proyecto. A continuación, se ha hecho un análisis del *middleware* en el cual se debía integrar la funcionalidad propuesta y se ha realizado un diseño de la solución. Dicha solución se ha implementado mediante el desarrollo de un módulo, que se integra en el *middleware*, que hace uso del *framework* OSVR (*Open Source Virtual Reality*) para la adquisición de datos del visor y que gestiona un protocolo de comunicación para el envío de datos a Blender Game Engine. También se ha desarrollado una extensión para Blender que recibe los datos del visor y los procesa para presentar la imagen correspondiente en el visor de realidad virtual. Además, se ha planteado el diseño de la ampliación del desarrollo llevado a cabo en este proyecto para que sea soportado también cualquier visor de realidad virtual basado en *smartphone*, analizando las soluciones existentes y evaluando los resultados obtenidos. Una vez terminada la fase desarrollo, teniendo un prototipo funcional, se ha realizado una batería de pruebas con gente discapacitada siendo considerados potenciales usuarios del sistema, y otra sesión de pruebas con usuarios sin discapacidad. Se han obtenido conclusiones positivas de la información obtenida durante las pruebas para trabajar en la mejora de los puntos débiles del sistema. Finalmente se han propuesto diferentes caminos que podrían tomarse para continuar ampliando este proyecto.

Abstract

The aim of this final project is to include the functionality offered by a head-mounted display (HMD) for virtual reality into a serious games project which aims to be used by people with reduced mobility. This other project counts on a middleware that works as a communication bridge between the Kinect sensor and the Blender Game Engine, which allows the creation of videogames controlled by the body movements of the player. First, an analysis has been made of the technology related to virtual reality and its evolution with the purpose of defining the current state of the art and the value this technology can add to the field of exergames. Later, the virtual reality devices market has been analyzed and divided into two different families: those based on smartphones and the rest, and the

most appropriate device has been selected to start the development of this project. Next, an analysis has been made of the existent middleware on which the proposed functionality has to be integrated and a solution has been designed for this integration. That solution has been implemented by the development of a module, which is integrated in the middleware that uses the OSVR framework for the data acquisition from the HMD and manages a communication protocol for the data submission to Blender Game Engine. Also, a Blender add-on has been developed which receives and processes the data from the HMD to present the user view according to the head position. In addition, an extension has been developed that adds support to any head-mounted display based on the use of a smartphone, analyzing the existent solutions and evaluating their working. Once the development stage has been concluded, a battery of tests has been performed with the functional prototype. Therefore, two groups of persons have been tested, with and without disabilities. The conclusions have been positive and the tests have given enough information to improve the weak points of the system. Finally, different ways have been proposed to continue and amplify this project.

Índice de Contenidos

Lista de Acrónimos

CITSEM: Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad

SDK: *Software Development Kit*

HMD: *Head-Mounted Display*

NASA: *National Aeronautics and Space Administration*

OSVR: *Open-Source Virtual Reality*

OSC: *Open Sound Control*

UDP: *User Datagram Protocol*

JSON: JavaScript *Object Notation*

PC: *Personal Computer*

OLED: *Organic Light-Emitting Diode*

RGB: *Red, Green, Blue*

API: *Application Programming Interface*

1. Introducción

"2016 será, por fin, el año de la realidad virtual". Así lo afirma la prensa de actualidad tecnológica [1], y es que durante este año la realidad virtual ha dejado de ser una promesa para ser un hecho. Las grandes compañías de videojuegos han apostado fuertemente por una tecnología que, con total seguridad, va a revolucionar los mercados en los que estas empresas se mueven. En pocos años esta tecnología ha pasado de ser mencionada únicamente en la ciencia ficción a ser noticia con frecuencia en los medios de comunicación como una innovación que llegará al público de forma inminente. En 2016 están siendo lanzados una gran variedad de modelos de visor de realidad virtual para las principales plataformas de videojuegos del mercado. También en 2016 los mayores estudios de videojuegos del mundo están publicando los títulos que serán utilizados con estos dispositivos de realidad virtual invirtiendo en su desarrollo masivas cantidades de dinero. Pero la revolución de la realidad virtual no solo va a afectar al mundo de los videojuegos si no que pretende abrirse paso en áreas como la medicina e impulsar innovaciones relevantes en esta.

Existe un género de videojuegos en los que se invierte menos dinero por estar enfocados a un segmento de mercado reducido. Se trata de los llamados "juegos serios", que pretenden conseguir un fin práctico además de entretener, y dentro de esta categoría se encuentran los *exergames* (nombre formado a partir de la unión de *exercise* y *games*). Éstos pretenden que el jugador realice ejercicio físico alimentando su motivación mediante las mecánicas habitualmente utilizadas en los juegos, es decir, mediante puntaciones, retos, logros y demás. Se trata de un tipo de videojuego que resulta de especial utilidad para personas que están obligadas a realizar ejercicio físico por motivos de salud. Sin embargo, aunque existen videojuegos de este género para las algunas de las principales plataformas, no se ha lanzado aún ninguno para ser jugado con un visor de realidad virtual, y probablemente falten algunos años para que algún gran estudio de videojuegos publique un título de estas características.

En el Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad (CITSEM) de la Universidad Politécnica de Madrid hay en marcha un proyecto que se centra en el desarrollo de *exergames* orientados a personas con movilidad reducida. La idea principal de este proyecto es la creación de un videojuego atractivo para los jugadores, pero que les obligue a realizar ejercicio físico para progresar en él. Para ello el videojuego se controla mediante los gestos y movimientos del usuario, que son capturados con la interfaz natural Kinect. El objetivo es que personas con cualquier tipo de discapacidad puedan realizar sesiones de rehabilitación jugando a este juego y que éste provoque que el usuario realice las sesiones con más motivación y durante más tiempo. Además, se pretende que el juego se adapte a las capacidades motrices del jugador, sin que le cause sobreesfuerzo. Dicho proyecto, denominado Blexer, comenzó con el desarrollo de un *middleware* y una extensión para el motor gráfico Blender Game Engine

que permite que se desarrollen videojuegos controlados mediante los movimientos del jugador capturados por el sensor Kinect.

El presente proyecto final de grado pretende llevar a cabo la integración de la funcionalidad ofrecida por los visores de realidad virtual en el proyecto Blexer. El valor que aporta la realidad virtual a los videojuegos convencionales es el aumento de la inmersión del jugador en el entorno virtual del juego, y en el proyecto Blexer esto puede resultar también de valor, haciendo que los jugadores se metan más en el juego y olviden por completo que están haciendo ejercicios de rehabilitación. Para que esto sea posible se propone el desarrollo de un módulo que se integre en el *middleware*, que ya hay desarrollado, que lleve a cabo la comunicación necesaria entre el visor de realidad virtual y Blender Game Engine. Se pretende hacer un desarrollo genérico y ampliable para que puedan ser soportados diferentes modelos de visor de realidad virtual y de este modo mantener el futuro producto final lo más asequible para el público.

En esta memoria se explica el proceso llevado a cabo durante el desarrollo del proyecto. Primeramente, se exponen con detalle los objetivos que se esperan cumplir a lo largo del desarrollo de éste. A continuación, se expone un análisis acerca del marco tecnológico en el que se enmarca el mismo, definiendo qué es la realidad virtual y cómo funciona, la evolución de ésta tecnología hasta llegar al estado actual y su penetración en el campo de los juegos serios. Como continuación de este análisis, se describen las características de los dispositivos de realidad virtual disponibles al comienzo del proyecto y su adecuación al mismo, decidiendo el modelo que mejor se ajusta a los requisitos de desarrollo. Los siguientes apartados exponen en profundidad los detalles de la solución propuesta, comenzando por un análisis del *middleware* en el que se debe integrar la solución. Posteriormente se explica en detalle el funcionamiento de la comunicación entre el visor de realidad virtual y el motor de videojuegos, empezando por el módulo para el *middleware* y continuando por la extensión de Blender que recibe y procesa los datos del visor. Después se plantean los problemas a solucionar para la ampliación del desarrollo llevado a cabo en este proyecto que consistiría en añadir soporte a visores de realidad virtual basados en *smartphone*, y se analizan las soluciones existentes para esta funcionalidad. A continuación, se describen las pruebas que se llevaron a cabo con potenciales usuarios del sistema, una vez acabado el desarrollo, y las conclusiones obtenidas de ellas. Para terminar, se proponen diferentes caminos a seguir para continuar ampliando este proyecto de final de grado.

2. Objetivos del proyecto

El objetivo principal de este proyecto es añadir la funcionalidad que ofrece la realidad virtual a un proyecto de creación de juegos serios orientados a la rehabilitación de personas con movilidad reducida.

Actualmente, enmarcado en las labores de investigación y desarrollo llevadas a cabo en el CITSEM, se están desarrollando un conjunto de herramientas que posibilitan la creación de videojuegos adaptados a personas que posean unas habilidades motrices reducidas. Esto es posible gracias a interfaces naturales de control como el sensor Kinect, que permite controlar el videojuego mediante gestos y movimientos, sin la necesidad de utilizar mandos o controles. Para crear los videojuegos se utiliza un motor gráfico de código libre llamado Blender Game Engine, que posibilita de una forma sencilla crear entornos virtuales, animarlos y dotarlos de interactividad gracias a un sistema de bloques o ladrillos con el que se puede generar la lógica del juego incluso sin nociones de programación. Blender Game Engine también permite crear una lógica más compleja mediante *scripts* desarrollados en Python.

Para comunicar Kinect con Blender Game Engine se tiene un *middleware* ya desarrollado desde el CITSEM por parte del alumno en prácticas Ignacio Gómez-Martinho [2] . Este *middleware* hace uso de los controladores de Kinect y el SDK (*Software Development Kit*) aportado por el fabricante para la adquisición de datos del sensor y gestiona el envío de estos datos de forma ordenada al motor gráfico. Para que Blender Game Engine interprete y procese correctamente los datos, se ha desarrollado una extensión (en inglés *add-on*) instalable que genera la lógica necesaria para añadir esta funcionalidad durante el desarrollo de un videojuego.

Con el afán de poder añadir valor a este tipo de juegos y que el usuario obtenga una inmersión mayor que en los videojuegos tradicionales y pueda evadirse aún más de la sensación de estar realizando ejercicios de rehabilitación, se pretende añadir al *middleware* ya desarrollado, un módulo que permita que estos videojuegos se puedan utilizar con un visor de realidad virtual comercial. Para esto, se estudiarán los diferentes dispositivos del mercado, se integrará uno en el *middleware* y se realizarán pruebas de efectividad de la funcionalidad añadida.

3. Descripción de la solución propuesta

3.1 Definición de realidad virtual

La realidad virtual se puede definir como un sistema de computación capaz de generar un mundo o ambiente artificial en el cual el usuario del sistema tiene la impresión de estar y la habilidad de interactuar y navegar. Para este fin el sistema pretende simular las percepciones sensoriales que hacen que el usuario se sienta inmerso en ese mundo y pueda tomar dichas percepciones como reales. Con este propósito, el sistema debe generar en tiempo real estas percepciones de modo que permita al usuario interactuar con el entorno a través de los diferentes canales sensoriales (oído, vista, tacto, olor o gusto).

La realidad virtual se basa en tres principios fundamentales: la inmersión, la interacción y la imaginación [3]. La inmersión hace que el usuario perciba sensaciones solo desde el mundo virtual de modo que esté lo más aislado posible en este mundo y lo perciba como

real. La interacción a través de dispositivos de entrada posibilita la modificación del entorno virtual y la recepción de respuestas sensoriales al usuario. La forma de conseguir esto es generando respuestas inmediatas en el mundo virtual a las interacciones del usuario, de forma que el tiempo en el mundo virtual sea exactamente igual que el tiempo real. Por último, la imaginación juega el papel de la creación del mundo virtual que genera el sistema de forma que se puedan percibir mundos que no existen dejando lugar a la creación artística.

Este sistema genera un flujo de datos bidireccional. Existe un canal de datos de entrada que toma datos de dispositivos como guantes de datos, localizadores que monitorizan la posición de un objetivo, etc. Por otro lado es necesario un canal de datos de salida que generen los estímulos al usuario para que este perciba el mundo virtual, por lo cual son necesarios dispositivos de salida, ya sea salida de audio, como unos auriculares o altavoces, salida de vídeo, como un visor de realidad virtual o cabina de simulación, o de tacto como unos guantes de datos.

Durante el desarrollo de esta tecnología han surgido diferentes clases de interfaces para el usuario, principalmente en cuanto a interfaces de salida de vídeo, ya que la vista es el sentido clave para lograr la inmersión del usuario. Sin visualización del mundo virtual, la sensación de inmersión es prácticamente nula. Por esto la investigación realizada en el campo de la realidad virtual se ha centrado más en esta clase de interfaz de salida, dando lugar a dispositivos como cabinas de simulación, que son pequeñas salas donde cada pared es una pantalla con la proyección del mundo virtual haciendo que todo lo que ve el usuario a su alrededor sea el mundo virtual. Existen sistemas de mapeo por vídeo, que replican sobre una pantalla una filmación del usuario añadiéndole una representación gráfica de un objeto con el cual puedan interactuar. Este sistema se acerca más a la realidad aumentada, que aunque guarda relación con la realidad virtual, no se trata exactamente de lo mismo y no es objeto de este proyecto. No obstante, la interfaz de salida que se ha impuesto debido a sus resultados en cuanto a sensación de inmersión, ha sido lo que en inglés se denomina *head-mounted display* (HMD) y que en castellano se puede denominar casco, gafas o visor de realidad virtual. Este casco funciona colocando una pantalla de alta resolución a pocos centímetros de los ojos del usuario y ocultando el resto del campo de visión de este, de forma que solo pueda ver la pantalla. Esta pantalla ofrece a cada ojo un punto de vista adecuado del mundo virtual para generar una sensación de profundidad similar a la que se obtiene en el mundo real. El visor de realidad virtual generalmente se acompaña de un localizador que monitoriza la posición y rotación del casco para que se muestre por la pantalla un punto de vista preciso, muy acorde con la posición real del usuario. La figura 1 muestra un esquema de la colocación más habitual del localizador.

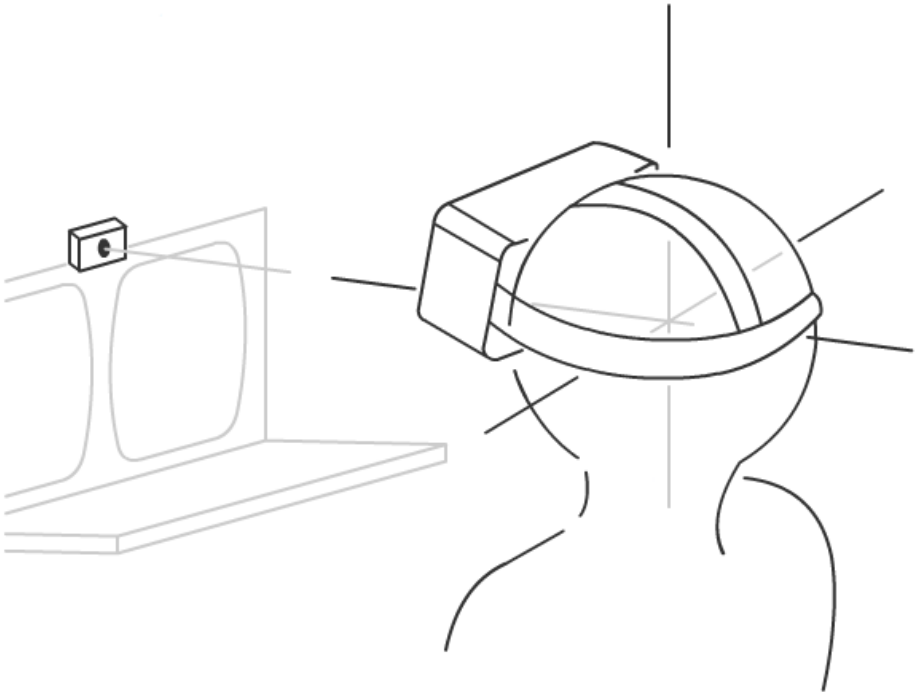

Figura 1 - Esquema de un visor de realidad virtual con un localizador de posición

Aunque el concepto de realidad virtual no es nuevo y existen cantidad de intentos por parte de muchas compañías por desarrollar sistemas de realidad virtual óptimos, no ha sido hasta la actualidad que el avance de la tecnología ha permitido desarrollar sistemas de realidad virtual que logren una sensación de inmersión óptima.

3.2 Situación actual de la realidad virtual

La realidad virtual no es una idea nueva en absoluto. Se han desarrollado prototipos de sistemas de realidad virtual inmersiva desde los años ochenta como el sistema VIVED desarrollado por la NASA (*National Aeronautics and Space Administration*) [4] orientado a la formación de futuros astronautas. Otras compañías crearon dispositivos similares, algunos llegando a ser comercializados, pero siempre con resultados no suficientemente inmersivos y sobre todo con unos precios que no ponían el sistema al alcance del consumidor medio. No será hasta 1995 cuando Nintendo lance al mercado Virtual Boy, una videoconsola que utiliza la realidad virtual con gráficos tridimensionales en rojo y negro. A pesar de la novedad que resultó, fue un fracaso comercial [5] debido a diversos motivos, que pusieron de manifiesto que no era el momento más adecuado para esta tecnología. El aspecto de estos dos dispositivos se muestra en la figura 2.

Figura 2 Izquierda: Sistema VIVED Derecha: Nintendo Virtual Boy

El mundo de los videojuegos ha sido el gran impulsor del desarrollo de entornos virtuales con gráficos tridimensionales hiperrealistas desde finales de los años 90. Este gran impulso ha provocado la aparición de potentes motores gráficos que generan en tiempo real mundos virtuales con un nivel de detalle que hace años era inimaginable. Actualmente existen una amplia variedad de motores gráficos con unas capacidades gráficas muy avanzadas. Algunos ejemplos pueden ser: Source 2, Unreal Engine, Unity 5, CryENGINE o Blender Game Engine, siendo este último el utilizado para este proyecto. La tecnología utilizada por estos sistemas en combinación con un hardware con la potencia suficiente, cuyo precio se reduce cada año a la vez que se lanzan sistemas más potentes, soluciona el problema de generar el mundo virtual en tiempo real con la calidad gráfica suficiente como para potenciar la inmersión y para reducir la sensación de mareo al utilizar los dispositivos.

Por otro lado, en los últimos años se ha rediseñado prácticamente desde cero el hardware de los sistemas de realidad virtual que hacen la función de interfaz entre el software que genera el entorno virtual y el usuario. Una pequeña empresa nacida en 2012 gracias al *crowdfunding* llamada Oculus VR, lanzó el primer prototipo de lo que finalmente ha definido los sistemas de realidad virtual actuales [6]. Su primer lanzamiento denominado Oculus DK1 y el siguiente denominado Oculus DK2 desataron una revolución en el mundo de los videojuegos gracias al gran resultado en cuanto a inmersión y a una campaña publicitaria estratégicamente diseñada para la nueva generación de aficionados a los videojuegos. En un periodo de dos años desde el lanzamiento de estos dispositivos, todas las grandes compañías tecnológicas han reaccionado al fenómeno de la realidad virtual, invirtiendo en ello y desarrollando sistemas propios que compiten entre ellos para dominar un segmento de mercado que prácticamente acaba de nacer y que pretende dominar el sector de los videojuegos en poco tiempo.

