UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Sistema de visión estereoscópica altamente inmersiva

Presentado por Pablo Seoane Fuente
en Universidad de Burgos — 13 de junio de 2021
Tutor: Carlos Cambra Baseca
Co-Tutor: Nuño Basurto Hornillos
Tutor de empresa: Alejandro Langarica Aparicio

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Carlos Cambra Baseca, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Pablo Seoane Fuente, con DNI 71308759L, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado **Sistema de visión estereoscópica altamente inmersiva**.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 13 de junio de 2021

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

No es ningún secreto que la robótica es un campo en continua expansión y que cada vez atrae a más desarrolladores ya que el abanico de problemas que soluciona es amplio y con un alto grado de adaptabilidad y escalabilidad. A esto se debe que también existan innumerables desarrollos pensados para ser acoplados a proyectos ya existentes para satisfacer necesidades de cliente que no habían sido contempladas con anterioridad. De esta capacidad de compenetrar diferentes soluciones nace la idea de este proyecto: dotar a sistemas robóticos de una visión estereoscópica que sea capaz de conseguir sumergir al usuario en una situación lo más realista posible.

Abordando lo que se desea conseguir, y para ser más exactos, se busca crear un sistema de comunicación en el que un extremo retransmitirá vídeo a través de una cámara estereoscópica y en el otro se encontrará un equipo de realidad virtual capaz de reproducirlo y devolver coordenadas de rotación que serán interpretadas por la primera parte de esta conexión para, en un futuro, simular el movimiento de la cabeza de un ser humano. La idea es lograr una primera implementación a través de un servidor local, ampliando el rango de comunicación si los resultados obtenidos son satisfactorios.

Para este proyecto se han utilizado principalmente 2 herramientas: el entorno de desarrollo *software Unity* y los dispositivos de realidad virtual *HTC VIVE*.

Descriptores

Tecnología inmersiva, realidad virtual, Unity, robótica.

Abstract

It's not a secret that robotics field is in continuous expansion and that it attracts more and more developers since the range of problems it solves is wide and with a high degree of adaptability and scalability. This is why there are also innumerable developments designed to be coupled with existing projects to satisfy customer needs that had not been previously contemplated. The idea for this project was born from this ability to compenetrate different solutions: to provide robotic systems with a stereoscopic vision that is capable of immersing the user in a situation that is as realistic as possible.

Dealing with what we want to achieve, and to be more exact, it is desired to create a communication system in which one end will broadcast video through a stereoscopic camera and at the other end there will be a virtual reality equipment capable of reproducing it and returning coordinates rotation that will be interpreted by the first part of this connection to, in the future, simulate the movement of the human head.

Two main tools have been used for this project: the *Unity* software development environment and the *HTC VIVE* virtual reality devices.

Keywords

Inmersive technology, virtual reality, Unity, robotics.

Índice general

Índice de figuras

Introducción

No es ningún secreto en el mundo actual que las tecnologías relacionadas con la realidad mixta están experimentando un desarrollo notable a la par que un aumento en su uso, tanto de forma lúdica, como laboral. Siempre en busca de la innovación. Es por eso que, aquellos sectores en los que la relevancia de estar a la orden del día en el ámbito del I+D es imprescindible, quieren explorar y valorar el amplio abanico de posibilidades que ofrecen aquellas experiencias que permiten sacar partido a entornos virtuales o realidades aumentadas.

De este modo, sabiendo que la Realidad Mixta o híbrida está formada por la combinación de Realidad Virtual (RV) y Realidad Aumentada (RA), indagaremos un poco más en la primera puesto que es la que se trabajará de ahora en adelante en este proyecto. Más concretamente, en el trabajo conjunto de este tipo de entornos con cámaras estereoscópicas.

Para poder entender correctamente el por qué del uso de estos dos tipos de tecnología, primero debemos hablar del cerebro humano y su percepción del entorno a través de la visión. Y es que el órgano encargado de la actividad del sistema nervioso, es capaz de interpretar las diferentes imágenes recibidas a través de los ojos de manera conjunta en un proceso denominado convergencia, produciendo así el fenómeno de la estereopsis o percepción tridimensional. Es decir, el cerebro es capaz de percibir la profundidad del entorno debido a la recepción de imágenes dispares originadas por la diferente posición de ambos ojos en la cabeza.

Entonces, ¿Cómo pueden compenetrarse por un lado la realidad virtual y por otro la visión estereoscópica? Sencillo. La principal funcionalidad que encontramos en este tipo de cámaras es la generación de imágenes a través de cada una de las dos lentes. Es decir, generar imágenes de la misma manera

que lo hacen los ojos en el cuerpo humano. Si a esto le sumamos la aparición de otra gama de dispositivos que permite enviar diferentes imágenes a cada ojo, conseguimos reproducir de una manera similar (aunque con menor calidad) la visión humana a través de una cámara.

Estructura de la memoria

- **Introducción:** Descripción del contenido del trabajo y de la estructura de la memoria y del resto de materiales entregados.
- **Objetivos del proyecto:** Explicación de los objetivos que se persiguen con la realización del proyecto. Ya sean técnicos, personales, marcados por los requisitos, etc.
- **Conceptos teóricos:** Apartado dedicado a la explicación y desarrollo de una serie de conceptos que facilitan la lectura y comprensión de la memoria.
- **Técnicas y herramientas:** Presentación las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto.
- **Aspectos relevantes del desarrollo del proyecto:** Exposición de los aspectos más interesantes del desarrollo del proyecto, a modo de resumen de la experiencia práctica vivida a lo largo del mismo.
- **Trabajos relacionados:** Similar al estado del arte en una tesis. Da pie a comentar algunos trabajos ya realizados en el campo en el que se ubica el proyecto.
- **Conclusiones y líneas de trabajo futuras:** Informe crítico que presenta líneas de mejora del proyecto, líneas de continuación sobre el trabajo ya realizado y conclusiones derivadas del desarrollo realizado.