A pesar de la gran variedad de dispositivos en desarrollo por diferentes marcas, prácticamente la totalidad de ellas comparten la mayoría de las características de

funcionamiento del sistema. El elemento principal de estos dispositivos de realidad virtual es un casco o gafas compuesto por una estructura que mantiene la interfaz de salida de vídeo en una posición fija frente a los ojos del usuario. Unas cintas elásticas que pasan por detrás y por encima de la cabeza mantienen el elemento fijado a la cabeza y ante los ojos. En el interior de este elemento se encuentra la interfaz de salida de vídeo que consta de una pantalla situada a pocos centímetros de los ojos. Según el dispositivo se trata de una única pantalla para ambos ojos o una por cada ojo. En cualquier caso, al estar a una distancia tan corta de los ojos, es necesario que la densidad de píxeles de la pantalla sea elevada con el objetivo de que el usuario sea incapaz de ver los píxeles aislados si no que sólo vea la imagen que éstos conforman. Entre la pantalla y los ojos se interponen unas lentes plano convexas específicamente diseñadas para aumentar el campo de visión del usuario. Estas lentes generan una distorsión en el campo de visión que deberá ser corregida por el motor gráfico al generar la imagen de salida. Estos sistemas de realidad virtual no solo reciben un flujo de vídeo para mostrarlo al usuario, si no que también generan un flujo de datos con los datos de posición y rotación del usuario, de los cuales dependerá la imagen generada por el motor gráfico. Para obtener estos datos se obtienen lecturas de un giroscopio electrónico de tres ejes y un magnetómetro en combinación con un sistema de localización más complejo y preciso que complementa los datos de rotación y aporta los datos de posición dentro de un rango de captación. Este sistema de localización funciona mediante una cámara de infrarrojos situada frente al usuario que detecta la posición de una serie de marcadores situados en el propio casco. Dependiendo del modelo, el sistema de localización puede utilizar varias cámaras con el objetivo de mejorar la precisión y aumentar el rango de captación del sistema de modo que se pueda utilizar aprovechando incluso todo el espacio de una sala. Muchos modelos aprovechan este sistema de localización para identificar también la posición y rotación de otros elementos que se añaden al sistema como mandos controladores para usar con las manos, como los mostrados en la figura 3. Esto hace que el avatar que se pueda estar utilizando en el videojuego tenga manos que responden con precisión al movimiento de las manos del jugador en el mundo real.

Figura 3 Ejemplo de uso de controladores para las manos de la marca Oculus

Aunque son los dispositivos anteriormente descritos los que están dominando la mayor cuota del mercado, existen otro tipo de sistemas basados en el mismo principio y con la misma apariencia que utilizan el hardware incluido en los *smartphones* actuales, que poseen los elementos más importantes de los dispositivos antes descritos: giroscopios electrónicos y pantallas de tamaño adecuado y con alta densidad de píxeles. Utilizando un soporte que coloque el *smartphone* frente a los ojos como el que muestra la figura 4, que interpone entre la pantalla y los ojos unas lentes adecuadas, se puede utilizar como sistema de realidad virtual a un coste muy bajo y con unos resultados aceptables.

Figura 4 Ejemplo de HMD basado en Smartphone

3.3 *Exergames* y Realidad Virtual

En el mundo de los videojuegos existe una categoría definida como juegos serios dentro de la cual se encuentran los videojuegos orientados al ejercicio (*exergames*) que pretenden hacer uso de elementos interactivos e inmersivos para estimular la movilidad del cuerpo y potenciar la actividad física a través del juego y la competencia. El sensor Kinect de Microsoft ha jugado un importante papel en el desarrollo de este tipo de juegos ya que permite llevar a cabo un análisis cinemático gracias a técnicas de captura de movimiento. Combinando este dispositivo con el diseño de juegos interactivos amenos que promuevan la actividad física para el progreso en el videojuego, se consiguen entornos que motivan al usuario a realizar actividad física haciéndole sentir que está jugando en lugar de estar ejercitándose. Se ha demostrado que aplicaciones interactivas que motivan el ejercicio suponen una experiencia positiva para realizar una rutina diaria son aceptados por un alto porcentaje de aficionados a la actividad física, haciéndola más amena [7]. Es por tanto evidente el valor que puede aportar este dispositivo y este tipo de desarrollos a la rehabilitación de personas con movilidad reducida que deben realizar ejercicios del tipo que estas aplicaciones potencian.

Por otro lado, la realidad virtual es una tecnología, que aunque no es nueva, está poniéndose ahora a disposición del gran público. Esta tecnología aporta un valor diferencial a todos los videojuegos en general que es la sensación de inmersión que

producen en el jugador. Esto consigue que en un juego adecuadamente diseñado el jugador experimente una sensación de evasión del mundo real e inmersión en el mundo virtual. Concretamente en el proyecto de *exergames* sobre el que se va a trabajar, se puede explotar además una característica propia de estos visores de realidad que es el seguimiento tan preciso que hacen de la posición de la cabeza. Con este método se puede complementar un punto débil de Kinect, que es la detección de los movimientos de la cabeza, y haciendo que estos movimientos puedan formar parte del control del juego.

Existen menos referencias de estudios realizados acerca del valor que la realidad virtual aporta a los *exergames*, no obstante es un campo de estudio que crece y que ya ha demostrado resultados positivos en la rehabilitación de pacientes afectados por derrame cerebral [8] y en juegos orientados a la motivación de la actividad física [9].

Lo novedoso del desarrollo que se pretende llevar a cabo en este proyecto es la combinación de herramientas a utilizar. Existen proyectos de *exergames* que utilizan el sensor Kinect para la rehabilitación de personas pero no se han encontrado referencias de sistemas que combinen Kinect con un sistema de realidad virtual para desarrollar este tipo de juegos en el motor gráfico de código abierto Blender Game Engine.

3.4 Análisis de los dispositivos actuales

El campo de los dispositivos de realidad virtual ha experimentado un crecimiento muy fuerte en los últimos dos años, por lo que el siguiente análisis trata aquellos dispositivos relevantes en el mercado en septiembre del año 2015, fecha en la que se realiza este análisis para seleccionar el modelo a utilizar en el proyecto. Los principales dispositivos de realidad virtual son los siguientes:

Oculus Rift

Oculus Rift nace de la mano de un joven ingeniero de 21 años que tras desarrollar varios prototipos consigue más de dos millones de dólares de financiación en una campaña de *crowdfunding*. Tras irrumpir en el mercado y crear una nueva tendencia en el sector de los videojuegos, Facebook adquiere la compañía por dos mil millones de dólares. Desde entonces, dos versiones destinadas al desarrollo se han puesto a la venta. La actual versión denominada Oculus Rift DK2 lleva disponible desde agosto de 2014 y pronto será relevada por la esperada versión comercial que ya ha sido presentada, pero que no se pondrá a la venta hasta el primer cuatrimestre del 2016. Los desarrollos creados con las versiones para desarrolladores serán compatibles con la versión comercial, y ésta versión se espera que tenga soporte para todos los videojuegos de la videoconsola Xbox One, así como juegos para PC hechos con plataformas como Unity, Unreal y Cry Engine así como con todos los desarrollos independientes que ya se están haciendo gracias a la gran comunidad que se ha creado en torno a este producto y a las herramientas de desarrollo que el fabricante ofrece.

Figura 5 Oculus Rift DK2

El modelo DK2 posee una pantalla OLED (*Organic Light-Emitting Diode*) de baja persistencia con una resolución de 1920x1080 píxeles bajo una tasa de refresco de 75 hercios. Además, posee un sistema de seguimiento de la posición del dispositivo para así enviar datos de la rotación y traslación en cada eje de la cabeza.

Samsung Gear VR

El modelo Samsung Gear VR surge de una colaboración entre Samsung y Oculus y aunque la electrónica es completamente distinta, el resultado es parecido al que se consigue con el modelo anteriormente mencionado, Oculus Rift. Se trata de un dispositivo que hace uso de la pantalla de los *smartphones* de gama más alta de Samsung, concretamente el Galaxy Note 4 y el Galaxy S6 y S6 Edge. Estos últimos dos modelos poseen una pantalla de 2560x1400 pixeles que aportan una gran densidad de píxeles. La mayor parte de los juegos utilizados por este dispositivo requerirán de un mando de control vendido aparte que facilita el control del videojuego.

Figura 6 Samsung Gear VR

HTC Vive

La compañía HTC en colaboración con la empresa Valve (propietaria de la popular plataforma Steam) ha desarrollado un dispositivo llamado HTC Vive. Este dispositivo está pensado para ir aún más lejos que el resto en cuanto a la percepción de la realidad virtual. Vive ofrece un sistema de seguimiento basado en láser que no solo obtiene la rotación de la cabeza sino también la posición de ésta en el espacio dentro de un rango muy amplio (una habitación). Además existen dos mandos para las dos manos cuya posición también es analizada y utilizada para controlar la posición de las manos del avatar o de objetos en el videojuego. La pantalla utilizada por este dispositivo tiene una resolución de 2400x1080 pixeles, lo que unido al gran sistema de seguimiento de la posición, produce una experiencia de realidad virtual muy lograda, sin embargo este dispositivo está pensado para jugadores que deseen una experiencia de realidad virtual máxima, ya que se requiere la instalación de dispositivos de seguimiento en la habitación donde se vaya a jugar, lo cual puede resultar complejo para jugadores esporádicos o no tan aficionados.

Actualmente es un proyecto en desarrollo y solo se envían prototipos a desarrolladores que la propia compañía selecciona de entre todas las solicitudes que reciben.

Figura 7 HTC Vive

Sony Project Morpheus

La compañía Sony, en vista de la evolución que esta tomando el mundo de los videojuegos, también ha desarrollado un dispositivo de realidad virtual para su videoconsola PlayStation al cual de momento llaman Project Morpheus. A día de hoy aún no se conocen demasiados detalles de las características del visor, pero sí que estará muy

orientado al juego multijugador online. Actualmente no tiene fecha de lanzamiento ni se distribuyen prototipos para desarrollo.

Figura 8 Sony Project Morpheus

Microsoft HoloLens

Por su parte, la compañía Microsoft ha entrado en este mundo dando un giro al concepto de visor de realidad virtual para hacer un visor de realidad aumentada que ha denominado HoloLens. Se trata de un dispositivo que permite ver hologramas añadidos al entorno en el que se encuentra el jugador, pudiendo incluso interactuar con dichos hologramas. De momento es solo un concepto y han sido desarrollado prototipos funcionales, pero la salida al mercado no tiene fecha determinada.

Figura 9 Microsoft Hololens

Google Cardboard

En la convención I/O 2014 Google presentó un visor de realidad virtual de bajo coste, Carboard. Este consiste en un cartón adecuadamente plegado para que tenga la forma de un visor de realidad virtual y soporte unas lentes ante los ojos del jugador, y un *smartphone* delante de las mismas. Haciendo uso de la pantalla, los giroscopios y sensores

del propio *smartphone,* recrea mediante las aplicaciones para Android preparadas para ello, la realidad virtual mejor conseguida al precio más bajo.

Figura 10 Google Cardboard

Avegant Glyph

Avegant Glyph es un modelo de visor multimedia ligero y mucho más pequeño que el resto de modelos, sin cables y con auriculares de alta fidelidad incorporados. Se trata de un elemento concebido más para contenido multimedia que para videojuegos, dado que no posee sensores de rotación y por tanto no ofrece sensación de realidad virtual, solo una forma distinta de poder visualizar contenidos audiovisuales, tanto en 2D como en 3D.

Figura 11 Avegant Glyph

Razer OSVR

En los últimos años muchas tecnologías han avanzado gracias al desarrollo de proyectos *open source* y en el ámbito de la realidad virtual existe también una alternativa de este tipo. El proyecto OSVR (*Open Source Virtual Reality*) nace con la idea de unificar las

interfaces de entrada y salida de los videojuegos. Es un proyecto modular que avanza por partes y que por el momento está en un estado de desarrollo muy parecido a modelos como Oculus Rift y HTC Vive. Al ser un proyecto *open source* no solo hardware, si no también software, cuenta con total flexibilidad y compatibilidad de desarrollo, incluso con los kits de desarrollo de otros fabricantes.

Figura 12 Razer OSVR HDK

Zeiss VR One

La empresa del ámbito de la óptica Zeiss por su parte ha lanzado un modelo con características similares al modelo Cardboard de Google pero con un montaje y acabado mucho más cuidado. Para funcionar precisa también de un *smartphone* ya que utiliza la pantalla de este.

Figura 13 Zeiss VR One

FOVE VR

Un modelo denominado Fove VR va un paso más allá en el desarrollo de visores de realidad virtual incluyendo un sensor que analiza la posición de la pupila de cada ojo del usuario, permitiendo usar esta información para interactuar con el videojuego. Se trata de un proyecto aún en desarrollo y no se conoce fecha de lanzamiento.

Figura 14 Fove VR

Características técnicas

En la tabla 1 se detallan las características técnicas de aquellos modelos que se pueden considerar adecuados para el proyecto con el objetivo de comparar cuál es mejor desde el punto de vista técnico.

Tabla 1 Comparativa de las características técnicas de los modelos de visor de realidad virtual

	Oculus	**HTC Vive**	**OSVR**	**Samsung Gear VR**	**Google Cardboard**
Precio	350 $	-	299 $	199 $	13,99 €
Resolución (en cada ojo)	960x1080	1080x1200	960x1080	1280x1440	Depende del *smartphone* utilizado
Latencia	< 20 ms	< 20 ms		< 20 ms	Depende del *smartphone* utilizado
Positional Tracking	Sí	Sí	Sí	No	No
Necesita *Smartphone*	No	No	No	Sí	Sí
Plataformas Compatibles	PC (Windows, OS X, Linux) y en 2016 Xbox	PC (Windows, OS X, Linux)	PC (Windows, OS X, Linux)	Android	Android
SDK propia	Sí	Sí	Sí	Sí	No
Modelo disponible	Development Kit 2	Kit de desarrollo para desarrolladores seleccionados	Hacker Development Kit a partir del 23 de octubre de 2015 en preventa	Ya disponible	Ya disponible

Adecuación al proyecto

En la comparativa llevada a cabo en la tabla 1 se puede observar como los diferentes modelos de visor de realidad virtual se encuentran posicionados de forma diferente en cuanto al objetivo que persiguen. Se observan dos grandes familias de visores de realidad virtual. La primera familia está formada por los que aprovechan como pantalla la de un *smartphone* que se le acopla, haciendo también uso de los giroscopios y sensores de este, reduciendo enormemente así su coste pero limitando su foco a los videojuegos y aplicaciones para móviles. Por otro lado, en la segunda familia se encuentran los que incluyen su propia pantalla y sensores, enfocados a plataformas de videojuegos más potentes como PC y videoconsolas.

Para poder elegir el dispositivo más adecuado es necesario analizar primero las limitaciones que se tienen dada la situación del proyecto en el cual se trata de integrar este concepto.

Primeramente, la plataforma principal sobre la que se basa todo este proyecto es Blender Game Engine. Este motor de juego es una potente herramienta cuyo objetivo es el desarrollo de videojuegos que compila aplicaciones ejecutables para Windows, Linux y OS X. Complementario a esta plataforma, se tiene un *middleware*, desarrollado en otro proyecto final de grado [10] dentro del marco de este proyecto, que lleva a cabo la adquisición de datos desde el sensor Kinect de Microsoft, y los manda al motor Blender Game Engine para controlar la interacción del videojuego.

En primer lugar es necesario seleccionar una de entre las dos familias de visores anteriormente mencionadas. Dado que Blender, a día de hoy, no compila para dispositivos móviles si no sólo para PC, lo más lógico es decantarse por aquellos dispositivos orientados a PC y videoconsolas. El inconveniente de estos dispositivos es su elevado coste frente a los de la primera familia, sin embargo presentan una gran mejora en cuanto a la experiencia inmersiva de la realidad virtual con respecto a aquellos que utilizan *smartphones*. Una posibilidad para poder realizar el desarrollo objeto de este proyecto sobre un dispositivo de la primera familia sería desarrollar un sistema de comunicación entre el *smartphone* y Blender Game Engine de modo que este enviara la imagen al *smartphone* mediante *streaming* con una tasa de fotogramas suficiente como para que la visualización sea fluida y que a su vez el *smartphone* envíe datos a Blender sobre la posición de la cabeza del jugador para poder generar las imágenes correctas. Para llevar a cabo este desarrollo sería necesario enfrentar dos problemas: por un lado el control de la cámara en Blender Game Engine con los sensores de orientación del *smartphone* y por otro la implementación de un sistema de *streaming* de muy baja latencia para un flujo de vídeo de alta calidad y resolución, a una tasa de fotogramas por segundo alta. Es un requisito deseable que el sistema pueda ser utilizado tanto con dispositivos de la segunda familia por aquellos que busquen una experiencia más realista a pesar del precio, como

con dispositivos de la primera para usuarios que prefieran poder acceder al sistema de una forma más económica.

Por tanto, se va a optar por desarrollar, en una primera fase, un sistema de comunicación genérico entre el *middleware* y Blender Game Engine que soporte un modelo de visor de realidad virtual comercial concreto, ya que este tipo de visor no tiene problemas para presentar la imagen generada por Blender, dejando abierta la posibilidad de añadir soporte a otros modelos de esta misma familia. En una segunda fase se va a plantear la ampliación del sistema para que soporte modelos de visor de realidad virtual basados en *smartphone*. Con el desarrollo realizado en la primera fase se habrá resuelto el problema del control de la cámara en Blender Game Engine para un visor de la segunda familia y solo hará falta adaptarlo para poder usarlo en un visor de la primera familia, basado en *smartphone*. También será necesario en esta segunda etapa enfrentar el problema de la transmisión bidireccional de datos entre el PC y el *smartphone*. Esta segunda fase se llevará a cabo en el caso de que entre dentro del alcance en tiempo de este proyecto.

Por tanto la integración en Blender Game Engine de un dispositivo de realidad virtual se llevará a cabo, en primera instancia, con un modelo de la segunda familia. De esta forma sólo queda decidir qué modelo de la segunda familia es el más adecuado para el proyecto. En primera instancia se deben descartar aquellos con los que ni siquiera se puede hacer la integración debido a que el fabricante no proporciona herramientas de desarrollo o no vende el dispositivo a desarrolladores independientes, con lo cual Sony Morpheus, Microsoft HoloLens y FOVE VR quedan descartados. Se debe decidir entre HTC Vive, Razer VR y Oculus Rift.

HTC Vive promete la mejor experiencia de realidad virtual de entre los tres, debido a sensores externos como mandos para las manos o sensores de posición. Sin embargo no se debe olvidar que ya se dispone de una interfaz natural para controlar el videojuego con el cuerpo que es Kinect, con lo que en principio esto no añadiría valor a la experiencia de juego. El principal inconveniente de este modelo es su estado de desarrollo, dado que solo se envían prototipos a desarrolladores seleccionados por el fabricante, y tras haber solicitado una unidad, la solicitud no ha sido contestada, con lo cual es imposible obtener una unidad para este proyecto.

Razer OSVR es un candidato muy fuerte a ser elegido dada su naturaleza. Se trata de un diseño *open source*, lo cual hace que todo el software y el hardware sea público, y sea posible adaptarlo, o modificarlo según la necesidad. Además, al ser *open source*, cualquier fabricante puede comercializarlo, lo que hará que en un futuro, se lancen modelos con un coste más reducido. El inconveniente que presenta este modelo es que solo un fabricante comercializa un kit de desarrollo, pero ha retrasado la fecha de preventa tres veces, lo cual hace imposible obtener una unidad en el corto plazo.

Oculus Rift es el primer fabricante de este tipo de dispositivo. Ya ha comercializado dos versiones para desarrolladores y el primer cuatrimestre de 2016 pone a la venta la versión comercial, siendo la primera empresa que lo haga. Al haber sido el primer dispositivo, ha copado el mercado, asentando el estándar para los modelos que han venido después. Esto ha hecho que hayan sido el resto de compañías las que se han adaptado a los desarrollos llevados a cabo para Oculus; por ejemplo, aquellos videojuegos desarrollados para Oculus son muy fácilmente adaptables a OSVR. Al haber sido los primeros, también cuentan una comunidad de desarrolladores mucho mayor que dan soporte y resuelven dudas en foros. En cuanto a la disponibilidad, Oculus Rift DK2 puede ser adquirido a través de su web oficial por cualquier particular o empresa.

De esta forma se concluye que el modelo más adecuado en la actualidad para desarrollar este proyecto es Oculus Rift DK2 por los siguientes motivos:

- Se ajusta a los requerimientos del proyecto.
- Es el único del cual se puede obtener un kit de desarrollo en el corto plazo.
- La cuota de mercado que posee hará que sea líder en su segmento, provocando que surjan en el futuro modelos compatibles de otros fabricantes a un coste más reducido.
- En una prueba en un *stand* de una feria de videojuegos se ha comprobado personalmente que el dispositivo sería adecuado en cuanto a peso y ergonomía para personas con movilidad reducida.

Como se ha comentado anteriormente, aunque la integración en el *middleware* se realizará para Oculus Rift DK2, el desarrollo de ésta se llevará a cabo pensando en que sea lo más genérico posible con el objetivo de poder añadir en un futuro a esta integración modelos diferentes de visor de realidad virtual.

4. Descripción de la solución propuesta

Este proyecto nace con la idea de complementar un proyecto más grande que pretende llevar a cabo el desarrollo de un *middleware* que interactúa con el motor gráfico Blender Game Engine para añadirle la posibilidad de controlar los juegos creados con dicho motor mediante hardware de interacción natural, más concretamente el sensor Kinect, desarrollado por Microsoft.

Este *middleware* tiene como objetivo servir como herramienta para el desarrollo de una serie de juegos serios orientados a personas con movilidad reducida que le permita hacer ejercicios beneficiosos para su recuperación física de una forma interactiva y entretenida, aplicando mecánicas de juego a los ejercicios para que el usuario no tenga la sensación de estar haciendo ejercicios, si no de estar jugando.

El objetivo de este proyecto es añadir la posibilidad de jugar a dichos juegos a través de un sistema de realidad virtual, lo cual les puede aportar valor dotándolos de la sensación de inmersión que la realidad virtual produce. Esto ayuda a que el usuario se olvide de que está realizando ejercicios y se divierta jugando al juego que ha de ser visualmente atractivo para potenciar esta función.

4.1 *Middleware* para Kinect

Se trata de un software que sirve de puente para poder usar el sensor Kinect en un videojuego creado con Blender Game Engine [11]. Este software se encarga de la adquisición de los datos de la Kinect en sincronía con el envío de los mismos al motor gráfico para que este los procese y controle los aspectos del videojuego que el desarrollador haya designado mediante la lógica del mismo.

Esta aplicación hace uso del kit de desarrollo de aplicaciones para Kinect para la adquisición de datos del sensor, con lo cual este proceso se lleva a cabo de forma nativa mediante los controladores del dispositivo.

Por otro lado, la comunicación entre el *middleware* y Blender Game Engine se lleva a cabo mediante el protocolo OSC (*Open Sound Control*). Este protocolo, habitualmente utilizado para la comunicación entre instrumentos musicales, ordenadores y otros dispositivos multimedia [12], se utiliza para encapsular la información de la posición de cada hueso del esqueleto detectado por Kinect en un paquete que a su vez es encapsulado en un datagrama UDP que es el protocolo utilizado para el transporte.

Además de la aplicación ejecutable sobre Windows es necesario una extensión en Blender Game Engine con la lógica necesaria para recibir y procesar los datos que le envía el *middleware*. Esta extensión, desarrollada en Python permite, mediante unos controles situados en el modo objeto del editor de Blender, añadir a la escena un esqueleto que contiene la lógica necesaria para recibir los datagramas y reaccionar ante los datos contenidos en ellos.

4.1.1 C# y .NET

El software desarrollado en el CITSEM en el que se integra la solución propuesta está desarrollado en C#. Este lenguaje fue creado por Microsoft en un intento de obtener un nuevo lenguaje de programación con el potencial de control a bajo nivel de C/C++ y la agilidad de desarrollo de Visual Basic. C# retiene la mayor parte de la sintaxis de C/C++ pero no es compatible con ellos ya que al haber sido desarrollado desde cero, elimina todas las características que hacían que fuera difícil trabajar con C/C++ como podían ser los punteros o la gestión de memoria. C# está creado con el objetivo de aprovechar en profundidad el entorno .NET sobre el cual también se apoya el desarrollo de este

proyecto. El entorno .NET Framework es un conjunto de cuatro elementos: el entorno común de ejecución, un conjunto de bibliotecas de clase, un grupo de lenguajes de programación y el entorno ASP.NET [13]. Este *framework* fue desarrollado con tres objetivos: primero, lograr aplicaciones más estables y seguras para Windows; en segundo lugar, facilitar el desarrollo de aplicaciones y servicios web que funcionen tanto en plataformas tradicionales como en dispositivos móviles; y por último, aportar un solo grupo de bibliotecas que permitieran el trabajo con varios lenguajes de programación diferentes. El *middleware* está desarrollado sobre la versión 4.5 del *framework* .NET.

La elección de este entorno limita en cierto modo el diseño de la solución a un lenguaje compatible con el entorno en el que se debe integrar. La razón de un desarrollo sobre este *framework* es la facilidad que aporta el SDK de Kinect para desarrollar sobre él la aplicación diseñada, permitiendo una sencilla y rápida comunicación entre este software y el hardware Kinect, aumentando la seguridad de la aplicación además de hacerlo estable y robusto.

4.1.2 Kinect y Kinect SDK

La tecnología detrás del sensor Kinect fue desarrollada por Microsoft y en 2009 fue anunciado el lanzamiento del primer dispositivo Kinect como periférico de la videoconsola Xbox 360. Este permitía controlar los juegos sin utilizar mando, situando al usuario ante el sensor. Para ello utiliza un sensor de infrarrojos con un emisor, un receptor y una cámara RGB. También posee un arreglo de micrófonos y un sensor que mide la inclinación horizontal del dispositivo [14]. Estos componentes están dispuestos como se muestra en la figura 15. Todo esto lo utiliza para generar una representación del usuario que se sitúe frente al sensor mediante un esqueleto, lo cual permite desde controlar cada una de las extremidades de un avatar hasta controlar cualquier juego con sistemas de control que adapte el esqueleto obtenido por el sensor.

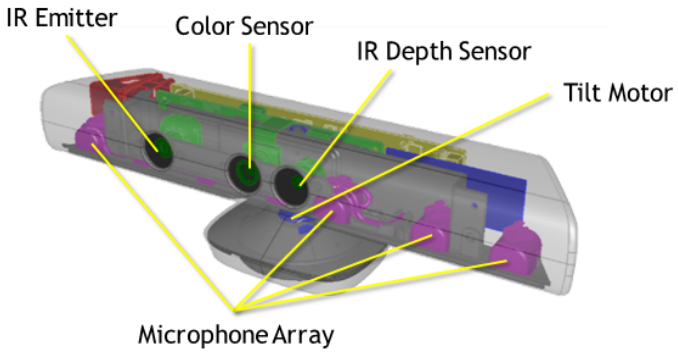

Figura 15 Componentes del sensor Kinect

En 2011 Microsoft decidió permitir el uso de este dispositivo para el desarrollo de aplicaciones nativas en su sistema operativo Windows 7, abriendo un controlador que permitía utilizar las funciones de reconocimiento de gestos, comandos de voz, objetos e imágenes. Con el objetivo de potenciar el desarrollo de este tipo de aplicaciones por

desarrolladores independientes se publicó también una serie de librerías en forma de kit de desarrollo de aplicaciones que facilitaba el uso de las funciones más habituales del sensor. Junto con este kit se lanzó también una versión del dispositivo orientada específicamente a ser utilizada con aplicaciones desarrolladas con este kit para ordenadores y se denominó Kinect for Windows.

El *middleware* hace uso de este kit de desarrollo para recibir los datos generados por el sensor y enviarlos a Blender Game Engine. La comunicación entre el *middleware* y el sensor sucede de forma transparente de cara a este proyecto, no obstante, es importante destacar las posibilidades que el sensor ofrece y se han usado en el *middleware* de forma que se pueda analizar la comunicación entre el *middleware* y Blender Game Engine. El sensor permite detectar hasta dos usuarios diferentes, diferenciando un esqueleto para cada uno de los usuarios, tal y como muestra la figura 16. En cada esqueleto, es posible obtener los datos de posición y rotación en cada hueso de forma que los datos obtenidos se pueden entregar en forma de vector tridimensional, con tres coordenadas, para conocer solamente la orientación del hueso, o en forma de cuaternión[1] con cuatro coordenadas para conocer además la rotación en su propio eje.

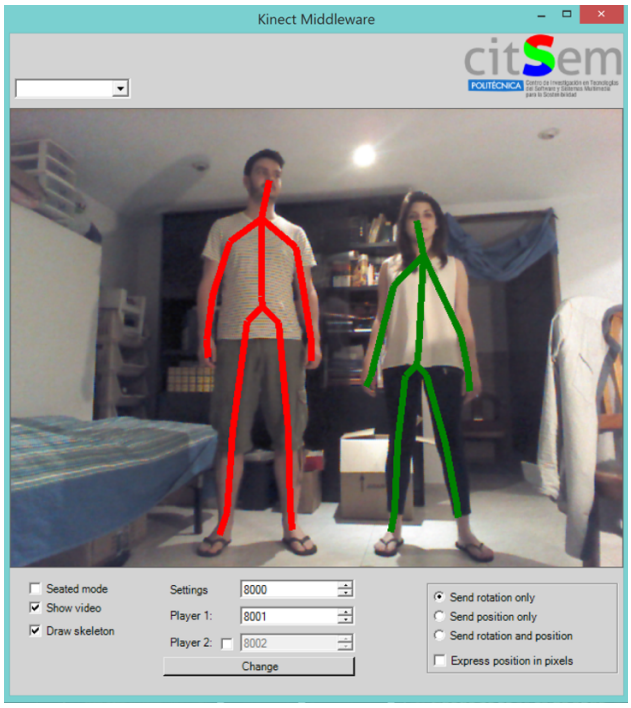

Figura 16 Interfaz gráfica del middleware durante la detección de dos jugadores

[1] Los cuaterniones son una extensión de los números reales similar a la de los números complejos. Mientras que los números complejos son una extensión de los reales por la adición de la unidad imaginaria i tal que $i^2 = -1$, los cuaterniones son una extensión generada de manera análoga añadiendo las unidades imaginarias: i, j, y k a los números reales tal que $i^2 = j^2 = k^2 = ijk = -1$. Los cuaterniones son frecuentemente usados en la generación de gráficos por ordenador para representar rotaciones. Puede encontrar más información acerca de los cuaterniones en [15].

Dadas las posibilidades del kit de desarrollo, el *middleware* se ha desarrollado de forma que es posible elegir el tipo de esqueleto que se desea que Kinect escanee, de posición o de posición y rotación. Además es posible mostrar la señal de vídeo que obtiene el sensor superponiendo el esqueleto detectado, pudiendo activar y desactivar estas funciones para ahorrar carga computacional mientras se está ejecutando algún juego. También es posible seleccionar los puertos de la comunicación que se lleva a cabo entre el *middleware* y Blender.

4.2 OSVR

Uno de los requisitos de diseño del proyecto es la posibilidad de que el producto desarrollado resulte asequible al público, de forma que puedan acceder a esta solución el mayor número de personas. Con este objetivo en mente, la solución se desarrolla de la forma más modular posible para que no sea una solución cerrada compatible con un único modelo de visor de realidad virtual, si no que permita añadir compatibilidad con diferentes modelos actuales y futuros realizando el menor número de cambios posible en el desarrollo y así poder dar soporte al mayor número de dispositivos.

OSVR es una iniciativa de código abierto que pretende servir como herramienta multiusos en el ámbito de la realidad virtual. Este *framework* pretende solventar los principales problemas e incompatibilidades que existen en relación a la realidad virtual, sirviendo de intermediario entre los controladores de los diferentes dispositivos de realidad virtual (visores, guantes, mandos, localizadores para las manos etc.) y los motores gráficos finales [16]. Se compone de un software servidor que administra las peticiones de datos a los dispositivos y la adquisición de los mismos así como la entrega de estos datos mediante una serie de librerías de código ya desarrolladas. La figura 17 presenta un esquema de todas las herramientas contenidas en el *framework*. Este *framework* hace uso por tanto de los propios controladores de los dispositivos, accediendo a ellos mediante el kit de desarrollo que cada fabricante de cada dispositivo haya distribuido. Esto hace que el acceso a los datos de los dispositivos se realice de forma nativa y eficiente además de centralizar el acceso a estos, pudiendo administrar diferentes dispositivos simultáneamente desde este mismo servidor. La compatibilidad con nuevos modelos de dispositivos de realidad virtual se lleva a cabo añadiendo complementos que habilitan al servidor para adquirir datos de estos nuevos dispositivos. La gran ventaja de esto es que haciendo llamadas genéricas al servidor como por ejemplo, pedir los datos de rotación de un HMD se hará con exactamente la misma llamada independientemente del modelo de HMD utilizado. Esto permite desarrollar aplicaciones con compatibilidad para múltiples modelos de dispositivos de realidad virtual.

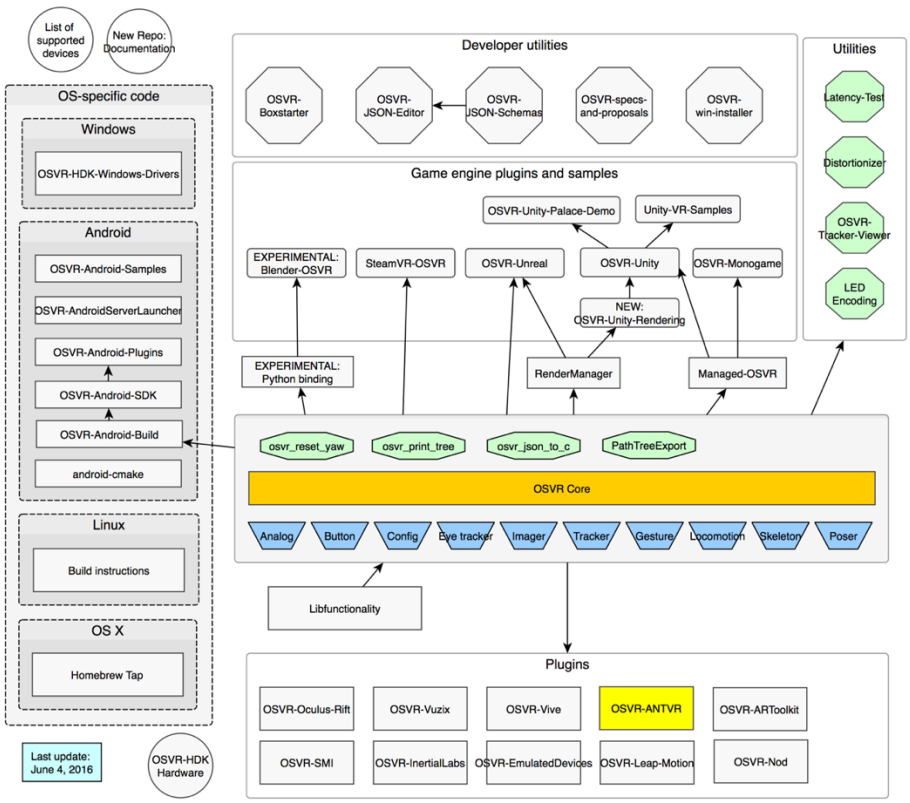

Figura 17 Directorio de proyectos de OSVR [16]

Este *framework* está desarrollado en C++, sin embargo, existe también un *wrapper* para que se puedan desarrollar aplicaciones cliente dentro de un entorno .NET tanto en las versiones 2.0 como 4.5. Esto permite integrar en el *middleware* para Kinect la aplicación objeto de este proyecto, de modo que todo funcione en una sola aplicación y sea lo más cómoda y sencilla posible.

Este software y todo su código es distribuido gratuitamente bajo una licencia Apache versión 2.0. Esta licencia permite reproducir o distribuir copias de este trabajo o una derivación de este a través de cualquier medio, con o sin modificaciones, en forma de código fuente o de ejecutable compilado, siempre y cuando se cumplan las condiciones de notificación de uso de código bajo licencia Apache [17]. Esto implica que es necesario añadir en el directorio principal del paquete de software distribuido una copia de la licencia y un fichero de texto *NOTICE* que incluya todos los avisos a los que obligue el software contenido en el paquete distribuido.

Por consiguiente, y siempre cumpliendo con las condiciones de licencia necesarias, se utilizará el *framework* para el desarrollo del presente proyecto.

4.2.1 Ventajas de OSVR frente a Oculus SDK

Como se ha explicado en el apartado anterior, OSVR hace uso del kit de desarrollo aportado por el fabricante de cada dispositivo. En este caso el dispositivo usado para el

desarrollo es el Oculus Rift DK2, de modo que una posibilidad sería utilizar directamente el kit de desarrollo que ofrece el fabricante. Sin embargo, utilizando OSVR obtenemos varias ventajas. Primeramente, el desarrollo se simplifica, dado que se utiliza el *wrapper* para .NET incluido en OSVR, que simplifica el proceso de adquisición de datos de posición y rotación del visor de realidad virtual, reduciendo el tiempo de desarrollo. Por otro lado, la razón más importante para utilizar este entorno es la compatibilidad. Este entorno permite que, con un único desarrollo, se utilicen diferentes modelos de visor de realidad virtual. Actualmente la compatibilidad se limita a unos pocos modelos comerciales de coste elevado. No obstante, la oferta comercial va aumentando y la facilidad para añadir modelos compatibles al *framework* es un punto a su favor.

Aparte del proyecto de software de código libre, OSVR cuenta con un proyecto de desarrollo hardware de un visor de realidad virtual de hardware libre. Actualmente la compañía Razer ha comenzado a fabricar el primer modelo de este diseño denominándolo Hacker Development Kit. Aún no se trata de un producto definitivo sino un kit para que los desarrolladores lo usen para hacer contenidos en realidad virtual y seguir mejorando la tecnología. Al tratarse de un dispositivo de hardware libre permite que en un futuro a medio plazo, otras marcas lo fabriquen compitiendo en precio y ofreciendo un visor de realidad virtual que seguramente llegue a ser muy asequible. La figura 18 muestra el despiece del visor mostrado por el fabricante. Por supuesto el *framework* servidor soporta este modelo de visor de realidad virtual de la misma forma que los otros modelos soportados.

Figura 18 Componentes del visor OSVR HDK1

Por todos estos motivos, el diseño de la aplicación objeto de este proyecto hace uso del *framework* OSVR para cumplir con los requisitos solicitados para esta funcionalidad. De

esta forma, se podrá añadir la funcionalidad de jugar a los juegos en entornos de realidad virtual sin estar asociado a un único modelo de visor.

4.3 Comunicación entre el HMD y Blender

El gran reto que presenta este proyecto, debido a que no se han encontrado referencias de nada parecido ya desarrollado aunque sí en proceso de desarrollo [18], es el diseño y la implementación de la comunicación entre OSVR y Blender Game Engine. Se parte de la base de que OSVR, desde el *middleware*, se encarga de la adquisición directa de los datos de posición del sistema de seguimiento del visor y los entrega en la forma requerida, ya sea como cuaterniones y vectores, o sólo como cuaterniones, independientemente del modelo de visor utilizado y con unos valores de latencia lo suficientemente bajos como para considerarlos imperceptibles por el usuario, este es por tanto un proceso transparente de cara al diseño de la comunicación del envío de datos a Blender Game Engine. Hay que centrarse entonces en el diseño de la comunicación entre el *middleware* y Blender Game Engine.

Es necesario tener en cuenta una serie de restricciones a la hora de definir el protocolo de comunicación de manera que este se realice de forma óptima. Primeramente, hay que tener en cuenta que para que un videojuego se pueda ejecutar de forma fluida en un sistema de realidad virtual, este tiene que poder generar los fotogramas por segundo suficientes como para que la imagen se perciba fluida, sin parpadeos o *flickering* y con un retardo muy reducido respecto al movimiento del jugador, con el objetivo de que la experiencia de juego sea inmersiva y no produzca mareos ni fatiga visual. Los fabricantes de los dispositivos anteriormente analizados han determinado la tasa mínima de fotogramas por segundo para que la experiencia de juego sea aceptable en 60 fotogramas por segundo. Esto supone que el periodo entre un fotograma y el siguiente es de aproximadamente 16,7 ms en el cual debe ocurrir: que OSVR realice la adquisición de los datos de posición del visor y se los pase al *middleware*; que el *middleware* envíe estos datos al motor gráfico; que el motor gráfico reciba los datos y que genere la imagen a presentar en el visor. Además, es imprescindible que la comunicación esté sincronizada, es decir, que los datos de posición correspondan al mismo instante de tiempo al que corresponde el fotograma presentado.