Anexos

- **Plan de Proyecto *software*:** Apartado que engloba información relativa a la gestión del proyecto.
- **Especificación de requisitos:** Descripción completa del comportamiento del sistema que se va a desarrollar.

- **Especificación de Diseño:** Realización del diseño *software* e interpretación del mismo.
- **Documentación técnica de programación:** Aspectos más relevantes del código desarrollado.
- **Documentación de usuario:** Manual de uso redactado de cara a la utilización del usuario.
- **Pruebas:** Pruebas realizadas.
- **Control de versiones:** Informe del control de versiones.

Objetivos del proyecto

A continuación, los objetivos serán categorizados debido a la diferencia de intereses que se denota dentro del proyecto. Presentando por un lado la parte empresarial, por otro la parte técnica y por último el ámbito personal. De esta manera haremos hincapié en lo que busca la empresa, lo que quieren cumplir los desarrolladores, y en este caso, lo que espero conseguir yo con el transcurrir del proyecto.

2.1. Objetivo empresarial

A nivel empresarial, se buscaba hacer frente a un proyecto de dimensiones notables que necesitaba de la cooperación de varios campos. Véanse robótica, programación, electrónica, visión y realidad virtual.

Es por esto que la tarea llevada a cabo sería una mezcla de 2 de estos campos, en concreto los de la programación y la realidad virtual. De tal manera que, como objetivo principal por parte de la empresa, se me propuso conseguir la comunicación por vídeo conferencia de 2 dispositivos, que además, permitiese a uno de los extremos de dicha conexión la recepción de las imágenes en un entorno de realidad virtual a modo de visión humana. Enumeraremos los siguientes objetivos entonces:

- Desarrollar un sistema de comunicación audiovisual eficiente entre 2 máquinas de una misma red local.
- Dotar a uno de los participantes del sistema de la capacidad de reproducir los elementos recibidos en un equipo de realidad virtual.

- Dotar al otro participante de la capacidad de enviar datos en forma de coordenadas ubicacionales/rotacionales.
- Establecer el punto de partida para que ITCL pueda continuar el desarrollo del proyecto a raíz de completar esta fase, consiguiendo así llegar a un proyecto comercial.

2.2. Objetivos técnicos

- Conectar 2 dispositivos de una red local creando un canal de intercambio de información a partir de la conexión lograda.
- Conseguir un envío óptimo de información en el canal de transmisión de datos, centrándome en el tránsito fluido de imágenes.
- Habilitar una bidireccionalidad que permita enviar y recibir tanto imágenes como otros datos.
- Conseguir la recepción de imágenes en tiempo real en los dispositivos de realidad virtual.

2.3. Objetivos personales

En cuanto a lo personal, indicaré tanto los objetivos iniciales como los que fueron surgiendo a raíz de desarrollar distintas partes del proyecto:

- Iniciarme en el desarrollo de proyectos dentro del mundo laboral.
- Explorar el campo de la realidad mixta (más virtual en este caso) pudiendo conocer todo lo que ofrece y si me resulta de interés o no.
- Lograr un manejo considerable de *Unity* puesto que es uno de los *frameworks* más utilizados, si no el más, en el mundo de los motores gráficos.
- Conocer y aprender a utilizar metodologías de trabajo ágiles.
- Conocer y aprender a utilizar herramientas de control de versiones.
- Conocer y aprender a utilizar herramientas de documentación, más concretamente LaTeX, que será la utilizada para la creación y desarrollo de la documentación pertinente.

Conceptos teóricos

Como punto de partida, es importante indicar que el desarrollo de todo el trabajo se hace interpretando el resultado como un videojuego. Por ello se trata de un ámbito que tiene grandes diferencias en cuanto a conceptos, técnicas y herramientas.

Debido a la diferencia que existe entre todo lo contemplado a lo largo de la carrera y el uso de un motor gráfico como *Unity* para desarrollo de realidad virtual, considero imprescindible hablar tanto del entorno de trabajo, como del campo para el que se lleva a cabo este proyecto.

3.1. Realidad virtual

Definiremos la realidad virtual[10] como el entorno de escenas y objetos de apariencia real —generado mediante tecnología informática— que crea en el usuario la sensación de estar inmerso en él. Dicho entorno es accesible gracias a la utilización de cascos o gafas de realidad virtual que son los encargados de reproducir de manera visual todo lo creado en un entorno de programación.

Sabiendo lo que es y cómo se consigue la interacción con este tipo de experiencia, cabe destacar algunos conceptos referentes a este campo:

Motores gráficos

Conjunto de rutinas y subprogramas que permiten renderizar, reproducir y dar movimiento a todos los modelos 3D o *sprites* (modelos 2D) que se pretenden mostrar en una pantalla[9].

Motores físicos

Expresión empleada en informática para referirse al *software* capaz de simular mediante programación ciertos sistemas físicos como la dinámica del cuerpo rígido, el movimiento de un fluido y la elasticidad. [45]

Motores de videojuego

También conocido como motor de juego. Definido como el conjunto de rutinas de programación que permiten el diseño, creación y funcionamiento de un videojuego. [44]

Típicamente están compuestos por un motor gráfico para renderizar gráficos 2D y 3D, un motor físico que simule las leyes de la física [40], animación, *scripting*, sonidos, inteligencia artificial, redes, retransmisión, gestión de memoria, escenarios gráficos y soporte para lenguaje por secuencia de comandos.

En nuestro caso, el motor de videojuegos que se ha utilizado es **Unity3D**[47] [30], plataforma de desarrollo que abordaremos más adelante dentro de este mismo apartado debido a la gran variedad de puntos en los que podemos indagar.