Partiendo de esta serie de premisas se decide que para que el modelo de comunicación entre el *middleware* y Blender Game Engine no sea muy diferente de cómo ya funciona con la Kinect (que funciona correctamente, con una latencia baja y para una cantidad de datos muy superior a la que se pretende mandar en este proyecto), se utilizará el mismo tipo de datagramas UDP (*User Datagram Protocol*) para transportar los datos, que irán encapsulados a su vez en paquetes OSC. Con el objetivo de que el *middleware* se sincronice en el envío de datos con la generación de fotogramas por parte del motor gráfico, este último enviará en cada fotograma una petición de datos al *middleware*, que responderá con un paquete que contenga los datos de ese instante concreto o con un

paquete vacío en caso de error. De esta forma, se ahorra carga computacional durante los tiempo donde el motor gráfico no necesite conocer los datos de posición del visor (en pantallas de carga o vídeo de cinemática, por ejemplo) y se sincroniza el arranque del *middleware* con el arranque del juego, con el objetivo de que sea una tarea sencilla para el usuario iniciar el juego con todos los periféricos que utilice. La figura 19 muestra un esquema del protocolo de comunicación diseñado.

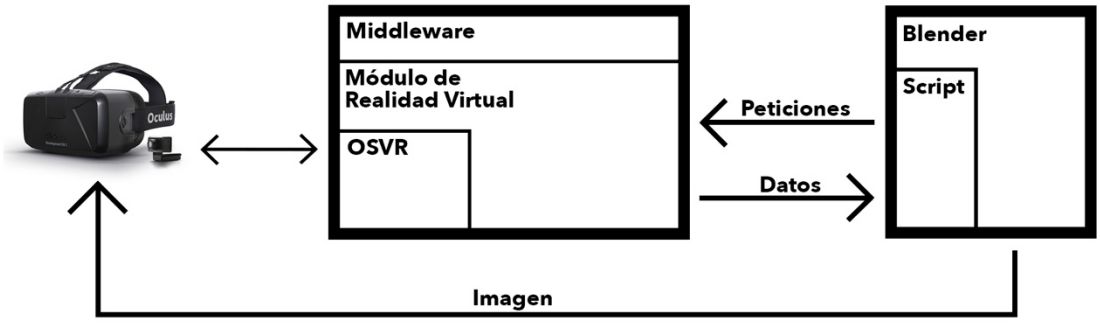

Figura 19 Diagrama de bloques del sistema

4.3.1 Módulo para el *Middleware*

Un requisito de diseño imprescindible en el proyecto es la capacidad de que este se integre en el *middleware* de Kinect de una forma modular, pudiéndose utilizar el *middleware* de forma independiente para un videojuego que utilice únicamente la funcionalidad que aporta la Kinect, únicamente la funcionalidad aportada por el visor de realidad virtual o ambas funcionalidades a la vez.

Para cumplir este objetivo de modularidad de la forma más óptima posible se ha diseñado un sistema de pestañas donde un solo programa maneja varios módulos, independientes y autónomos, contenidos cada uno en una pestaña. En consecuencia, se obtienen una serie de pestañas donde cada una maneja una funcionalidad diferente.

Cada pestaña se debe desarrollar en un proyecto de Visual Studio diferente y debe heredar de la clase “TabPage”. Dicha clase representa el contenido de una pestaña dentro de un control “TabControl”. El proyecto debe estar configurado para que el resultado de la configuración de este no sea un ejecutable por sí mismo sino una librería dinámica, que es un conjunto de funciones sin una función principal que permita que se ejecute por si sola. Estas librerías dinámicas, a diferencia de las librerías estáticas son enlazadas al ejecutar el programa, no al compilarlo, y es el sistema operativo el que debe encontrar dichas librerías. En el sistema operativo Windows estas librerías tienen la extensión “.dll”.

La información de cada librería correspondiente a cada módulo del *middleware*, incluyendo la dirección donde está definida, se guarda en un objeto de datos

"WindowReference" cuyo resultado es también otra librería dinámica (".dll"). Estas últimas librerías se generan gracias a un programa ya desarrollado, ejecutable por ventana de comandos, que crea un objeto "WindowReference" por cada módulo que se le liste, generando además un fichero binario que contiene una lista con todas las referencias a estos módulos llamado "TabList.bin".

Además de todo lo descrito hasta ahora, es necesario un ejecutable que se encargue de gestionar la ejecución de las pestañas. Este ejecutable el resultado final del diseño del *middleware*, el que arrancará el usuario final antes de empezar a jugar y se denomina "Chiro" [10]. Chiro posee un controlador para las pestañas que al ejecutarse consulta el fichero "TabList.bin" para conocer qué pestañas tiene que cargar. A continuación carga las pestañas mediante la información que adquiere del "WindowReference" correspondiente a cada una de ellas. La figura 20 muestra un esquema que ayuda a entender el funcionamiento de este conjunto de herramientas.

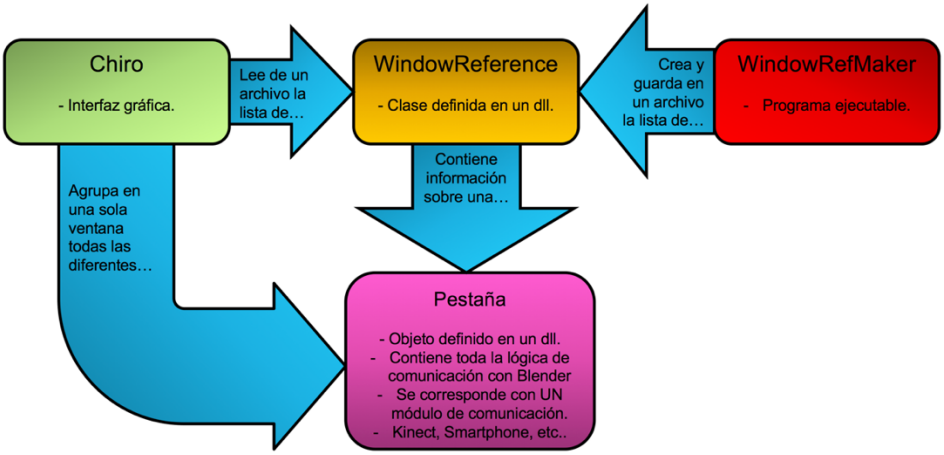

Figura 20 Estructura de proyectos para Chiro [19]

Con esta arquitectura, como se muestra en la figura 21, el *middleware* consigue la modularidad impuesta por el requisito que obligaba a poder ejecutarlo para usar cada funcionalidad por separado o todas de forma simultánea.

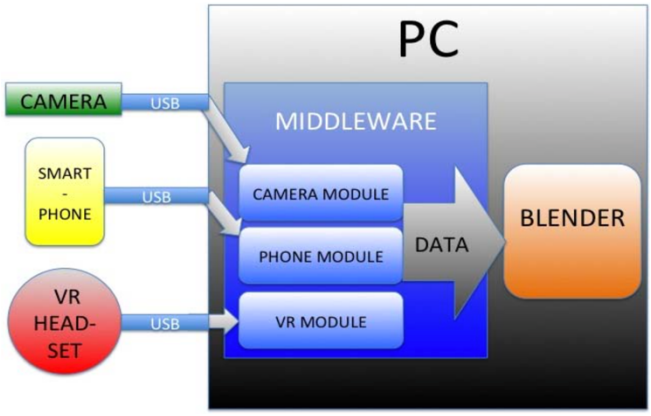

Figura 21 Diseño modular del middleware Chiro [19]

Con relación a la pestaña relativa a la funcionalidad de la realidad virtual, y su funcionamiento interno, caben destacar diferentes aspectos: el primero es que, siendo lo primero que se ejecuta todo el código relativo a la interfaz de usuario, a continuación se crea un nuevo hilo para que toda la gestión de datos del visor de realidad virtual y la comunicación con el motor gráfico se realice de forma concurrente. El segundo aspecto a destacar es que este mismo hilo lanza también la aplicación servidor de OSVR con una configuración por defecto de forma que el usuario tenga que arrancar un solo ejecutable y no dos. El protocolo de comunicación diseñado se ha implementado mediante dos *sockets*, uno de escucha asíncrono que espera la llegada de las peticiones de Blender Game Engine y que cuando recibe dicha petición, ejecuta las órdenes necesarias para la adquisición de los datos de posición y rotación del visor correspondientes a ese momento, los encapsula y envía el paquete para que el motor gráfico lo procese. Todos los paquetes se envían a la dirección IP local (127.0.0.1) y a los puertos especificados por el usuario en la interfaz gráfica de usuario, estando configurados por defecto los puertos 8004 para recepción de datos en Blender y 8005 para recepción de peticiones en el *middleware*. La elección de estos puertos por defecto se debe a que son los siguientes puertos libres después de los ocupados en la configuración por defecto del módulo de comunicación con Kinect del *middleware*. Para identificar de qué puertos se trata tanto en el *middleware* como en Blender, se le ha asignado un nombre a dichos puertos siendo MiddlePort HMD (8005) el puerto de escucha de peticiones en el *middleware* y BlendPort HMD (8004) el puerto de escucha para la recepción de datos en Blender Game Engine.

En lo relativo al protocolo de comunicación, se han establecido dos tipos de peticiones diferentes, para usar una u otra según las necesidades de diseño del videojuego. La diferencia radica en que con una de las peticiones se piden los datos de posición y rotación del visor de realidad virtual mientras que con la otra se piden sólo los datos de rotación. Esto es así porque si un juego solo necesitara los datos de posición, se estarían enviando datos de más que deberían procesarse para descartarse. De este modo, si solo se requieren datos de posición se ahorra memoria y carga computacional. Cuando se recibe una petición u otra el *middleware* llama al servidor OSVR para adquirir los datos ya sean de posición y rotación o solamente de rotación. Estos datos se encapsulan en un paquete OSC similar al utilizado para la comunicación entre Kinect y Blender. La especificación del protocolo OSC permite que cada paquete contenga un mensaje único o un contenedor con varios mensajes únicos. Antes de los mensajes, el protocolo especifica que se debe añadir una cabecera con un campo para indicar el instante al que corresponde dicho paquete y otro campo para indicar el tamaño del mensaje. A continuación el protocolo especifica que cada mensaje único debe contener una cabecera con dos cadenas de caracteres, la primera para indicar la dirección de destino del mensaje expresado de forma similar a un directorio de Windows (en este caso el campo de dirección se rellena con "/HMD/rotation" en el caso del mensaje con los datos de rotación y "/HMD/translation" para los datos de posición). La segunda cadena de caracteres expresa la cantidad y el tipo de los datos contenidos en el mensaje (en este caso ",ffff" equivalente a cuatro datos de

tipo *float*, para los paquetes de rotación, ya que se trata de un cuaternión formado por cuatro números con decimales y “,fff” para los datos de posición al tratarse de un vector tridimensional expresado en tres números con decimales)

En la figura 22 se muestra el tráfico de paquetes generado por la comunicación capturado por el analizador de tráfico Wireshark. Se puede observar como cada petición al puerto 8005 recibe una respuesta con un paquete OSC que contiene un contenedor con los datos de posición y rotación del visor de realidad virtual.

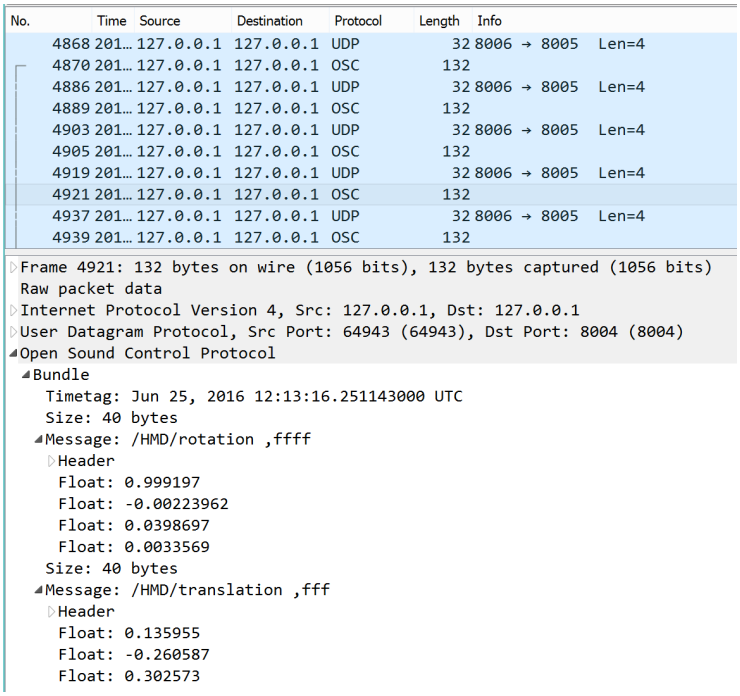

Figura 22 Captura del tráfico generado en la comunicación entre Blender y el módulo de realidad virtual del middleware

La interfaz gráfica de usuario se ha diseñado con una serie de controles que sirven tanto para gestionar los parámetros de funcionamiento del módulo como para comprobar la correcta ejecución de este. Primeramente se encuentra un menú desplegable que permite seleccionar entre los modelos soportados por el módulo. Junto a este menú se encuentra un botón designado a reiniciar el servidor. La funcionalidad de este es reiniciar el servidor con la configuración adecuada para atender al modelo de visor de realidad virtual seleccionado en el menú desplegable que se encuentra junto a él. Esto es así porque el servidor OSVR tiene una serie de ficheros JSON (JavaScript *Object Notation*) de configuración alojados en un directorio donde cada uno corresponde a un modelo de visor soportado. Para que este pueda manejarlo debe arrancar cargando la configuración del modelo que se pretende utilizar. El servidor carga el fichero que se le indique como argumento al arrancarlo, siempre y cuando exista dicho fichero. En caso contrario carga la configuración por defecto que actualmente es la correspondiente al visor Oculus Rift DK2 utilizado para las pruebas de este proyecto. Actualmente, dado que solo se ha podido llevar a cabo el desarrollo con un solo modelo de visor de realidad virtual, es este el que

aparece por defecto en el menú desplegable. En la parte inferior de la ventana hay también dos controles correspondientes a los puertos utilizados para recibir peticiones de datos y para enviar datos. Estos valores se pueden cambiar de ser necesario y los cambios surtirían efecto al pulsar el botón adyacente rotulado con la palabra "*Change*", que reinicia los *sockets* con los valores seleccionados. En caso de que se seleccione el mismo valor para los dos puertos, los *sockets* no se reiniciarán y aparecerá un aviso de color rojo en el selector de los puertos. Junto a los controles de selección de valores para los puertos se ha situado otro botón que inicia la herramienta de pruebas OSVR *Tracker View*. Esta herramienta muestra un sistema de coordenadas y unos ejes situados en la posición en la que se tenga el HMD, y que se mueven de la misma forma que este, trasladándose y rotando de forma solidaria. Resulta una herramienta útil para comprobar el correcto funcionamiento de la comunicación entre el servidor OSVR y el HMD. Además, la interfaz gráfica posee una casilla de verificación con la etiqueta "Mostrar vídeo" que muestra en un cuadro el contenido del monitor secundario del ordenador. Resulta muy beneficioso dado que el visor de realidad virtual se configura como un monitor secundario, pero sólo puede verlo el usuario del visor, cualquier persona que lo acompañe no podría ver lo que está viendo el jugador ni conocer su progreso en el videojuego. De este modo es posible ver en una pequeña ventana lo mismo que está viendo el jugador. Es importante mencionar que esta característica no está optimizada y al utilizarla se ralentiza el funcionamiento del módulo provocando que el juego se ejecute también de forma menos fluida. La figura 23 muestra la interfaz descrita.

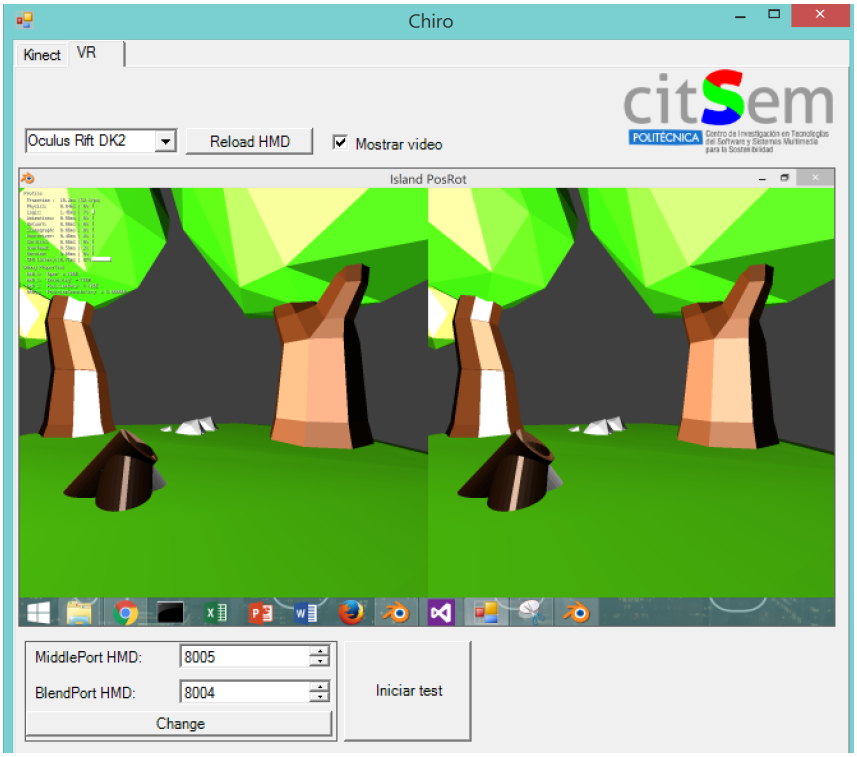

Figura 23 Interfaz gráfica del módulo de realidad virtual para el middleware

En un futuro, dado que el proyecto OSVR está en constante desarrollo, el módulo podrá adaptar las diferentes funcionalidades que este *framework* ofrezca, existiendo la posibilidad de integrar una pequeña aplicación que genere ficheros de configuración para el servidor válidos para un visor de realidad virtual, que aún no esté soportado pero que pueda ser adaptado para estarlo.

4.3.2 Extensión para Blender

El software Blender al ser un programa de código abierto, permite modificarlo según las necesidades del usuario mediante extensiones. Por medio de un *script* escrito en el lenguaje de programación Python es posible añadir funcionalidades que ayuden al usuario desarrollador en su flujo de trabajo, permitiéndole incluso modificar la interfaz gráfica de usuario para añadir aquellos controles que le sean útiles. En este proyecto es necesario un sistema y unos ajustes determinados para las cámaras utilizadas en cada escena del videojuego para lo cual resulta adecuado el desarrollo de una extensión que genere este sistema de cámaras simplificando el trabajo al desarrollador que esté trabajando en el diseño del videojuego, ya que podrá utilizar el sistema de cámara generado por la extensión y adaptado a la funcionalidad de la realidad virtual como si se tratara de cualquier otra cámara. Esto además permite adaptar cualquier videojuego ya desarrollado en Blender Game Engine para que pueda ser utilizado con el visor de realidad virtual simplemente sustituyendo la cámara concreta en la que se quiera usar el visor por el sistema de cámara que genere la extensión.

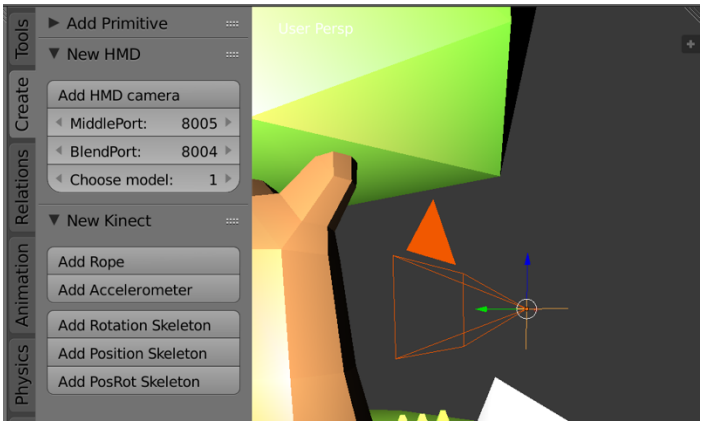

Figura 24 Controles que proporciona la extensión y cámara que genera con objeto "empty" de referencia

De esta manera se ha desarrollado una extensión que cumple con los propósitos descritos anteriormente: facilitar y agilizar el proceso de desarrollo de cualquier videojuego que integre esta funcionalidad. Esta extensión añade en la pestaña "*Create*" del modo objeto un pequeño panel con dos opciones y un botón, tal y como muestra la figura 24. Las opciones sirven para seleccionar los puertos a los que enviar las peticiones de datos y recibir las respuestas con los datos. El botón sirve para añadir a la escena, en la posición donde se encuentre el cursor 3D, un sistema de cámara con toda la lógica necesaria para enviar y recibir datos en los puertos seleccionados en las opciones recién descritas. Los

elementos que añade la extensión a la escena al pulsar este botón son una cámara y un objeto "*empty*" estando la cámara emparentada al objeto "*empty*". Al crearlos además les asigna un nombre siendo el objeto "*empty*" llamado "HMDRef_1" y la cámara "HMD_1" en el caso de ser la primera cámara de este tipo que se añade en la escena, aumentando el número que aparece en el nombre según aumente el número de cámaras en la escena. El objeto "*empty*" sirve como referencia de posición y rotación para la cámara. Si por ejemplo se desarrollara un videojuego donde el protagonista es un personaje y todo se desarrolla desde el punto de vista de este, habría que unir el objeto "*empty*" al cuello del personaje, de forma que la cámara seguiría sus movimientos al estar emparentada al objeto "*empty*". El sentido de que esto sea así reside en que las variaciones de posición del visor tienen como efecto un movimiento proporcional de la cámara respecto del objeto de referencia, el objeto "*empty*". De este modo se consigue que la posición de la cámara siga al personaje independientemente de cómo se controle el movimiento de este, y las variaciones de posición del visor de realidad virtual solamente se vean reflejados en pequeños cambios en el punto de vista del personaje, como si este moviera el cuello.

Para que todo esto ocurra, la extensión, además de crear el sistema de cámara en la escena, añade al objeto cámara los ladrillos lógicos y las propiedades necesarias. Concretamente añade un sensor "*Always*", que se activa en cada fotograma generado, enlazado con un controlador "Python" que ejecuta un *script* denominado "RunHMD.py" cuyo funcionamiento se explicará en el siguiente apartado junto con las propiedades y la función de todas ellas. Además de añadir esta lógica, ajusta algunos parámetros necesarios de la cámara, fijando la distancia focal correcta, configurando la cámara en modo "*stereo*" y "*side by side*" y el tipo de *frame* en modo "*extend*". Se puede observar toda la lógica contenida en el objeto cámara en la figura 25.

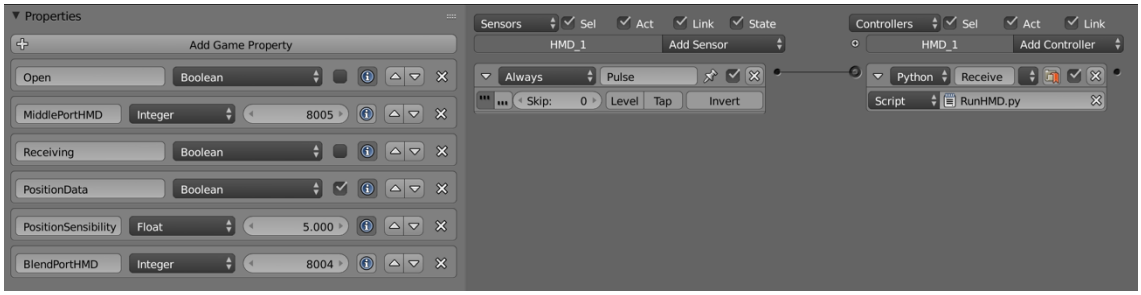

Figura 25 Lógica y propiedades generados por la extensión en el objeto cámara

Por consiguiente, esta extensión permite crear sistemas de cámara ya configurados para su uso en cualquier desarrollo, agilizando el flujo de trabajo y reduciendo el tiempo de desarrollo de un juego que vaya a utilizar esta funcionalidad. El anexo 1 de la presente memoria contiene una explicación detallada sobre cómo utilizar el conjunto de la extensión y el *script* para crear videojuegos que utilicen esta función, así como modificar videojuegos para que puedan ser utilizados con un visor de realidad virtual.

4.3.3 *Scripts* en Blender

La extensión explicada en el apartado anterior enlaza el sistema de cámara que genera a un *script* que se ejecuta periódicamente cada vez que un fotograma es generado. Pues bien, este *script* se ha desarrollado también bajo el lenguaje Python y con la API (*Application Programming Interface*) de desarrollo que brinda Blender y realiza las funciones de petición de datos al *middleware*, recepción de la respuesta con los datos procesamiento de los datos para la generación del fotograma.

La primera operación que realiza el *script* es obtener el objeto al que está asociado el controlador desde el que se ejecuta el *script* para crear dos *sockets* asociados a este objeto: uno de recepción de datos, por defecto en el puerto 8004 denominado "BlendPortHMD" y otro de envío de peticiones por defecto al puerto 8005 denominado "MiddlePortHMD". Los puertos de ambos *sockets* reciben el mismo nombre que en el módulo del *middleware* y ambos deben de estar configurados igual en ambos extremos (en el *middleware* y en Blender) para que exista la comunicación. Es necesario remarcar que las dos opciones mostradas en el modo objeto que servían para elegir los puertos a la hora de la creación del sistema de cámara, son relevantes solo en ese momento. Si se cambian dichas opciones tras haber creado el sistema, estos cambios no tienen efecto sobre dicho sistema ya creado. Por el contrario, una vez creado el sistema de cámara es necesario cambiar los valores de los puertos en los campos de las propiedades creadas en el objeto cámara con los nombres de los puertos ("MiddlePortHMD" y "BlendPortHMD"). Esto debe ser así porque el *script* crea los *sockets* a partir del valor que tengan las propiedades del objeto en el momento de crear cada fotograma.

A continuación el *script* comprueba primeramente una propiedad *boolean* denominada "Open" que funciona como interruptor de encendido del sistema. Esto es útil para activar y desactivar la comunicación si durante un juego existen momentos donde no es necesaria la recepción de datos de posición del visor de realidad virtual. Después se comprueba otra propiedad *boolean* denominada "PositionData" y en función de si es verdadera o falsa envía una petición de tipo rotación y posición o una petición de solo rotación. Tras haber enviado la petición se activa el *socket* de recepción que espera la respuesta por parte del *middleware* con los datos requeridos. Al recibirlos, primero comprueba que el paquete no esté vacío y en tal caso desencapsula el paquete OSC, comprueba si es el paquete del tipo solicitado (rotación y posición o solo posición) y aplica la rotación al objeto cámara directamente y la variación de la posición respecto del objeto "empty" utilizado como referencia. En caso de no recibirse el paquete a tiempo, el *socket* lanza una excepción que permite que el juego no se bloquee y continúe aunque no tenga datos para actualizar la cámara en ese fotograma.

El *script* se ha desarrollado de forma que no utiliza las funciones propias del intérprete de comandos de Blender si no específicamente las que utiliza el motor gráfico Blender Game Engine. Esto permite arrancar el juego en modo *standalone* en una venta

independiente que si se abre en el monitor secundario, que es como está configurado el visor de realidad virtual, permite visualizar el videojuego a la perfección.

4.3.4 Resultados

El módulo que se pretendía crear como objetivo de este proyecto se ha desarrollado satisfactoriamente. Este se ha integrado en el *middleware* Chiro de forma que funciona como un módulo más de este. Por tanto, se ha conseguido que los controles de la interfaz gráfica de esta funcionalidad estén unificados con los que ya tenía el *middleware*. Todos los procesos que lanza este módulo son independientes y pueden ejecutarse concurrentemente con los de cualquier otro módulo. El protocolo de comunicación implementado funciona de la forma esperada, sin añadir latencia a la generación de fotogramas en Blender. La lógica añadida a la cámara de Blender se ejecuta también de forma fluida permitiendo una rápida ejecución de las peticiones y el procesado datos recibidos. Tan solo la función introducida en la interfaz gráfica que permite ver lo que se está visualizando en el visor de realidad virtual ralentiza la ejecución del juego cuando se activa. Es por esto que habría que optimizarla para mejorar su funcionamiento. Las pruebas subjetivas durante la fase de desarrollo han demostrado que el sistema permite que se ejecuten juegos en Blender a 60 fotogramas por segundo sin causar sensaciones de mareo en un entorno virtual estático. Esta fluidez en la ejecución se ha conseguido en dos ordenadores de prueba que contaban con un hardware lo suficientemente potente como para no tener problemas de latencia en la ejecución.

4.4 *Smartphone* como HMD

Tal y como se explica en el apartado de análisis de los dispositivos de realidad virtual disponibles en la actualidad, la forma más asequible de obtener un sistema que permita obtener una experiencia de juego inmersiva es utilizando el hardware de un *smartphone* para tal fin. En la actualidad, prácticamente todos los *smartphones*, independientemente de a qué gama pertenezcan, tienen sensores giroscopios y algunos también magnetómetros que permiten hacer una monitorización de las variaciones de rotación que puedan sufrir los dispositivos. Además, cada vez más teléfonos incorporan pantallas con una densidad de píxeles realmente alta que permiten utilizar la pantalla de estos como interfaz de salida en la que mostrar el entorno virtual.

En la actualidad ya es posible utilizar estos dispositivos de este modo situando el *smartphone* en un soporte específicamente diseñado como el mostrado en la figura 26. Esto es posible gracias a que es el propio *smartphone* el que se encarga de ejecutar el videojuego llevando a cabo las labores de adquisición y procesado de datos de posición como de renderizado de imagen en su propia pantalla. Los soportes se pueden comprar cada vez en más superficies y su precio es muy reducido al no contener electrónica alguna. Incluso es posible fabricar uno con materiales tan habituales en los hogares como el cartón obteniendo unos resultados igualmente buenos.

Figura 26 Soporte utilizado para usar el smartphone como HMD

A pesar de que de esta forma sea tan sencillo utilizar un *smartphone* como visor de realidad virtual, plantea problemas al intentar utilizarlo de la misma forma que se utiliza el Oculus Rift DK2 en el *middleware*. El principal impedimento para poder utilizar cualquier *smartphone* para integrarlo en el *middleware* es el envío de datos en ambas direcciones con una latencia suficiente como para permitir una experiencia de juego fluida que no cause sensación de mareo o similar. Primeramente, sería necesaria la adquisición de datos de rotación por parte del *smartphone* y el envío de los mismos al *middleware,* el cual se encargaría de enviarlos a Blender Game Engine como si se tratara de cualquier otro visor de realidad virtual ya soportado. Este procesaría los datos de la misma forma y renderizaría en tiempo real una secuencia de vídeo que debería ser presentada en la pantalla del *smartphone* con la menor latencia posible. En los siguientes apartados se va a abordar el análisis de los retos que esto plantea con el objetivo de buscar la solución más óptima.

4.4.1 Captura de rotación

Al igual que el dispositivo utilizado para el desarrollo realizado, Oculus Rift DK2, los *smartphones* poseen sensores de orientación que permiten evaluar las variaciones que se producen cuando el dispositivo se mueve. Esto es suficiente como para obtener unos datos bastante precisos acerca de la orientación del dispositivo, pero no de la posición, dado que en los visores de realidad virtual diseñados para este propósito, se utiliza un sistema de seguimiento externo mediante una cámara de infrarrojos, como ya se explicó anteriormente. Esto reduce sensiblemente la experiencia de inmersión en el mundo virtual ya que a pesar de mover el usuario la cabeza en el mundo real, no percibirá un cambio de

perspectiva en el mundo virtual. A pesar de ello, la sensación es, por supuesto, lo suficientemente inmersiva como para no desestimar esta opción.

La adquisición de estos datos mediante la lectura de los sensores integrados en el hardware del *smartphone* debe hacerse necesariamente desde una aplicación específicamente desarrollada para este fin. Los diferentes sistemas operativos para móvil como Android e iOS ofrecen en su kit de desarrollo de aplicaciones funciones suficientes como para llevar a cabo esta tarea dado que es habitual que muchos juegos y aplicaciones que se ejecutan en el propio dispositivo hagan uso de estos sensores para el control de la aplicación o juego.

Una vez obtenidos estos datos deben ser enviados al *middleware* dentro de paquetes a través del canal adecuado. Se trata de una cantidad de datos muy pequeña ya que se trata de enviar las componentes de un vector o un cuaternión para cada fotograma, lo que supone una tasa binaria de algo menos de 8 kbps. El canal para el envío de estos datos podría ser tanto *bluetooth*, que en su versión 3.0 permite hasta 24Mbps [20], como wifi.

La velocidad a la que se envíen estos datos y sobre todo la reducción máxima de la latencia de envío, recepción y procesado es vital para que el sistema pueda llegar a funcionar con la fluidez requerida. Además, dado que debe estar sincronizada la adquisición de datos con la generación de fotogramas para lo cual, de la misma forma utilizada para sincronizar el visor Oculus Rift DK2, la primera acción dentro del flujo de la comunicación será la petición que envíe el motor gráfico. Esto ya aumenta, aunque sea una cantidad mínima, la latencia del sistema.

4.4.2 Envío de imagen a *smartphone*

El punto más crítico de este sistema se encuentra en la forma de presentar la imagen generada por el motor gráfico Blender Game Engine en la pantalla del *smartphone* de la forma adecuada. En la configuración por defecto, la imagen renderizada tiene una resolución de 1920 píxeles de ancho por 1080 de alto. Esto corresponde con la resolución de la mayoría de los visores de realidad virtual disponibles, aunque algunos tienen una resolución algo mayor, dado que es la resolución mínima para que la densidad de píxeles sea suficiente y así situar la pantalla a pocos centímetros de los ojos del usuario y que no se aprecien en exceso los píxeles por separado. Además, la mayoría de los *smartphones* de gama media tienen una pantalla con dicha resolución, habiendo también dispositivos de gama alta con una resolución mayor. De cualquier modo es la resolución adecuada para un funcionamiento correcto del sistema. También hay que tener en cuenta que la tasa de fotogramas necesaria para que estos sistemas de realidad virtual provoquen una inmersión en el usuario sin causar sensación de mareo es de 60 fotogramas por segundo. Una secuencia de vídeo a una resolución de 1920x1080 píxeles a 60 fotogramas por segundo genera un flujo de datos con un régimen binario de cerca de 6 Gbps cuando el vídeo no tiene ningún tipo de compresión. No hay ningún canal disponible en un

smartphone con ese ancho de banda, lo cual hace necesario comprimir el vídeo hasta obtener un régimen binario lo suficientemente reducido como para ser transmitido con muy baja latencia al *smartphone* sin comprometer en exceso la calidad del vídeo. Esta codificación debe ser realizada desde el ordenador que está ejecutando el videojuego y esto supone una carga computacional muy elevada para el ordenador, lo que significa que se debe seleccionar un *codec* de vídeo adecuado, que suponga la menor carga computacional posible, para no comprometer la fluidez en la ejecución del videojuego por parte del motor gráfico.

4.4.3 Soluciones existentes analizadas

Se ha realizado una búsqueda de soluciones que aporten esta funcionalidad: ejecutar un videojuego en un ordenador y poder jugarlo mediante realidad virtual utilizando como interfaz un *smartphone*. Las tres soluciones parten de una aplicación para Android que lleva a cabo el envío de datos de orientación y recibe el flujo de vídeo para presentarlo en la pantalla del *smartphone*.

Moonlight

Se trata de una aplicación de código abierto que promete *streaming* de vídeo desde un PC a un *smartphone* con una latencia baja a 60 fotogramas por segundo. Permite además utilizar mandos de control soportados, como los de la Xbox de Microsoft. Se puede enviar la imagen a diferentes plataformas, tanto a iOS como a Android e incluso a una Raspberry Pi. Es gratuito, tanto las aplicaciones cliente para los *smartphone* como la aplicación servidor que realiza el *streaming*. Además, al ser una solución de código abierto, su código está disponible en un repositorio de GitHub para su descarga y modificación si fuera necesario. Esta solución en especial es una implementación de código abierto del protocolo NVIDIA GameStream que permite transmitir los juegos desde el PC a cualquier dispositivo NVIDIA Shield. La cualidad de este protocolo, que significa una limitación, es que aprovecha la arquitectura y potencia de las tarjetas gráficas NVIDIA para hacer la codificación del vídeo, lo cual impide que pueda ser utilizado con tarjetas gráficas integradas de Intel o dedicadas de AMD, que son lo más habitual en los ordenadores de gama media y baja.

KinoConsole

Este software nace con el propósito de poder jugar a videojuegos instalados en el PC y que se ejecutan en este controlándolos desde un dispositivo móvil como un *smartphone* o una tableta pero no está orientado específicamente a la realidad virtual aunque si plantea la posibilidad de utilizarlo para ese fin. Igual que Moonlight, promete transmitir la señal de vídeo a 60 fotogramas por segundo y con una latencia reducida desde el monitor del ordenador a la pantalla del *smartphone* o tableta de forma que este funcione como monitor. La interfaz gráfica de usuario del software posibilita configurar la sincronización entre el *smartphone* y el PC, permite buscar en el propio PC los juegos instalados, estén

preparados para la realidad virtual o no, y configurar la entrada de los controles para los juegos de realidad virtual. En principio parece una solución muy aceptable dada su compatibilidad con cualquier tarjeta gráfica y que se trata de un software gratuito en su versión más básica. No obstante, no se ha podido realizar una prueba de funcionamiento ya que da errores a la hora de sincronizar el servidor con el *smartphone* y no ha sido posible hacerlo funcionar.

RiftCat

Este software es algo más completo que un servidor de *streaming*. Se trata de un entorno parecido a la exitosa plataforma de juegos online Steam, ya que es necesario tener una cuenta de usuario y descargar su aplicación de escritorio para empezar a utilizarlo. Esta aplicación de escritorio posee un apartado de sincronización con el *smartphone* con una interfaz limpia y sobre todo muy sencilla que funciona rápidamente sin complicadas configuraciones. También posee un navegador para su biblioteca de juegos disponibles desde la cual se pueden descargar diferentes títulos y ejecutarlos desde ese mismo software, realizándose automáticamente la sincronización y el comienzo de la transmisión de vídeo al *smartphone*. Tiene también un panel de configuración que sugiere configuraciones óptimas según el modelo de *smartphone* que se utilice y permite cambiar y personalizar las opciones de codificación del vídeo así como cambiar el motor de codificación en función del hardware del que disponga el PC. Se trata de una aplicación clara y sencilla de utilizar que ofrece un resultado sorprendente. La latencia durante el juego es realmente baja y la calidad del vídeo emitido es aceptable. En el entorno de pruebas utilizado, con una red inalámbrica como la que se puede encontrar en un hogar medio, y un *smartphone* de gama media (LG Nexus 5X) el funcionamiento es correcto. La transmisión del vídeo a veces presenta fallos en la codificación como los mostrados en la figura 27, y puede llegar a cortarse durante breves instantes, pero la sensación general es inmersiva. A pesar del correcto funcionamiento del sistema hay que destacar que la experiencia de juego está muy por debajo en calidad que la obtenida en el visor de realidad virtual Oculus Rift DK2. De hecho la prueba realizada ha sido corta porque producía una experiencia inmersiva pero incómoda, provocando sensación de mareo a pesar de ser un juego en el que el entorno no estaba en movimiento y el personaje del juego tampoco. Para poder utilizar el *smartphone* como visor se ha utilizado un soporte específicamente diseñado para tal fin obtenido a través de una gran superficie de venta online por un valor de 12 euros. El modelo de soporte utilizado no resultaba para nada cómodo debido a su tamaño, que era demasiado pequeño, a pesar de que se trataba de un modelo para adulto.