Dispositivos RV

Como ya hemos dicho anteriormente, para poder disfrutar de una experiencia inmersiva en el ámbito de la realidad virtual es necesario poseer o tener acceso a unas gafas o un casco RV. Al no tratarse de equipamiento de bajo coste, es imprescindible hacer un pequeño análisis pre-compra que nos permita hacer una buena elección del hardware que vamos a adquirir, y para ello actualmente se prioriza el cumplimiento de los siguientes criterios:

- Sistema operativo y características del terminal, puesto que influirán en la variedad de aplicaciones a las que pretendemos dar uso. Factor que influirá directa y proporcionalmente en el precio.
- Portabilidad del casco, de cara a tratarse de un dispositivo que se pretende mover o de un dispositivo que probablemente esté fijo. Además aquí se puede recalcar que hay dispositivos con mayor adaptabilidad a la variedad de espacios físicos en los que se pueden instalar.

En nuestro caso utilizaremos los dispositivos pertenecientes al paquete *HTC VIVE*, pero hablaremos de ellos más adelante, en el apartado de herramientas.

3.2. Unity 3D

Dentro de la descripción de *Unity* como concepto teórico, trataré varios términos que también podrían haber sido explicados en el punto anterior, pero he decidido incluirlos aquí debido a que mi trato con ellos ha sido única y exclusivamente desde esta herramienta. Por lo tanto aquí explicaré algunos elementos cuyo conocimiento es imprescindible en el desarrollo de un proyecto de realidad virtual con *Unity*, y más adelante, en el apartado correspondiente, abordaré *Unity* como herramienta.

FPS (Frames per second)

También conocidos como tasa de fotogramas[39][50] (o *framerate*), se define como la cantidad de imágenes consecutivas que se muestran por pantalla en cada segundo de vídeo.

Cuando estás viendo un vídeo, un videojuego, una película, etc. En realidad ves imágenes fijas consecutivas que pasan tan rápido que hacen captar al ser humano una sensación de movimiento constante. Esta velocidad de transición entre imágenes fijas está determinada por los FPS, que siempre que superen la cifra de 12, estipulada como máxima percepción de imágenes fijas por segundo que diferencia el cerebro humano, comenzará a generar sensación de movimiento constante.

En el mundo del desarrollo con motores de videojuego es importante un número considerable de FPS ya que no sólo se busca conseguir una reproducción visual fluida del contenido, sino que también hay ciertos factores de la programación acotados a los *frames*, tales como la periodicidad de algunas operaciones. Por esto va a ser un concepto a tener en cuenta durante el desarrollo del proyecto.[24]

Escena

Podemos definir las escenas[26] como cada uno de los niveles que compondrán nuestro programa o aplicación dentro de *Unity*. En ellos encontraremos todos los elementos que sean necesarios para la creación de las funcionalidades deseadas, de tal manera que veremos objetos simples, compuestos, cámaras, objetos vacíos que únicamente alberguen un *script*, planos, contenedores de interfaces...

GameObject

Los *GameObjects*[27] son objetos fundamentales en *Unity* que representan personajes, props, y el escenario. Estos no logran nada por sí mismos pero funcionan como contenedoras para *Components*, que implementan la verdadera funcionalidad.

Podemos diferenciar diferentes tipos de objetos:

- *GameObject* vacío: En primera instancia, un *GameObject* únicamente tiene alberga el componente *Transform*, el cual dota al objeto de posición y rotación dentro de la escena. Esto es considerado un *GameObject* vacío, puesto que únicamente contiene el componente que *Unity* impone como imprescindible.
- *GameObject* primitivo[29]: Son aquellos con forma predefinida por Unity, únicamente los mencionaremos: cubo, esfera, cápsula, cilindro, plano y *quad*.
- *GameObject* complejos: Nosotros mismos los definimos como complejos o compuestos, puesto que podremos encontrar objetos con 2 componentes, u objetos con un número sustancialmente grande de componentes, haciendo así que consigan la funcionalidad deseada por el desarrollador.

Cámara

Las cámaras son los elementos encargados de capturar y reproducir el entorno virtual al desarrollador. A través de la manipulación de las cámaras podremos adaptar el entorno a las diferentes necesidades de desarrollo.

Los objetos cámara vienen con una serie de propiedades implícitas que únicamente mostraremos:

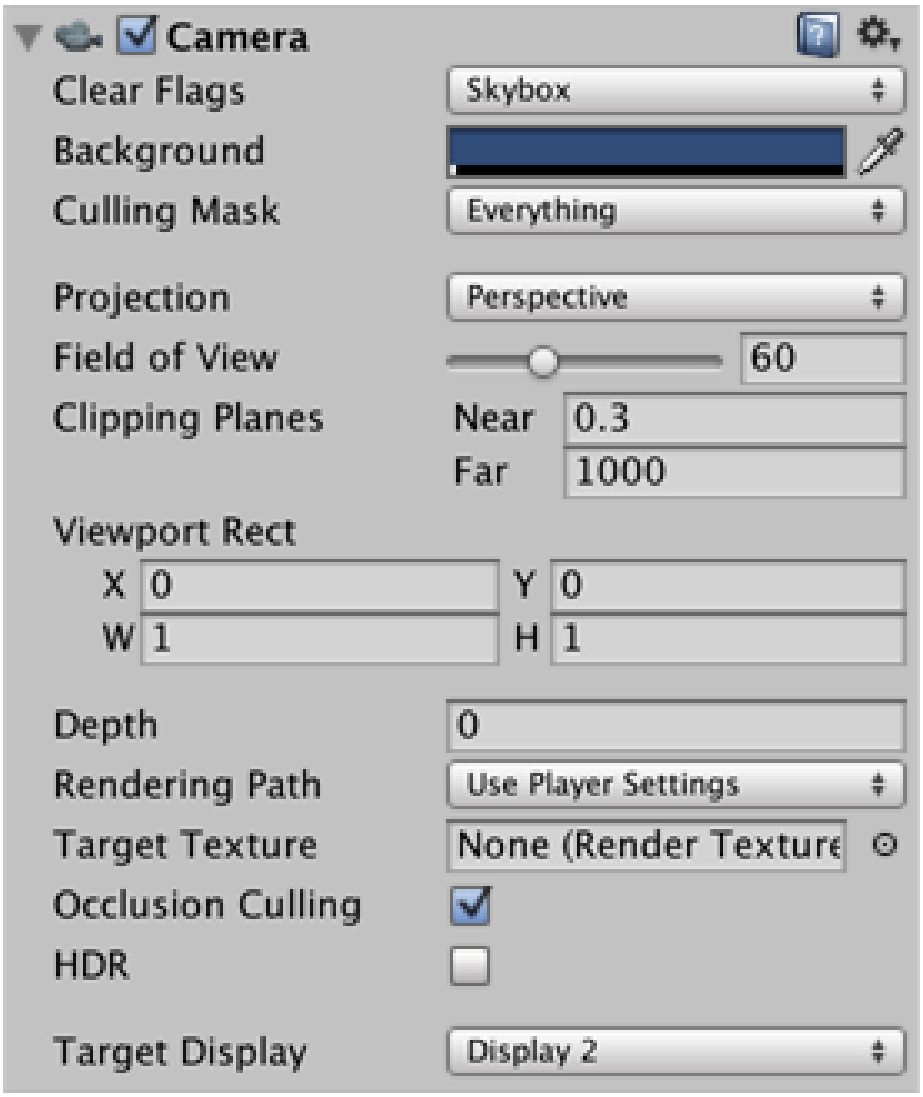

Figura 3.1: Imagen de las propiedades de la cámara de Unity

Realidad Virtual en Unity

La integración de la realidad virtual en *Unity* está muy trabajada, puesto que en este caso, únicamente hemos tenido que acceder a las opciones del proyecto y activar la selección que habilita la reproducción en dispositivos de realidad virtual.

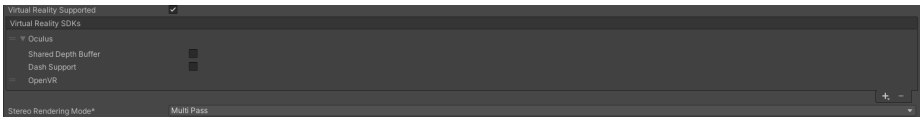

Figura 3.2: Imagen de la ventana de activación de Realidad Virtual en Unity

Por otro lado, cabe destacar el uso de un *plugin* que será profundizado en el apartado de herramientas y que facilita la configuración de cámara, controladores y escenario.

Técnicas y herramientas

A continuación comentaremos algunas de las técnicas y herramientas utilizadas durante el proyecto, explicando sus aspectos más relevantes y los porqués de las diferentes elecciones. De igual manera indicaremos algunas de los elementos descartados. En los anexos, tanto en el D: Manual del programador como en el E: Manual de usuario, muchas de las herramientas serán comentadas en referencia a su instalación y uso.

4.1. Metodologías ágiles

Desde ITCL se propuso el seguimiento del proyecto mediante la adaptación del trabajo al desarrollo software a través de metodologías ágiles[38][42]. Estas técnicas (como *Scrum*[1]), obviamente, no son del todo aplicables puesto que están pensadas para trabajos colaborativos o en equipo. Aún así, se tomó la decisión de que el tutor empresarial hiciera de mánager y desempeñara las siguientes labores:

- Programación de la duración de cada sprint.
- Consejo sobre las posibles tareas a completar en cada sprint.
- Reuniones periódicas para revisión y posibles correcciones del trabajo.
- Proponer cualquier tipo de modificación, consiguiendo así simular peticiones de cliente.

4.2. Herramienta de gestión de proyecto

Antes de hablar sobre la herramienta de gestión de proyecto, hay que mentar que desde la Universidad se invita al alumno a llevar a cabo una gestión y almacenamiento de las tareas realizadas/ a realizar en un repositorio. En este caso se eligió *GitHub* como repositorio puesto que de las aconsejadas por la UBU, me pareció la más extendida e intuitiva.

Como herramienta de gestión se valoraron diferentes posibilidades, pero tras elegir *GitHub*[7] como repositorio por ser el único útil de este tipo visto durante la carrera, también se recurrió a la que consideré, era la herramienta con mayor grado de integración en este repositorio, y además, también había sido utilizada durante los estudios del grado, *ZenHub*[51].

4.3. Control de versiones

Un control de versiones[36][3][2] es un sistema que registra modificaciones en archivos o conjuntos de archivos a lo largo del tiempo. De modo que cuando el usuario desee, podrá establecer una versión, y así, si en algún momento es necesario, volver a ella a pesar de haber realizado modificaciones posteriores.

En el mundo de la informática se hace normalmente (y en este proyecto) para tener controlados los cambios que se ejecutan en un programa o aplicación.

GitKraken

En ITCL la herramienta que se utiliza actualmente, y a la que como estudiantes, y gracias a la licencia profesional que se nos proporciona por serlo, tenemos también acceso, es *GitKraken*[8].

Gitkraken cuenta con una interfaz amigable y con integraciones de *GitHub* que hacen de uso algo simple e intuitivo.

Alojamiento del repositorio

El alojamiento del repositorio donde se encuentra el proyecto real es un servidor privado de ITCL al que no se tiene acceso a no ser que sea desde la red local, o desde la VPN que tiene configurada.

4.4. Motor de videojuego: Unity3d

En lo referente a la elección de motores gráficos (o de videojuego) no hubo ninguna duda. A pesar del creciente potencial que está desarrollando el que considero es, el mayor rival o competidor de *Unity*, *Unreal Engine*, decantarse por el motor de *Unity Technologies* fue sencillo.

Tras haber explicado ciertos conceptos que nos ayudan a conocer algunos de los elementos más sencillos, pero a su vez más importantes de esta herramienta de desarrollo, hablaremos de la forma de utilizar *Unity* como herramienta. Ya que cuenta con la fama de ser una aplicación de fácil uso, repleto de atajos para facilitar lo que podrían ser labores algo más minuciosas, y además, debido a la enorme comunidad de usuarios que posee, cuenta también con infinidad de foros, tutoriales, experiencias previas de usuarios y opiniones que hacen que muchas de las complicaciones que me pudiera ir encontrando, estuvieran dentro de casuísticas que habrían sido problemas para otros y probablemente haber sido resueltos por expertos o usuarios con mayor grado de experiencia.