Figura 27 Captura de la pantalla del smartphone cuando se daban errores en la transmisión

4.4.4 Resultados

Primeramente cabe destacar que no se ha llegado a realizar ningún desarrollo específico para dar soporte al *middleware* a este tipo de sistema de realidad virtual porque se escapa del alcance en tiempo del presente proyecto. Sólo se trata de un análisis que pretende alcanzar la que sería la solución más óptima para llevar a cabo dicho desarrollo. También es importante señalar que aunque se trata de un desarrollo complicado para obtener un resultado óptimo, es posible conseguir un funcionamiento aceptable como se ha demostrado en el apartado anterior con la prueba del software RiftCat.

Dada la comunicación bidireccional que este sistema requiere, sería necesario llevar a cabo el desarrollo de una aplicación para móvil que realizara la gestión de la recepción de peticiones de datos, adquisición de los mismos y envío de los datos al *middleware* así como la recepción del vídeo generado por Blender Game Engine. Los dos grandes sistemas operativos para *smartphone* permiten el desarrollo de una aplicación de estas características. No obstante, los dispositivos iOS son todos de gama alta y tienen un coste elevado frente a los dispositivos Android, que al haber una grandísima gama de dispositivos, los hay más asequibles. Dado el requisito de diseño de que el sistema pueda ser accesible al mayor número de personas posible, es más adecuado hacer el desarrollo para Android de forma nativa. Estudiando los posibles métodos de desarrollo de la aplicación quizá se pueda encontrar una forma de hacer un solo desarrollo compatible con ambas plataformas con una eficacia aceptable.

En cuanto al canal de comunicación a utilizar, realmente solo hay como opciones *bluetooth* o wifi. El *bluetooth* es adecuado para el envío de los datos de posición así como para la recepción de las peticiones de datos, sin embargo no tiene un ancho de banda suficiente como para incluir además el flujo de datos generado por el vídeo comprimido

que solo puede ser transmitido mediante wifi. Teniendo en cuenta este hecho, parece más adecuado establecer un protocolo de comunicación a través de la conexión wifi que implemente todas las comunicaciones necesarias para el funcionamiento del sistema. De este modo se reducirá sensiblemente el consumo de batería del *smartphone* al no tener que usar la conexión *bluetooth*. En resumen, se trataría de una aplicación para *smartphone* capaz de sincronizarse con el *middleware* para que ambos estén comunicados, recibir peticiones de datos de orientación del dispositivo, llevar a cabo la adquisición y envío de dichos datos y por último, la recepción del flujo de datos de vídeo, descodificarla y presentarla en pantalla. Todo esto con una latencia inferior al periodo de duración de un fotograma (16,7 ms).

Del lado del *middleware*, sería necesario adecuar el desarrollo de este para realizar el reenvío de las peticiones de datos, recibidas de Blender Game Engine, a través de la red local hasta el *smartphone* y recibir a través de la red los paquetes con los datos de orientación del dispositivo. Esta modificación no lleva consigo demasiada complicación, pero es necesario implementar también un método de compresión del flujo de datos de vídeo generado por Blender Game Engine y el envío de este a través de la red local. Este módulo de envío de datos de vídeo puede ser implementado mediante cualquiera de las diferentes soluciones de código abierto para *streaming* que están disponibles para uso bajo licencias Apache.

En caso de que en un futuro se desease das soporte también a otro motor gráfico se podría continuar utilizando esta adaptación. Lo que sí sería necesario implementar en dicho motor sería toda la lógica ya implementada en Blender que realiza las peticiones de datos y recibe las respuestas para procesarlas. Al enviar la imagen generada por el motor gráfico a través de *streaming* al *smartphone*, no importa cual sea el motor, lo imprescindible es enviar el flujo de vídeo que este genere al *smartphone*. Además, el middleware es el que gestiona la adquisición de datos de posición, independientemente del motor gráfico al que luego se los envíe.

5. Pruebas realizadas

Una vez terminado el desarrollo, se llevó a cabo una serie de pruebas con el objetivo de comprobar el correcto funcionamiento del sistema así como tratar de llevar a cabo una medición del valor aportado por la funcionalidad creada. Dado que el objetivo final del proyecto madre de este es crear un videojuego serio orientado a la rehabilitación de personas con movilidad reducida mediante *exergames*, primeramente se llevó a cabo una sesión de demostración con una fisioterapeuta experta en el área, la cual pudo evaluar el valor del proyecto además de aconsejar los movimientos y ejercicios más utilizados con este tipo de personas. También se adaptaron los minijuegos ya desarrollados para poder ser utilizados en el modo de realidad virtual, de esta forma se pudo comparar las experiencias de los jugadores al utilizar el mismo juego en una pantalla tradicional con la experiencia de jugarlo en realidad virtual.

5.1 Sesión con fisioterapeuta

Se ha llevado a cabo una sesión de demostración con una fisioterapeuta especializada en el tipo de rehabilitación que pretende reforzar este proyecto. Con el objetivo de que ella pudiera evaluar el proyecto y su valor para potenciar la clase de ejercicios que le son necesarios realizar a sus pacientes se le han mostrado todos los minijuegos diseñados uno por uno, mostrándole los movimientos necesarios para progresar en el juego y la mecánica del mismo. A continuación, dado que aún no se tenía adaptado ninguno de los minijuegos al visor de realidad virtual, se le ha explicado el objetivo del visor y se le ha mostrado una demostración de un entorno virtual estático para que comprobara la sensación que aporta la realidad virtual.

Sus impresiones han sido, en general, positivas. En cuanto a los videojuegos sus comentarios hacían ver que le parecían apropiados y entretenidos y en cuanto a los movimientos necesarios para progresar en ellos, también le han parecido adecuados y correctos para el tipo de ejercicios que sus pacientes realizan en sesiones de rehabilitación. También ha aportado ideas de movimientos útiles, tanto con brazos como con cabeza y cuello. Ha destacado que el tipo de pacientes con los que está habituada a tratar suelen buscar formas alternativas para conseguir realizar tareas con menos esfuerzo, lo cual puede influir en los juegos, ya que si el paciente descubre que haciendo cierto movimiento que le cuesta menos esfuerzo avanza igual en el juego, se acabará realizando un esfuerzo mucho menor y probablemente unos ejercicios que no son los que están pensados para el juego. Esto hace que sea importante la estabilidad de la detección del esqueleto proporcionado por el sensor Kinect y una programación correcta de la lógica del juego para que sólo se pueda avanzar realizando el ejercicio correctamente, en la medida en la que el paciente pueda.

En cuanto a la realidad virtual, ha hecho comentarios positivos acerca de lo que puede motivar al paciente, sin embargo, ha advertido de la sensación de mareo que pueden provocar los sistemas de realidad virtual que no estén bien optimizados, ya que decía haberlo probado con anterioridad y haber sentido esta sensación.

5.2 Pruebas con discapacitados

En la semana del 13 al 19 de junio se llevó a cabo una batería de pruebas de situaciones de juego reales con potenciales usuarios del sistema. Durante cuatro días se recibieron voluntarios con diferentes discapacidades como atrofia muscular espinal, parálisis cerebral, distrofia muscular o distrofia muscular de Duchenne. El objetivo de las sesiones era probar el sistema y que compartieran sus impresiones acerca del mismo, buscando puntos débiles y de mejora. Para realizar las pruebas se habilitaron tres puestos por los que pasó cada voluntario. El primer puesto servía para probar un software denominado MoKey (*Motion Keyboard*) que sirve para traducir los movimientos capturados por el sensor Kinect a pulsaciones de teclas del teclado. El segundo puesto era para comprobar

el funcionamiento del *middleware* en conjunto con los minijuegos desarrollados en las semanas anteriores. A este conjunto se le ha denominado con el nombre Blexer. Finalmente en el tercer puesto estaban habilitados los mismos minijuegos que se probaban en el segundo puesto pero habilitados para ser jugados con el visor de realidad virtual mediante el módulo desarrollado en este proyecto.

El proceso de pruebas al que se sometían los voluntarios comenzaba con una breve introducción al proyecto, donde se explicaban los objetivos del proyecto para después pasar a rellenar un cuestionario previo a las pruebas. El objetivo del cuestionario era obtener información sobre del voluntario, acerca de su discapacidad, sus hábitos de uso de ordenadores y videoconsolas y sus preferencias en el ámbito de los videojuegos.

Posteriormente comenzaban a probar cada puesto comenzando siempre por el primer o el segundo puesto, dado que el tercero dependía directamente de los datos obtenidos en el segundo puesto y por tanto estos dos puestos debían realizarse en orden. Cuando un voluntario comenzaba las pruebas en el segundo puesto la primera operación siempre era lograr la correcta detección del jugador por la Kinect. En el caso de aquellos voluntarios que no utilizaban silla de ruedas no había problema y situándose frente al sensor rápidamente se conseguía una correcta captura del movimiento de éste. Sin embargo, con algunos voluntarios que usaban sillas de ruedas eléctricas grandes, las cuales no podían girar el asiento u opciones similares, existieron problemas para encontrar una colocación adecuada del sensor respecto del usuario, llegando incluso a ser imposible la detección con un voluntario en particular. Es necesario destacar que el *middleware* posee un modo de detección de esqueleto denominado "Modo Sentado" que solo detecta brazos, hombros y cabeza con el cual si se conseguía una detección aceptable en estos casos, pero no se pudieron probar los juegos correctamente dado que la extensión de Blender que recibe los datos del *middleware* no estaba preparada para ese modo. Una vez conseguida la correcta detección, se ejecutaba un *script* en Blender con el que se obtenían los parámetros de movimiento máximo de los usuarios y se generaba un fichero con sus parámetros personales que afectaban al comportamiento del control de los minijuegos, haciendo que estos pudieran ser controlados de una forma similar con independencia de la capacidad de movimiento del jugador. Fue importante probar esta característica con gente que poseía discapacidades tan dispares y observar el comportamiento de los juegos al poner en funcionamiento esta característica que pretende, precisamente, paliar esas diferencias en los jugadores. A continuación pasaban a probarse los cuatro minijuegos que se habían desarrollado. El primero de ellos se trataba de subir una escalera vertical, obligando al jugador a realizar los gestos necesarios para subir este tipo de escalera. Otro juego proponía remar en una barca realizando movimientos con los brazos hacia delante y hacia detrás. El siguiente juego consistía en golpear a una serie de topos que salen en orden aleatorio de unos agujeros con un mazo. El mazo se controlaba con el brazo derecho del jugador, de la misma forma que si este estuviera sujetando el mazo con su mano. Estos tres juegos habían sido desarrollados por el alumno de prácticas Pablo Parra Iglesias. Finalmente el último juego permitía controlar el vuelo de un avión en un cielo donde

había una serie de anillos que se debían atravesar. El control del avión se llevaba a cabo con los brazos, permitiendo girar a un lado o a otro según la inclinación de la recta que une una mano con otra, es decir se podía girar a la izquierda elevando el brazo derecho y dejando el brazo izquierdo más bajo y a la derecha elevando el brazo izquierdo y manteniendo el derecho abajo. Para controlar el timón de profundidad del avión era necesario inclinar el torso hacia delante para bajar o hacía detrás para subir. Este juego había sido desarrollado por Ignacio Gómez-Martinho.

Tras terminar estas pruebas en el segundo puesto, el voluntario pasaba al tercer puesto donde se probaban los cuatro juegos anteriormente descritos pero habilitados para ser jugados con el visor de realidad virtual. La única diferencia en los juegos era que el punto de vista cambiaba y se veía como se muestra en la figura 28, con el objetivo de que se jugara al juego en primera persona y se lograra más inmersión en el juego.

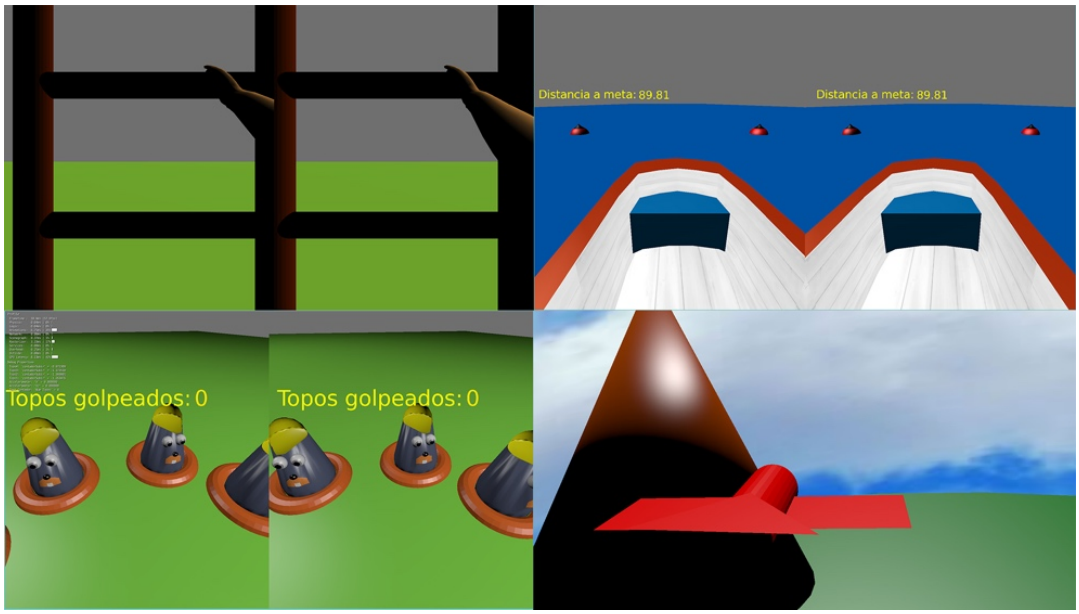

Figura 28 Capturas de pantalla de los juegos vistos desde el visor de realidad virtual

En el juego de la escalera se observaron reacciones relativas principalmente a errores de programación o diseño del juego, como que la cámara sólo se desplazaba al desplazar el brazo derecho y no al desplazar el izquierdo, que la cámara estaba situada más abajo de lo que realmente debería con respecto al personaje del juego y en general hubo problemas con la detección del esqueleto y la forma en la que el juego respondía al movimiento real del jugador. Para que el jugador perciba que posee el control total del juego es importante que sus movimientos se vean reflejados fielmente en el avatar del juego, es decir, cuando el jugador levanta un brazo y lo está mirando con el visor de realidad virtual, el brazo del avatar debe reaccionar de forma similar a como lo está moviendo el jugador, de otra forma el jugador tiene la sensación de que el juego no está respondiendo realmente a sus movimientos. Aquellos voluntarios que probaron por primera vez un visor de realidad virtual con este juego mencionaron la impresión que les causa al principio, siendo una

impresión inesperada pero positiva y en general, ningún comentario acerca de posibles mareos.

Figura 29 Voluntaria jugando al juego de subir una escalera con el visor de realidad virtual

En el juego de las barcas las opiniones han sido dispares pero casi siempre relativas al control del juego más que a la realidad virtual. A varios les costó encontrar el movimiento preciso para hacer que la barca avanzara aunque finalmente consiguieron dar con él. Hubo comentarios sobre que el juego se hacía largo y monótono, dándose el caso concreto de un niño que pidió el siguiente juego porque ese le aburría. En este caso algún voluntario si ha mencionado que le causaba algo de mareo pero la mayoría han mencionado más la impresión de la realidad virtual que la sensación de mareo.

En tercer lugar los voluntarios jugaban al juego de golpear topos. Las opiniones al probar este juego coincidían mucho en la dificultad de usar el mazo, que daba la sensación de que no respondía con suficiente fidelidad a los movimientos del jugador. Algunos casos incluso era imposible para el jugador conseguir que el mazo llegara a golpear a los topos. Otra opinión generalizada, incluso en los caso donde conseguían manejar el mazo mejor, era que resultaba muy complicado llegar a golpear los topos que salían por el lado izquierdo del jugador, ya que el avatar tenía el mazo en el brazo derecho y por tanto esos topos le quedaban a una distancia mayor. En cuanto a las opiniones acerca de la realidad virtual, muchas fueron en la dirección de la dificultad, ya que por la posición de la cámara, en el visor de realidad virtual no se veían todos los topos a la vez, sino que había que girar la cabeza a derecha e izquierda para ver cuando salían. También hubo dos casos en los que les resultó más sencillo el juego en realidad virtual respecto a sin realidad virtual, pudiendo controlarlo mejor.

En último lugar los voluntarios probaban el juego del avión. Este juego se ponía siempre en último lugar con el objetivo de que los voluntarios ya se hubieran acostumbrado al uso del visor de realidad virtual y que ya poseyeran cierta práctica en el control del avión, ya que se trataba de un juego ya de por si complicado de controlar, y que con el visor de realidad virtual se complicaba incluso más. Sólo aquellos que consiguieron hacerse con el control del juego en el puesto anterior pudieron controlarlo con mediana soltura en este puesto, pero la opinión general era que el control era muy poco intuitivo, lo cual les hacía salirse del mapa virtual creado, dejándoles la pantalla en gris y resultándoles realmente difícil volver al mapa. Ni siquiera haciendo varios intentos con cada usuario conseguían hacerse con el control. A pesar de ser el juego que más sensación de mareo podía haber provocado, hubo pocos usuarios que se quejaran de ello.

En relación a los ejercicios realizados por los voluntarios cuando usaban el visor de realidad virtual respecto a cuando no lo usaban, no se apreciaron diferencias significativas en los movimientos realizados, realizaban los movimientos completos esforzándose del mismo modo. En los casos en los que existiendo mala detección del usuario o mala respuesta del juego a sus movimientos los jugadores si exageraban más los movimientos para intentar que el juego respondiera mejor, a pesar de que no fuera una cuestión de amplitud de los movimientos si no más bien de errores en la detección o en la respuesta del juego.

Al finalizar las pruebas los voluntarios rellenaron un breve cuestionario donde se les preguntaba acerca de sus impresiones al llevar a cabo las pruebas de modo que pudieran aportar algo de información útil para la continuación del desarrollo de la plataforma. La parte final de la encuesta iba enfocada a la experiencia de haber probado los minijuegos con el visor de realidad virtual. Primeramente se les preguntaba si habían probado anteriormente un visor de realidad virtual, ya que la primera vez que se usa un dispositivo de este tipo suele causar una impresión bastante fuerte en el usuario, pero es importante tener en cuenta que esa sensación es mucho menor cuando el usuario está acostumbrado a utilizar visores de realidad virtual. Ninguno de los voluntarios lo había probado, lo cual también demuestra que es una tecnología muy novedosa y que por eso resulta también muy atractiva para el público. La prueba de esto es que los voluntarios que han pasado las pruebas mostraban interés en llegar al puesto de la realidad virtual para poder probarlo por primera vez. Incluso, una persona que no venía a hacer las pruebas si no a acompañar a un voluntario quiso probar cómo era jugar a un juego en realidad virtual. Las preguntas realizadas en el cuestionario que los usuarios debían responder, valorando entre 0 y 5 siendo 0 muy poco y 5 mucho, son las que aparecen en la tabla 2 que muestra también la media de las respuestas de los seis voluntarios que respondieron la encuesta en las tres primeras sesiones de pruebas.

Tabla 2 Media de las respuestas al cuestionario posterior a las pruebas

Preguntas	**Respuesta Media**
¿Te has sentido desorientado?	0,3
¿Has sentido algún tipo de mareo que te haya hecho querer dejar de jugar?	0,5
¿Te ha resultado más entretenido jugar con las gafas que sin ellas?	3,6
¿Crees que jugarías durante más tiempo con las gafas que sin las gafas?	2,5
¿Estarías dispuesto a pagar más por poder jugar a este tipo de juegos en realidad virtual en lugar de una pantalla convencional?	3
Si pudieras jugar a tu juego favorito con unas gafas virtuales adaptadas a tu *Smartphone*, ¿lo harías?	4,2

De las primeras dos preguntas se puede obtener la conclusión de que a los voluntarios no les ha producido una sensación de desorientación, mareo o agobio que les supusiera un impedimento para seguir jugando. Esto es un hecho a destacar, ya que es el efecto más adverso que puede producir el visor de realidad virtual y sin embargo, a pesar de que los juegos no estaban siendo ejecutados a 60 fotogramas por segundo, que sería la situación ideal de funcionamiento para obtener una correcta experiencia de realidad virtual, si no a una tasa de fotogramas algo menor debido a limitaciones de hardware. Por otro lado se observa que, de forma general, a los voluntarios les ha parecido más entretenido jugar a los mismos juegos en un visor de realidad virtual que en una pantalla convencional, aunque esta conclusión puede haber sido producida en parte por la impresión de haber utilizado un visor de este tipo por primera vez, como se explica en el párrafo anterior. En cuanto al tiempo de juego, los voluntarios no han obtenido una sensación de que fueran a aumentar su tiempo de juego, lo cual puede deberse a que el uso del visor produce más fatiga visual que las pantallas convencionales. Se observa también que los voluntarios estaban dispuestos a asumir un sobrecoste por añadir la característica de la realidad virtual al sistema y que les parece una buena idea poder jugar a sus juegos favoritos en realidad virtual desde su *smartphone*.