A todo esto hay que añadirle que ITCL dispone de elementos que me podrían proporcionar ciertas soluciones a diferentes problemáticas ya que posee una licencia Pro de *Unity* y una serie de *plugins* de pago que, llegado el caso, resultarían de utilidad.

4.5. Entorno de programación

En este apartado de herramientas y técnicas hablaremos de 2 IDE (*Integrated Development Enviroment*) ya que desarrollé la mitad del proyecto con uno, y la otra mitad con el otro.

Visual Studio

Este entorno de desarrollo es propiedad de *Microsoft* y es compatible con una variedad amplia tanto de lenguajes de programación, como de entornos de desarrollo web[48][14].

Es el IDE propio de *Unity*, ya que te aconseja su instalación a la par que la principal. Fue el utilizado durante la primera parte del proyecto.

Rider

Como segunda opción hablamos de *Rider*[11], un entorno de *Jetbrains* creado en base a otros IDEs inspirados en IntelliJ que se integró con *Unity* en 2017, y al que también tenemos acceso gracias a la licencia pro de estudiante que se nos concede.

Durante la segunda parte del proyecto fue el encargado de albergar toda la programación del proyecto, y he de agradecer a uno de los trabajadores de la empresa el consejo ya que, bajo mi punto de vista se trata de un entorno mucho más adaptable al gusto del programador.

4.6. Node.js y NodeDss

A la hora de realizar la conexión local entre 2 dispositivos nos basamos un en nodo que haría de servidor(*NodeDSS*), pero para poder ejecutarlo, era necesario un entorno extra(*NodeJs*).

Node.js

Node.js[12] es un entorno de tiempo de ejecución de *JavaScript* (de ahí su terminación en . js haciendo alusión al lenguaje *JavaScript*). Este entorno de tiempo de ejecución en tiempo real incluye todo lo que se necesita para ejecutar un programa escrito en *JavaScript*.

NodeDSS

NodeDSS es un señalizador implementado por *Microsoft* que permite conseguir conexiones entre pares (peer-to-peer/P2P) a través de un servidor *Node.js*. El único inconveniente de cara a futuras implementaciones es que sólo autentica el otro extremo de la conexión a través del reconocimiento de su *deviceID*.

4.7. OpenVR

OpenVR es la integración de *Unity* que permite al entorno renderizar todo el contenido en dispositivos de realidad virtual. A veces, si un dispositivo no tiene su propia integración en *Unity*, es necesario conectarlo con *SteamVR* ya que unifica la detección de ciertos dispositivos y sí es reconocido por *OpenVR*.

4.8. Plugins de Unity

Los *plugins*[31] de *Unity* nos permiten introducir código externo a nuestra aplicación. En función de la herramienta de creación del código que se quiere importar, encontraremos *Managed Plugins* o *Native Plugins*.

WebRTC

El *plugin* de *WebRTC* es una integración de código que nos iba a permitir trabajar con un canal de datos que habilita una conexión de transmisión de imágenes fluida.

JetBrains Rider Editor

Plugin que permite la comunicación de *Unity* con el editor de scripts *Rider* creado por *JetBrains*.

Visual Studio Code Editor

Plugin que permite la comunicación de *Unity* con el editor de scripts *VisualStudio* creado por *Microsoft*.

TextMesh Pro

Plugin potente y fácil de usar(también conocido como TMP), que utiliza técnicas avanzadas de representación de texto junto con un conjunto de sombreadores personalizados; ofrece mejoras sustanciales en la calidad visual al tiempo que brinda a los usuarios una flexibilidad increíble en lo que respecta al estilo y textura del texto. *Unity* lo instala por defecto en los proyectos de la versión utilizada.

Unity UI

Plugin para desarrollar interfaces de usuario para juegos y aplicaciones. Es un sistema de interfaz de usuario basado en *GameObject* que utiliza componentes y la vista de juego para organizar, colocar y diseñar interfaces de usuario. *Unity* lo instala por defecto en los proyectos cuando se crea un *GameObject* de tipo UI.

OpenVR Desktop

Plugin[23] que permite activar el soporte para crear y compilar aplicaciones para dispositivos soportados por *SteamVR*.

4.9. SteamVR

SteamVR[32][35] es la plataforma que permite a ciertos dispositivos realizar las configuraciones de utilización necesarias para que el usuario experimente situaciones de realidad virtual. Además tiene incluidas funciones de reconocimiento con otros programas para ampliar así el abanico de posibilidades que la RV ofrece.

4.10. Hardware RV

A la hora de hablar de los dispositivos de realidad virtual utilizados, únicamente hablaremos sobre los dos que utilicé durante el desarrollo. El único motivo por el que se probaron varios es para comprobar si la integración de uno u otro era más complicada de cara a elegir para futuros proyectos.

HTC VIVE

El modelo de HTC utilizado fue el *HTC VIVE* de 2016[41][33]. Con las siguientes especificaciones:

- Frecuencia de actualización de 90 Hz, pantallas con resolución de 1080x1022, más de 70 sensores y área máxima de actuación de 4.6mx4.6m.
- Acompañadas de mandos que hacen la función de controladores.
- Es necesario instalar unas estaciones que recrean un espacio digital a partir del espacio físico que abarcan.

Oculus Quest 2

Este modelo perteneciente a *Facebook* está fechado en 2020[6]. Con las siguientes especificaciones:

- 6GB de RAM y procesador *Qualcomm® Snapdragon™ XR2*, pantallas con resolución de 1832 x 1920-

- También viene acompañado de mandos que realizan la función de controlador.
- No necesita instalaciones puesto que utilizando los mandos y a través de los sensores del casco se elige la zona a recrear de manera digital según lo deseado por el usuario.

4.11. Assets de Unity

Antes de explicar algunas de las herramientas descartadas hablaremos sobre los *assets*, ya que varias de ellas lo son.

Un *asset*[28] de *Unity* es un elemento que puedes usar en tu juego o Proyecto. Un *asset* puede provenir de un archivo creado fuera de *Unity*, como un modelo 3D, un archivo de audio, una imagen o cualquiera de los otros tipos de archivo compatibles con *Unity*. Existen también algunos tipos de *assets* que puedes crear dentro de *Unity*, como *Animator Controller, Audio Mixer o Render Texture.*

4.12. Técnicas o herramientas descartadas

En esta sección, hablaré de cuál iba a ser la funcionalidad de cada herramienta, y de porqué finalmente no fue seleccionada.

Azure

Azure[13] es un servicio de *Microsoft* que ofrece infinidad de posibilidades. Su primer motivo de uso iba a ser el de albergar el servidor al que se conectarían los dos extremos de la retransmisión audiovisual, pero debido a la complejidad de la programación y al hecho de que tiene un costo según el tiempo de uso, se prefirió no optar por su compra.

Photon Unity Networking

Photon Engine[18] es un *asset* de *Unity* que facilita la creación de servidores de conexión multijugador. Está pensado para albergar varios jugadores en escenas de juego, o *chats* escritos para interaccionar durante las diferentes sesiones. Finalmente se descartó porque tenía un enfoque dirigido a videojuegos, además de requerir cierto tiempo de estudio hasta dominarlo. Como extra, tenía funciones de pago.

FMETP Stream

FMETP Stream[22] es un asset específico dedicado a añadir funciones de retransmisión en vivo a las aplicaciones de Unity. Se descartó por su precio.

Aspectos relevantes del desarrollo del proyecto

En este apartado se tratará de indicar los aspectos que más dificultades generaron en las diferentes fases del proyecto: planteamiento, diseño, desarrollo, codificación, creación, compilación, etc.

Es posible que se trate de un apartado muy variado y que muchos de sus elementos no tengan una conexión implícita entre sí, pero todos han sido necesarios para llevar a cabo la consecución del proyecto de manera satisfactoria.

5.1. Conectividad entre 2 máquinas

Uno de los primeros problemas que surgió fue el planteamiento y elección del método de conexión a realizar. Además de estar presente desde muy temprano, también mantuvo su presencia durante gran parte del proyecto, ya que de manera transversal al resto de tareas, el tipo de conexión muchas veces generaba cuellos de botella en otras labores relevantes, como por ejemplo, el envío y recepción de imágenes.

De esta forma, y bajo mi punto de vista, el método de conexión a implementar entre 2 ordenadores es un trabajo que requiere bastante tiempo de estudio puesto que, de no estar muy familiarizado con las herramientas (que son las que limitan las posibilidades), se necesita una adaptación considerable al abanico de opciones que permiten los distintos programas que se vayan a utilizar. La información acerca de cómo se conectarán finalmente las dos máquinas está explicada en el anexos D: Manual del programador, más concretamente en el apartado de instalación y ejecución del proyecto.

En este caso en concreto, el número de versiones que fui realizando con distintos tipos de conexiones para el intercambios de datos fue considerable, y por ello me centraré en los principales protocolos que traté de implementar:

Modelo TCP/IP

Este modelo[43] es una descripción de protocolos utilizados para asentar las guías generales de operación que permiten que un equipo pueda comunicarse dentro de una red. Provee conectividad de extremo a extremo indicando formato, dirección, transmisión, enrutamiento y recepción de datos.

Dado que *Unity* contiene elementos pertenecientes a este tipo de protocolos para establecer conexiones servidor-cliente, es fácil encontrar ejemplos prácticos en los que basarse para amoldar las funcionalidades ofrecidas por el programa y adaptarlas a las necesidades del usuario. Este fue el principal motivo por el que se decidió utilizar este modelo en primera instancia.

Tras varias pruebas con los elementos implementados en *Unity*, se llegó a una conclusión determinante, y es que a pesar de tener ya implícito en la conexión un protocolo de confirmación de recepción de datos (lo cual es muy útil para no recibir información en desorden), fue imposible optimizar el envío de imágenes lo suficiente para que se consiguiera una retransmisión fluida de 2 ráfagas de imágenes diferentes, ya que como el propio proyecto pide, cada cámara debía mandar las imágenes captadas a un ojo. Por esto mismo se buscó una implementación basada en el siguiente protocolo a explicar.

Protocolo UDP

Este modelo[46] basado en el intercambio de datagramas[37] hace que la conexión previa no sea necesaria, puesto que los datagramas ya llevan en su cabecera la información de direccionamiento.

El problema de este protocolo llegó cuando me di cuenta de que carecía de control de flujo y confirmación, lo cuál indicaba que los paquetes de información enviados, a pesar de tener una velocidad de transmisión mayor (factor que solucionaba el tema del rendimiento de FPS), podían generar trabas, ya que algunas imágenes no llegaban íntegras, o lo hacían en desorden.

Tras evaluar con el personal de la empresa la posibilidad de implementar un código de control de flujo, se decidió dejar a un lado el UDP y optar por el último protocolo del que hablaremos.

Protocolo WEB RTC

Es el protocolo[34] [49] elegido para el desarrollo de este proyecto, el cuál permite añadir capacidades de comunicación en tiempo real a una aplicación. Admite vídeo, voz y datos genéricos que se envían entre pares, lo que permite a los desarrolladores crear potentes soluciones de comunicación de voz y vídeo.

Se trata de una tecnología implementada sobre un estándar abierto, por lo que los desarrolladores tienen la posibilidad de generar soluciones óptimas y potentes de comunicaciones audiovisuales. Además, está respaldado por grandes marcas como *Apple*, *Google* o *Microsoft*.