A continuación de las preguntas que valoraban la experiencia se preguntaba acerca de si los voluntarios poseían un *smartphone* y a qué gama pertenecía el modelo que poseían considerando gama baja los modelos con un valor entre 100 y 250 Euros, gama media entre 250 y 400 Euros y gama alta de 400 a 700 Euros. Los resultados fueron que excepto un voluntario muy joven, el resto todos poseían un *smartphone* y la mayoría eran de gama media o alta. Esta es una información valiosa de cara a la continuación de este proyecto con la integración de los *smartphones* como pantallas del visor de realidad virtual. El

hecho de que en general los potenciales usuarios de la plataforma posean modelos de gama media o alta aumenta las probabilidades de que la experiencia de realidad virtual conseguida por cada usuario sea satisfactoria.

Por último había un espacio para comentarios generales acerca de la experiencia con el visor de realidad virtual. En general los comentarios fueron positivos y muy enfocados hacia la impresión tan particular que causa el uso de un dispositivo de este tipo por primera vez. Hubo un comentario acerca de lo que tenía que forzar la vista para utilizar el visor y el temor que esto le producía si lo utilizase durante periodos prolongados.

Las conclusiones en cuanto al uso del visor de realidad virtual son positivas, siendo así prácticamente todas las valoraciones de todos los voluntarios, manifestando una menor sensación de mareo de lo esperado. Lo aspectos principales en los que hay que mejorar son: primeramente, la optimización del juego y del hardware en el que se ejecuta para que se pueda ejecutar a 60 fotogramas por segundo, lo cual disminuirá la sensación de fatiga visual, mareo o desorientación del jugador; y segundo, la adecuación del control y el diseño del juego a la realidad virtual. Como ya se ha comentado en esta memoria, es importante que el juego haya sido diseñado teniendo en cuenta que va a ser jugado con un visor de realidad virtual para poder sacarle todo el potencial a esta característica. El ejemplo más claro se observa en el juego del avión. Siendo el juego que en principio puede parecer más adecuado para causar una gran impresión en el jugador mediante la realidad virtual. Un sistema de control inadecuado arruina la experiencia de juego del usuario, haciéndole querer dejar de jugar.

5.3 Pruebas con voluntarios no discapacitados

Una vez terminadas las sesiones de pruebas del sistema con voluntarios discapacitados se celebró una sesión de pruebas idéntica pero esta vez enfocada a usuarios sin problemas motrices. Todos los voluntarios que acudieron a esta sesión de pruebas eran muy jóvenes, entre 9 y 13 años de edad. Se trataba de voluntarios habituados a manejar videojuegos en su tiempo libre, ya fueran en una videoconsola o en un teléfono móvil inteligente. Las pruebas realizadas siguieron fielmente el protocolo elaborado para las sesiones con personas discapacitadas, probando por un lado el sistema MoKey y por otro lado el sistema Blexer comenzando por el puesto donde se lleva a cabo la adquisición de parámetros de movimiento del usuario. Se jugaba a los minijuegos en un monitor de ordenador convencional y después se pasaban al siguiente puesto donde se jugaba a esos mismos minijuegos pero en el visor de realidad virtual. Es importante destacar que al tratarse de voluntarios sin discapacidades físicas se eliminaban los problemas de detección que se habían presentado en las sesiones anteriores. El sensor Kinect detectaba a la perfección el esqueleto de los jugadores.

Los resultados fueron mejores en el control de los videojuegos que en las sesiones anteriores, no costándoles apenas esfuerzo realizar los ejercicios necesarios para progresar en los juegos y consiguiendo unas puntuaciones mucho más altas. La tabla 3 muestra una comparativa de la media de los resultados obtenidos por cada grupo de pruebas. Esto es un resultado lógico, que ya se esperaba antes de realizar las pruebas, pero que aporta información acerca de qué aspectos hay que mejorar cuando el jugador sí posee una discapacidad. Es necesario ajustar la tolerancia de los movimientos y la amplificación de los mismos con una mejor precisión para que aquellos jugadores con problemas motrices puedan alcanzar a jugar como lo han conseguido hacer los jugadores sin discapacidad. También se debe investigar la fórmula para conseguir una detección lo más correcta posible del esqueleto del jugador en el caso en el que éste no pueda moverse de una silla de ruedas.

Tabla 3 Media de las puntuaciones obtenidas en las pruebas

Juego	Discapacitados	Sanos
Escalera	75,4 s	48,1 s
Barca	100,6 s	70,8 s
Topos	9 topos	16,5 topos
Avión	0,4 anillos	7,1 anillos

En cuanto a las valoraciones de los voluntarios relacionadas con la experiencia del uso del visor de realidad virtual para jugar a los minijuegos, fueron muy relacionadas con la impresión que causa el probar un dispositivo de este tipo por primera vez. Todos los voluntarios comentaron la inmersión tan grande que produce el uso del visor y valoraron la experiencia como positiva. Sin embargo, a diferencia de cuando se hicieron pruebas con voluntarios discapacitados, esta vez sí mencionaron que tenían sensación de mareo y querían parar de jugar. Estos comentarios siempre se produjeron en el último minijuego, en el cual la cámara seguía a un avión que iba volando controlado por el jugador. Analizando las situaciones y comparando entre un tipo de jugador y otro es importante destacar lo siguiente: a los voluntarios con discapacidad les fue realmente complicado hacerse con el control del juego y que el avión actuara como ellos querían, y esto hizo que prefirieran dejar de jugar rápidamente, sin llegar a experimentar ninguna sensación de mareo. Por el contrario, los voluntarios sin problemas de movilidad consiguieron hacerse rápidamente con el control del avión y manejarlo con soltura, consiguiendo hasta tres voluntarios atravesar más de diez anillos en pocos minutos cuando en las sesiones con discapacitados nunca se habían cruzado más de cuatro. Esto se tradujo en mayor tiempo de juego. La diferencia entre este y otros juegos y la razón por la que este sí causaba más sensación de mareo es el movimiento de la cámara y el entorno. En este juego el mundo está continuamente moviéndose y el jugador debe controlar con la cabeza el campo de visión que posee. En otros juegos la cámara no se trasladaba, sólo rotaba en el punto fijo en el que se encontraba copiando las rotaciones que el jugador hacía con su cabeza. En conclusión, es muy importante que el diseño del juego sea realizado teniendo

en cuenta que éste va a ser jugado en realidad virtual, para que potencie esta característica al máximo y produzca al jugador la menor fatiga visual y disminuya al mínimo la posible sensación de mareo.

5.4 Conclusiones de las pruebas

Los resultados de las pruebas fueron positivos. Primeramente se obtuvo la aprobación de los movimientos necesarios para progresar en cada juego por parte de una fisioterapeuta experta en el área, quién además aportó información sobre movimientos útiles para futuros desarrollos de otros juegos de este tipo. La pruebas con gente discapacitada fueron positivas ya que, a pesar de las dificultades por problemas de detección o diseño de los juegos, los voluntarios pudieron en su mayoría completar las pruebas de forma satisfactoria y además se obtuvo información para realizar mejoras en el desarrollo. En cuanto a las valoraciones acerca del uso del visor de realidad virtual fueron mayoritariamente positivas, confirmando el valor que aporta esta característica al proyecto.

Dejando a un lado las valoraciones técnicas, se considera necesario mencionar que los comentarios recibidos de parte de los voluntarios que se sometieron a las pruebas acerca de todo el proyecto y la iniciativa de llevarlo a cabo fueron muy positivos, elogiando mucho la labor realizada en un campo donde muy pocos dedican su esfuerzo, que es el de las personas con discapacidad.

6. Conclusiones

Todo el trabajo llevado a cabo a lo largo de este proyecto, desde la fase de análisis y documentación, la fase de desarrollo y las pruebas realizadas, ha arrojado varias conclusiones acerca de la tecnología de realidad virtual y el valor que esta puede aportar al campo de los *exergames*.

Se han obtenido conclusiones del análisis de la realidad virtual realizado al inicio del proyecto. Toda la tecnología necesaria para que la realidad virtual pudiera funcionar de la forma correcta para ofrecer la experiencia que promete, alcanzó su punto de madurez hace pocos años. Es por esto que a pesar de que no se trata de una idea para nada nueva, es ahora cuando se ha podido implementar de forma adecuada, además de a un precio que la hace accesible al público.

Una de las conclusiones obtenidas de la fase de diseño e implementación ha sido que la enorme cantidad de recursos y herramientas para el desarrollo software que existen en la actualidad facilitan enormemente los desarrollos incluso para tecnologías tan novedosas como la realidad virtual.

Tras comparar la experiencia subjetiva de uso del visor de realidad virtual utilizado en el proyecto con la que ofrece un visor basado en *smartphone* se ha concluido que la

experiencia en un visor como Oculus es notablemente mejor, produciendo una inmersión más creíble para el usuario y menor fatiga visual. No obstante hay que destacar que los visores basados en *smartphone* ofrecen una experiencia muy buena si se tiene en cuenta el coste que tienen. Además del coste, cuenta con la ventaja de que no existe un cable que conecte el visor a otro dispositivo, mientras que los otros visores si lo tienen y a veces puede entorpecer el juego. En definitiva se trata de dos enfoques diferentes para lograr la sensación de realidad virtual, uno de bajo coste y con un resultado aceptable, y otra de un coste elevado que busca resultados óptimos.

Este proyecto ha acercado la realidad virtual a los *exergames*. Se partía de la hipótesis de que esta funcionalidad podía aportar valor al aumentar la sensación de inmersión en el juego y así ayudar a que el jugador se olvide de que está realizando ejercicio y se entretenga jugando. Los resultados obtenidos en las pruebas afirman que a los usuarios les ha resultado positiva la experiencia de jugar a este tipo de juegos en realidad virtual. No obstante no hay que dejar de lado el hecho de que la mayoría de ellos no habían probado esta tecnología antes y la impresión que causa su uso por primera vez pudo haber exagerado el valor que este sistema aporta realmente. Por tanto se concluye que sí es una funcionalidad que aporta valor a los *exergames*, siempre y cuando se cumplan los requisitos de correcto funcionamiento y diseño adecuado del juego, ya que si no se cumplen, esta funcionalidad puede resultar negativa en la experiencia del jugador. No obstante, hay que destacar que aunque sí aporta valor, no resulta una característica clave que potencie enormemente el objetivo de estos juegos, que es aumentar el tiempo y la motivación en las sesiones de rehabilitación.

7. Futuro trabajo

La realidad virtual está en auge. Hace tiempo que se vienen abriendo paso nuevas formas de interaccionar con los videojuegos fuera de los ya habituales mandos llenos de botones para buscar posibilidades en el movimiento del jugador como hizo la videoconsola Wii de Nintendo o el sensor Kinect de Microsoft. Ahora que la tecnología lo permite, la realidad virtual ha llegado para cambiar la forma de interaccionar con los videojuegos, dejando a un lado los monitores convencionales para convencer al jugador de que se encuentra en el mundo virtual del juego. Esto abre cantidad de puertas para desarrollos innovadores que se adapten a esta tecnología y expriman todo su potencial.

Este proyecto ha iniciado un primer acercamiento a esta tecnología para el proyecto de *exergames* desarrollado en el CITSEM pero aún queda mucho para llegar a aprovechar todo lo que la realidad virtual puede aportar. Así como el gran mercado está adaptándose a ella y va dando pasos hacia su adopción, muchos de estos pasos son aplicables a este proyecto. A continuación se describen diferentes áreas en las que se puede continuar el trabajo realizado durante este proyecto.

***Smartphone* como visor de realidad virtual**

Como ya se ha expuesto, aún no hay una solución en el mercado que ofrezca los resultados necesarios para obtener una experiencia de realidad virtual realmente inmersiva cuando se trata de un videojuego que se ejecuta sobre un PC. No obstante, el hardware contenido en los *smartphones* es suficiente como para conseguir un experiencia comparable a la generada por los visores de realidad virtual si el videojuego se está ejecutando en el propio dispositivo. Ya se han mencionado las ventajas que podría tener que este sistema funcionara con la misma fluidez, calidad de imagen y latencia si el juego se ejecutara en un ordenador que si se ejecutara en el dispositivo. Por todo esto y por la forma en la que se ha desarrollado la comunicación entre el *middleware* con OSVR y con Blender, dejando la puerta abierta a añadir el soporte de cualquier *smartphone* como visor de realidad virtual, el siguiente paso en este proyecto sería llevar a cabo este desarrollo. En el apartado 4.4 del presente documento analiza el sistema y describe una propuesta para enfrentar los problemas que este desarrollo pudiera plantear.

Contenidos adaptados a la realidad virtual

La estrategia comercial que han seguido los fabricantes pioneros en esta tecnología ha sido favorecer la creación de contenidos de valor adaptados a sus sistemas de realidad virtual antes incluso del lanzamiento al público de los propios sistemas. No habría tenido sentido poner a disposición del público un sistema de realidad virtual excelente y que ofrece una experiencia incomparable si no existiesen contenidos compatibles que los usuarios pudieran consumir en estos dispositivos. Es por eso que varios fabricantes han puesto en venta primeramente prototipos prematuros de los sistemas que estaban desarrollando pensados para que los desarrolladores puedan usarlos y generar contenidos para ellos.

Por el mismo motivo resulta imprescindible generar contenido compatible con el sistema desarrollado en este proyecto para que este pueda ser de utilidad real. Para poder llevar a cabo las pruebas descritas en el apartado 7 se han adaptado una serie de minijuegos desarrollados por otros alumnos en el CITSEM en el marco de los *exergames* para comparar con los voluntarios la experiencia de juego obtenida utilizando un mismo juego con el visor de realidad virtual y sin él. A pesar de que los minijuegos son perfectamente jugables con el visor y la experiencia de juego es agradable e inmersiva, no se trata de videojuegos desarrollados específicamente para ser jugados con un visor de realidad virtual y esto hace que no se aprovechen todas las posibilidades de este tipo de experiencia. Por tanto se propone la creación en un futuro de un videojuego específicamente desarrollado para ser utilizado en un visor de realidad virtual, sin dejar de lado el objetivo principal: que sirva a personas con movilidad reducida para llevar a cabo ejercicios de rehabilitación. De este modo, se recomienda buscar inspiración en los patrones de diseño seguidos en los nuevos videojuegos para realidad virtual que lleguen a ser éxito de ventas en los próximos años y en las guías de desarrollo propuestas por los fabricantes, haciendo uso de todas las buenas prácticas y recomendaciones que se pueden encontrar en estas. De este modo se puede conseguir un juego realmente inmersivo que

resulte entretenido jugar y que obligue al jugador a realizar los ejercicios necesarios para avanzar por el juego, así se conseguirá potenciar el objetivo de olvidar la sensación de estar haciendo rehabilitación y potenciar la sensación de estar jugando.

Modelos compatibles

Al ser una tecnología tan en auge, muchos fabricantes de tecnología están aún desarrollando sus dispositivos. Dada la rápida evolución que están experimentando todos los dispositivos electrónicos en la actualidad, es de esperar que en los próximos años se lancen modelos cada vez más potentes, precisos y baratos. Lo que hoy es una tecnología novedosa y cara, dentro de unos meses puede ser superada por otra mejor y más barata, tal y como ha evolucionado el mercado de los *smartphones* en cuestión de pocos años. Es por eso que se puede esperar que en el medio plazo surjan nuevos modelos de visor que pongan esta tecnología al alcance de cualquiera. La iniciativa OSVR en colaboración con el fabricante de periféricos de PC para aficionados a los videojuegos Razer, han diseñado un visor de realidad virtual con la misma filosofía que su entorno software de código libre. Han fabricado y lanzado al mercado (durante el desarrollo de este proyecto) un kit para desarrolladores a un coste reducido en comparación con otros fabricantes. Además han abierto el diseño para que cualquiera pudiera fabricarse su propio visor, publicando todos los esquemas electrónicos y mecánicos así como el *firmware* usado. Todo esto supone que en un futuro no muy lejano habrá dispositivos que cumplirán con uno de los requisitos de este proyecto que es ser lo más accesible para el público. Por este motivo se propone que con el lanzamiento de estos futuros dispositivos más asequibles se vaya añadiendo soporte a los más adecuados y populares de forma que el poder usar esta funcionalidad en el *middleware* suponga el menor sobrecoste posible.

Sonido 3D

Al cambiar de un modo radical la forma de presentar el mundo virtual al usuario, es necesario redefinir todas las interfaces de salida de datos que aportan información al jugador. La realidad virtual redefine la salida de vídeo que muestra la parte visual del videojuego pero es importante analizar como afecta este cambio a los sistemas de reproducción de sonido generado en el videojuego. Hasta la irrupción de la realidad virtual, todos los juegos se controlaban casi en su totalidad de una forma estática, generalmente sentados pero siempre teniendo como punto de referencia la pantalla que muestra los datos. La realidad virtual elimina esa referencia y obliga a que si el jugador quiere ver lo que su personaje en el videojuego tiene detrás, tenga que girarse para poder verlo, con lo cual el jugador cambia su posición durante el juego. Si el sonido del videojuego es emitido por altavoces estáticos situados en algún punto de la sala es fácil realizar el diseño sonoro desde un punto de vista técnico: igual que desde el punto de vista del jugador el entorno virtual que percibe permanece estático, las fuentes sonoras también permanecen estáticas. El problema es diferente si el sonido generado por el juego es emitido por auriculares, ya que estos se colocan fijos con respecto a la cabeza del

usuario y cambian de posición junto con esta, por tanto, en este caso, aunque el entorno virtual permanece estático para el jugador, las fuentes sonoras cambian de posición con sus movimientos. Por consiguiente, los algoritmos que calculan la mezcla de sonido que se emite por cada altavoz es diferente en el caso de usar auriculares que si se utilizan altavoces.

En el caso de utilizar altavoces existe bastante sitio para el desarrollo y la innovación ya que se pueden utilizar sistemas de sonido envolvente como el novedoso Dolby Atmos que consiguen que el usuario pueda percibir la localización de los eventos sonoros de una forma muy precisa. También es posible la implementación de algoritmos y efectos que hagan los sonidos más reales, modificando el sonido emitido en base a modelos psicoacústicos que potencien la localización de los sonidos o implementando efecto Doppler en fuentes sonoras en movimiento.

Por otro lado, en el caso del uso de auriculares para el juego, es imprescindible primero la implementación de algoritmos que calculen la mezcla emitida para cada oído del usuario en función de la posición de este con respecto a las fuentes sonoras en el juego. Del mismo modo que en el caso de utilizar altavoces, es posible la implementación de efectos que ayuden a la precisión en la localización de eventos sonoros y de efecto Doppler para fuentes en movimiento.

Con todas estas innovaciones se crearía un entorno de realidad virtual sonora que complementaría a la realidad virtual proporcionada por el visor para lograr una inmersión en el videojuego aún mayor.

8. Lista de referencias bibliográficas

[1] El Androide Libre, "2016 será por fin el año de la realidad virtual", 24 de Feb. 2016, http://www.elandroidelibre.com/2016/02/el-ano-en-el-que-la-realidad-virtual-dejo-de-ser-una-promesa.html. [Acceso: 26 de Jun. 2016].

[2] I. Gómez-Martinho González, "Interfaz de comunicación entre Blender y Kinect", Informe de prácticas, CITSEM, UPM, Madrid, España, 2015

[3] G. Burdea, P. Coiffet and P. Ducher, *Tecnologías de la realidad virtual*. Barcelona, España: Paidós Ibérica, 1996.

[4] M. MCGreevy, "The virtual environment display system", National Aeronautics and Space Administration, Moffett Field, CA, EE.UU. 1991.

[5] M. Zachara and J. Zagal, "Challenges for success in stereo gaming", *Proceedings of the International Conference on Advances in Computer Enterntainment Technology - ACE '09*, 2009.

[6] S. Parkin, "Oculus Rift", *MIT Technology Review*, 2016. [en línea] https://www.technologyreview.com/s/526531/oculus-rift/. [Acceso: 2 de Jun. 2016].

[7] J. Edison Muñoz, J. Felipe Villada and J. Giraldo Trujillo, "Exergames: una herramienta tecnológica para la actividad física", *Revista Médica de Risaralda*, vol. 19, no. 2, pp. 126-130, 2013.

[8] C. Kaminer, K. LeBras, J. McCall, T. Phan, P. Naud, M. Teodorescu and S. Kurniawan, "An immersive physical therapy game for stroke survivors", *Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility - ASSETS '14*, 2014.

[9] S. Finkelstein and E. Suma, "Astrojumper: Motivating Exercise with an Immersive Virtual Reality Exergame", *Presence: Teleoperators and Virtual Environments*, vol. 20, no. 1, pp. 78-92, 2011.

[10] I. Gómez-Matinho González, "Desarrollo e implementación de middleware entre Blender, Kinect y otros dispositivos", PFG, Dept. Teoría de la Señal y Comunicaciones, UPM, Madrid, España, 2016.

[11] M. Eckert, I. Gómez-Martinho, J. Meneses and J. Martínez Ortega, "A modular middleware approach for exergaming", *IEEE ICCE*, Berlin, 2016.

[12] "Introducción al OSC (Open Sound Control)", 3 de Ago. 2010, http://www.inventable.eu/2010/08/03/introduccion-al-osc-open-sound-control-primer-parte/. [Acceso: 15 de Jun. 2016].

[13] J. Ferguson, *La biblia de C♯*. Madrid: Anaya Multimedia, 2003.

[14] Microsoft Developer Network, "Kinect for Windows Sensor Components and Specifications", 2016, https://msdn.microsoft.com/en-us/library/jj131033.aspx. [Acceso: 2 de Jun. 2016].

[15] A. Hanson, *Visualizing quaternions*. San Francisco, CA, EE.UU.: Morgan Kaufmann, 2006.

[16] OSVR, "OSVR Developer Portal - Contributing", 4 de Jun. 2016, http://osvr.github.io/contributing/. [Acceso: 25 de Jun. 2016].

[17] The Apache Software Foundation"Apache License, Version 2.0", 2016, http://www.apache.org/licenses/LICENSE-2.0. [Acceso: 2 de Jun. 2016].

[18] OSVR, "BlendOSVR/OSVR-Blender", 2016, https://github.com/BlendOSVR/OSVR-Blender. [Acceso: 17 de Jun. 2016].

[19] I. Gómez-Martinho, "Estructura del middleware Chiro", Madrid, 2016.

[20] Jim Lindblom, "Bluetooth Basics - learn.sparkfun.com", 2016, https://learn.sparkfun.com/tutorials/bluetooth-basics/common-versions. [Acceso: 24 de Jun. 2016].

[21] Oculus VR, "Developer Center - Documentation and SDKs | Introduction to Best Practices", 2016, https://developer.oculus.com/documentation/intro-vr/latest/concepts/bp_intro/. [Acceso: 18 de Oct. 2015].

Anexo I - Manual de usuario

Manual del desarrollador de videojuegos

La realidad virtual es una interfaz de salida diferente a la que los usuarios tienen por costumbre utilizar en una videoconsola convencional y esto es una característica muy relevante a la hora de diseñar la jugabilidad de un videojuego. La compañía que ha sido pionera en esta tecnología, Oculus, ha publicado una guía de diseño [21] con recomendaciones y pautas para el diseño de videojuegos preparados para jugarse en realidad virtual. Es recomendable estudiar esta guía antes de acometer el diseño de un videojuego hecho para ser jugado con un visor de realidad virtual.

Para poder utilizar el *middleware* y desarrollar un videojuego en Blender Game Engine, primeramente, es necesaria la instalación de la extensión de nombre "HMD Receiver Camera" en Blender.

- Para realizar la instalación debe, primeramente, abrir el software Blender para después pinchar sobre el menú Archivo y seleccionar "*user preferences*".
- Se abrirá una ventana que permite configurar las preferencias de usuario del software, estando divididas en diferentes pestañas.
- Pinchando sobre la pestaña "*Add-ons*" se muestra un listado de las extensiones disponibles señalando cuales están instalados, cuales se pueden instalar y cuales están activos para su uso además de la información relacionada con cada extensión.

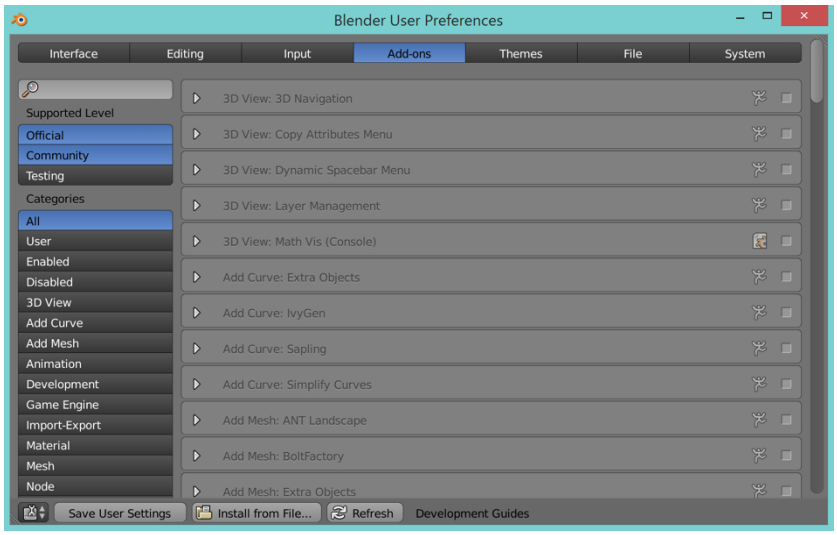

Figura 1 Ventana de instalación de extensiones

- Dado que la extensión que se pretende instalar no se encuentra en la lista, es necesario pulsar sobre el botón "*Install from File…*".
- Se abre una ventana que nos permite navegar por los diferentes directorios. Se debe buscar el directorio donde se encuentra el fichero "HMD.py", seleccionarlo y pulsar el botón "*Install from File…*".

- Por último, es necesario activar la extensión con la casilla de verificación que aparece a la derecha de la extensión que se acaba de instalar.

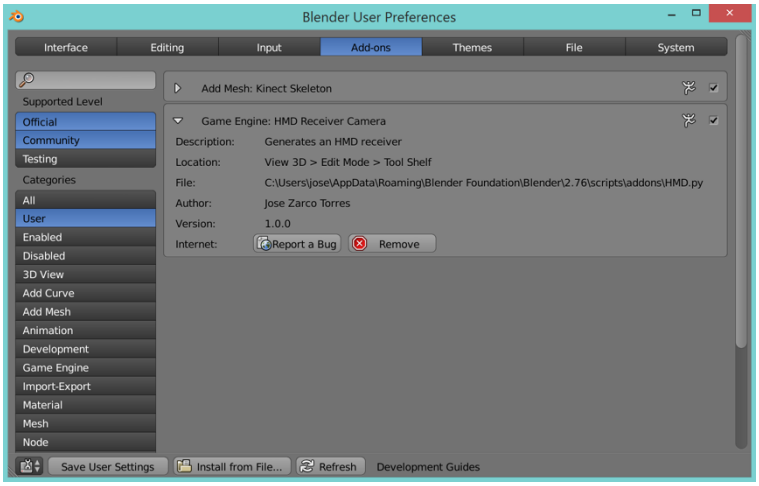

Figura 2 Extensión ya instalada

Esta extensión añade un botón con la etiqueta "*Add HMD Camera*" al menú "*Create*" del modo "*Object Mode*" que añade a la escena una cámara preparada para recibir y actuar ante los datos de posición del visor de realidad virtual.

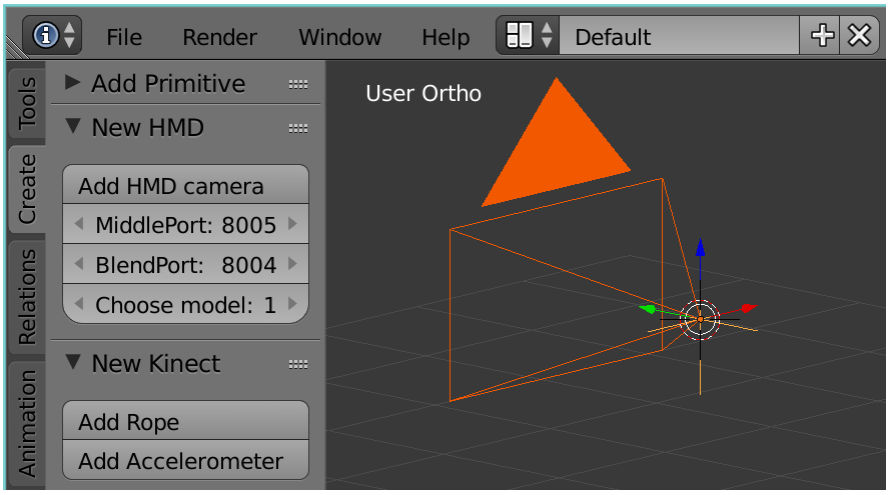

Figura 3 Botones añadidos a la interfaz al instalar la extensión y cámara generada al hacer clic sobre "Add HMD camera"

Cuando se añade la primera cámara de este tipo a la escena es necesario enlazar al objeto cámara el *script* que envía las peticiones de datos al *middleware* y recibe y procesa los resultados. Para ello es necesario seguir los siguientes pasos:

- Primero es necesario deseleccionar todos los objetos. Si algún objeto o conjunto de objetos se encuentra seleccionado se puede deseleccionar pulsando la tecla "A" del teclado.
- A continuación, se selecciona el objeto cámara, bien pulsando sobre él en la vista tridimensional con el clic derecho del ratón o bien seleccionándolo en la ventana del *outliner*.

- Después es necesario cambiar la vista de "*Default*" a "*Game Logic*".
- En la vista "*Game Logic*" se pueden observar los ladrillos lógicos asociados al objeto en la parte inferior del espacio de trabajo. En la parte derecha se encuentra el editor de texto, donde se debe pulsar sobre el botón "*Open*" para abrir un *script* existente desde un fichero.

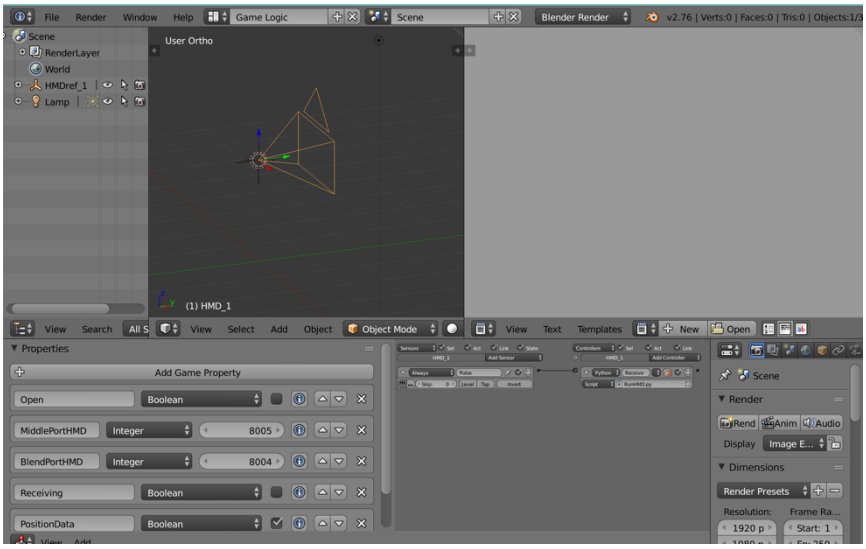

Figura 4 Vista "Game Logic" con el objeto cámara seleccionado

- A continuación, se navega hasta el directorio donde se encuentre el fichero "RunHMD.py", se selecciona y se pulsa sobre el botón "*Open Text Block*".
- Finalmente, en el ladrillo lógico controlador se selecciona el *script* "RunHMD.py".

Una vez creada la primera cámara, no es necesario abrir el *script* y enlazarlo cuando se creen más cámaras; éstas reutilizarán el *script* que se ha enlazado en la primera.

Cómo se puede observar en la figura 4 dónde se muestran las propiedades asociadas al objeto cámara, existe una propiedad de tipo *boolean* llamada "PositionData". Cuando esta variable tiene el valor *True* las peticiones que se mandan al *middleware* son de tipo "orientación y posición", mientras que si esta variable está con valor *False* se enviarán peticiones de tipo "sólo orientación". Cuando se utilice el modo "orientación y posición" es necesario tener en cuenta el valor de la variable "PositionSensibility". Este valor es el factor por el que se multiplican los datos recibidos desde el *middleware*, pudiendo variar la sensibilidad en el desplazamiento de la cámara según el valor de esta variable.

Tal y como se indican en las guías de desarrollo de Oculus, la forma óptima para utilizar este tipo de cámara es colocarla en el punto de vista del personaje principal, de modo que el mundo virtual se perciba en primera persona y fuera el usuario el protagonista del juego. La forma de hacer esto se resume en las siguientes indicaciones:

- Colocar el cursor 3D pinchando con el botón izquierdo del ratón sobre el lugar geométrico del punto de vista del protagonista del videojuego.

- Pinchar sobre el botón "*Add HMD Camera*" para que se cree la cámara.
- Seleccionar en el menú "*Outliner*" el objeto padre de la cámara, que es un objeto "*empty*". Seleccionar a continuación con el botón derecho sobre la malla del personaje protagonista a la que se desea emparentar la cámara (el tronco o el cuello) y con ambos seleccionados pulsar las teclas "Control" + "p" y pinchar sobre "*Object*".

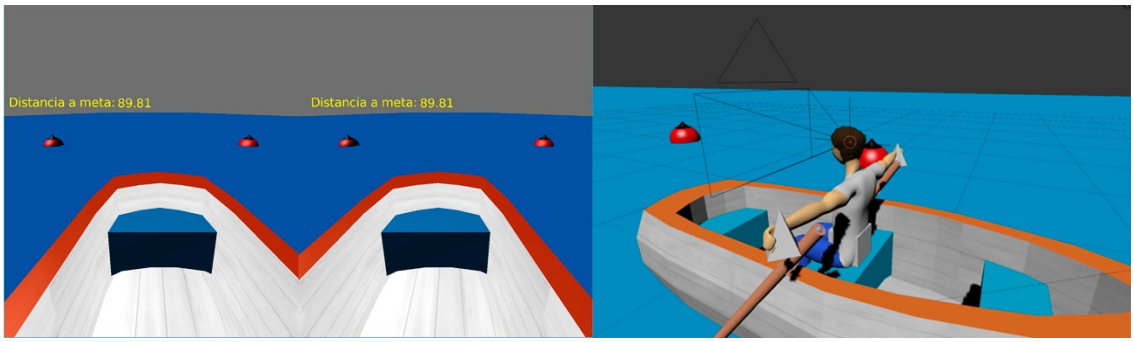

Figura 5 Ejemplo de juego adaptado para usar con HMD colocando una cámara en el punto de vista del avatar

Es necesario remarcar varios aspectos. Primeramente, aunque en una misma escena se puedan añadir varias cámaras de este tipo, es necesario que solo la cámara activa ejecute el *script* durante la ejecución. No obstante, es recomendable que toda la acción se lleve a cabo desde una misma cámara dado que los cambios de cámara en la realidad virtual pueden desorientar al usuario. También es importante tener en cuenta que los puertos seleccionados en las propiedades del objeto cámara sean siempre los mismos que los configurados en el *middleware*, de otro modo el sistema no funcionará de ningún modo.

Manual del desarrollador del *middleware*

El módulo desarrollado en este proyecto se integra dentro del *middleware* Chiro [10]. Este *middleware* gestiona la carga de una serie de pestañas que corresponden a los módulos que aportan funcionalidad al mismo. Cada módulo está desarrollado en un proyecto de Visual Studio independiente. Concretamente el módulo VR está desarrollado sobre Visual Studio 2012 en un proyecto llamado "VRForms". Al abrir este proyecto se presenta el código de la clase "VRWindow", que es la clase principal del proyecto. Todo el código se encuentra comentado, tanto la función que desempeña cada una de las variables declaradas al principio del código como la utilidad de cada función declarada. Las funciones más importantes del código son las siguientes:

- `Iniciarosvr()` – Arranca el servidor OSVR en un proceso a parte y con la configuración marcada en el desplegable de la interfaz gráfica. Para ello, arranca el ejecutable `osvr_server.exe` contenido en el directorio "`\VR\osvr_server\bin`", añadiendo como argumento el directorio "`\configs`" y el nombre del fichero JSON (*JavaScript Object Notation*) de configuración que contenga los parámetros para cada modelo de visor de realidad virtual. Si se quisiera añadir soporte a otro modelo, se

debe añadir el fichero correspondiente al directorio "`\configs`", se debe añadir una opción en el desplegable de la interfaz gráfica y se debe añadir un caso más al *switch* contenido en la función `cambioConfig()`.

- `IniciarVentana()` – Contiene toda la configuración de la interfaz gráfica de usuario. Esta es la función a modificar para añadir, quitar o modificar controles de la pestaña VR del *middleware*.

- `HiloVR()` – Esta función contiene la actualización de los datos del HMD. Se trata de una adaptación de la función de ejemplo que da OSVR para una aplicación cliente.

- `IniciarLector()` – En el código de esta función se define todo lo relacionado con la inicialización de los *sockets* para la comunicación entre Blender y el *middleware*. Uno de ellos es un *socket* asíncrono no bloqueante que escucha el puerto que se le ha configurado y lanza un evento cuando recibe un paquete. Este evento es el que provoca el envío de los datos del HMD encapsulados en un paquete OSC a través del otro *socket* inicializado en esta función.

El proyecto está configurado para tener como resultado de la compilación una biblioteca de clases y como versión de .NET Framework de destino la 4.5. Estos parámetros vienen dados por los requisitos del *middleware* para su correcta integración en él. Para acceder a estos parámetros hay que entrar en menú "Proyecto – Propiedades de VRForms…".

Para que el servidor OSVR pueda realizar la adquisición de datos, primeramente deben ser instalados los controladores del visor de realidad virtual. En la actualidad, el único visor compatible es el Oculus Rift DK2 y los controladores compatibles con el *middleware* corresponden al *runtime* ejecutable adjunto a este paquete de software. Se trata de la versión 1.7 del *runtime* proporcionado por el fabricante que implementa el SDK 0.6.0.1. Se han realizado pruebas con versiones posteriores del *runtime* obteniendo resultados negativos por lo que se recomienda utilizar la versión 1.7 para un funcionamiento óptimo.

Una vez instalados los controladores del visor a utilizar, se puede arrancar el *middleware*. Cuando este se inicia, se muestran diferentes pestañas, de entre las que se debe seleccionar la pestaña VR. En esta pestaña se puede configurar el modelo de visor de realidad virtual utilizado. Cuando se cambie el modelo a utilizar, es necesario pulsar sobre el botón "*Restart Server*" para que el cambio surta efecto. Este botón reinicia el servidor OSVR con la configuración seleccionada en el desplegable del modelo de visor. También es posible cambiar la configuración de los puertos para la comunicación con el juego, pero este parámetro solo debe ser cambiado por expresa indicación de cada juego, ya que por defecto los puertos de comunicación son los mostrados al iniciar el software. Si se

cambian los puertos, es necesario también pulsar en sobre el botón "*Change*" para que los cambios surtan efecto.

Finalmente, es importante destacar que, para el correcto funcionamiento de este software, debe utilizarse siempre en combinación con un videojuego desarrollado específicamente en Blender Game Engine para ser utilizado con este *middleware*, de otra forma el software no funcionará de modo alguno.