A nivel personal, la idea de utilizar este protocolo viene dada por la inspiración producida por una aplicación de uso diario como es *Discord*[4]. Ya que la funcionalidad que ofrece es bastante óptima y tras conocer que su implementación está basada en esta solución de estándar abierto, la empresa me ha dado el visto bueno y hemos decidido probarlo.

5.2. Optimización de imágenes

La optimización de imágenes fue un punto complicado de afrontar, ya que inicialmente ni siquiera valoré la posibilidad de que fuera a suponerme problemas, puesto que la retransmisión de una sola imagen era fluida desde la primera prueba con el protocolo TCP/IP.

Una vez se comenzaron a enviar 2 secuencias de imágenes diferentes por el mismo *socket*, comenzó el problema, ya que los *frames* por segundo descendieron notablemente. Y a pesar de acabar solucionando esto modificando la codificación de las imágenes haciendo que su extensión fuera .PNG en vez de .JPG, surgió otro problema, el *delay* de la retransmisión, que no sólo era notable, sino que aumentaba con el tiempo debido a la acumulación de imágenes previas en la recepción de datos del cliente.

De esta manera, y tras recurrir a 2 protocolos diferentes, se acabó consiguiendo una retransmisión en tiempo real que no generaba ninguna de las problemáticas anteriores.

5.3. Ajuste de imágenes a la escena

Otro asunto importante a resolver fue la adaptación de las imágenes que recibía el cliente de cada cámara del *host* para que éstas se reprodujeran

en cada ojo respectivamente. En los anexos C: Especificación de diseño y D: Manual del programador se podrá apreciar cómo están organizadas las escenas para que esto se consiga.

Para llevar a cabo esta tarea primero hubo que lidiar con el uso de elementos *Canvas*[25] en las diferentes escenas de *Unity*, pudiendo así organizar la disposición de las mismas tanto para la reproducción de la aplicación en pantalla, como para la reproducción en las gafas de realidad virtual.

La solución desarrollada finalmente consistió en hacer que los planos donde se reproducían las imágenes fueran hijos cada uno de una cámara, añadiéndoles unos componentes que los obligaban a adaptarse a la resolución de la pantalla.

A continuación mostraremos el inspector de componentes de cada uno de los 3 elementos que conforman la estructura de reproducción de imágenes, siendo el padre la cámara, que tiene como hijo al *Canvas* y este a su vez tiene otro hijo que es el plano donde se representan las imágenes:

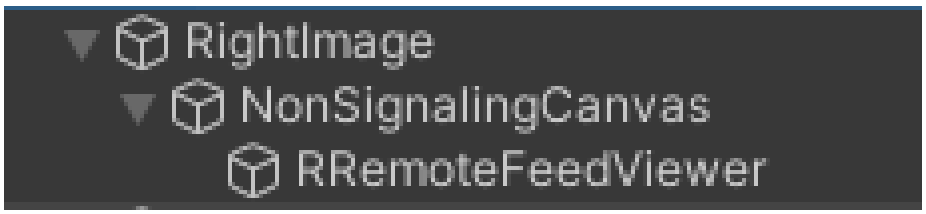

Figura 5.3: Estructura de los componentes de la imagen

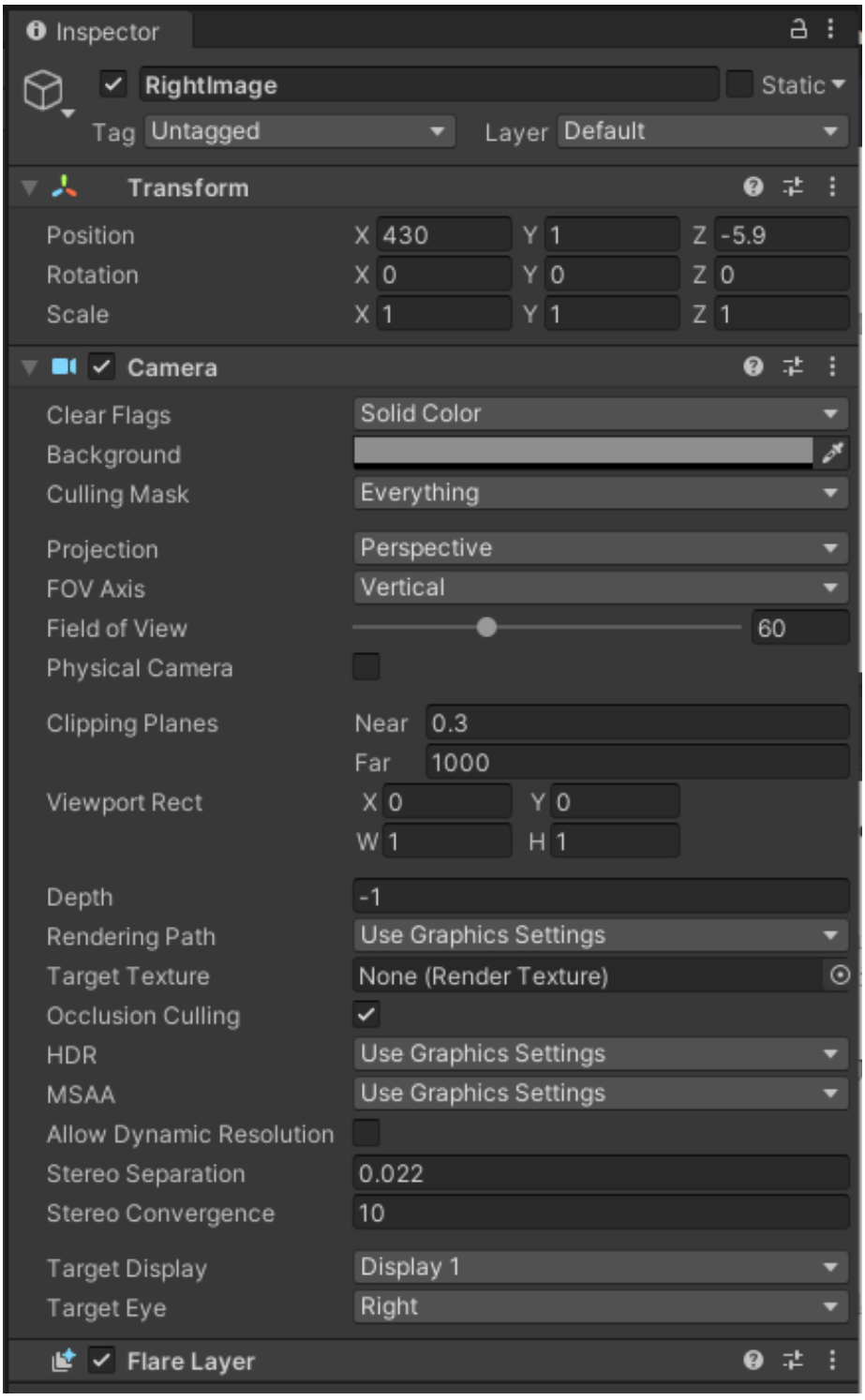

Figura 5.4: Inspector de los componentes de la cámara

A continuación en el inspector del *Canvas* podremos ver que su renderización funciona acorde al campo de visión de la cámara (*Render Mode ->Screen Space - Camera*) y que el método de escalado de sus elementos corresponde al tamaño de la pantalla (*UI Scale Mode ->Scale With Screen Size*).

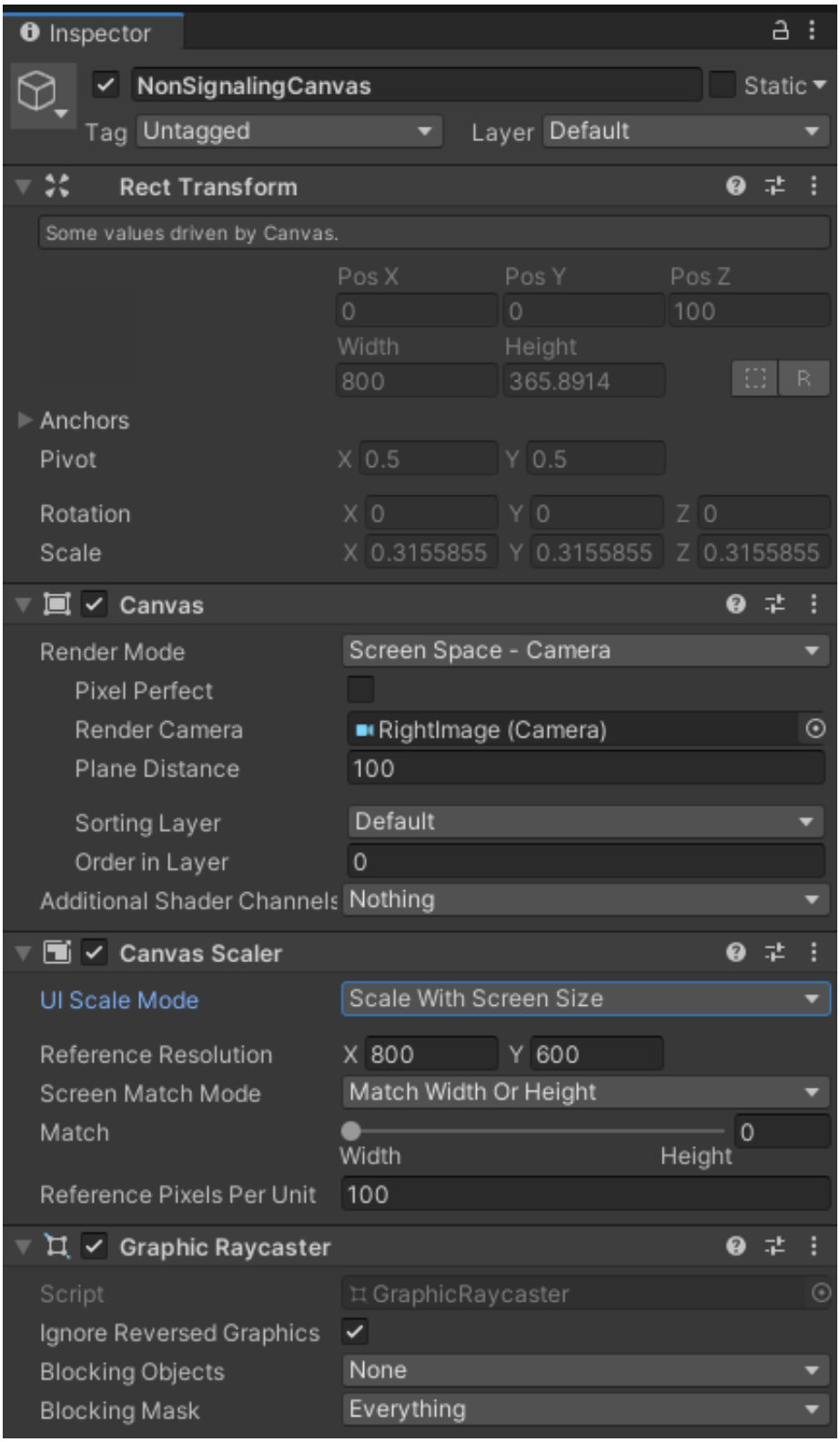

Figura 5.5: Inspector de los componentes del *Canvas*

5.4. Ajuste de cámara a la realidad virtual

Una vez conseguido el ajuste de imágenes dentro de la escena, se llegaba al conflicto de una correcta reproducción de la rentransmisión audiovisual del *host* desde las gafas de RV situadas en el lado del cliente. Tras indagar un poco en las funcionalidades que ofrece el editor de *Unity*, se acabó encontrando una opción que permite seleccionar dónde queremos que se reproduzca lo que ve la cámara: en la parte derecha, en la izquierda, o en ambas. Simplemente había que asignar a una cámara la reproducción para el lado derecho (ojo derecho) y a la otra cámara la reproducción para el lado izquierdo (ojo izquierdo).

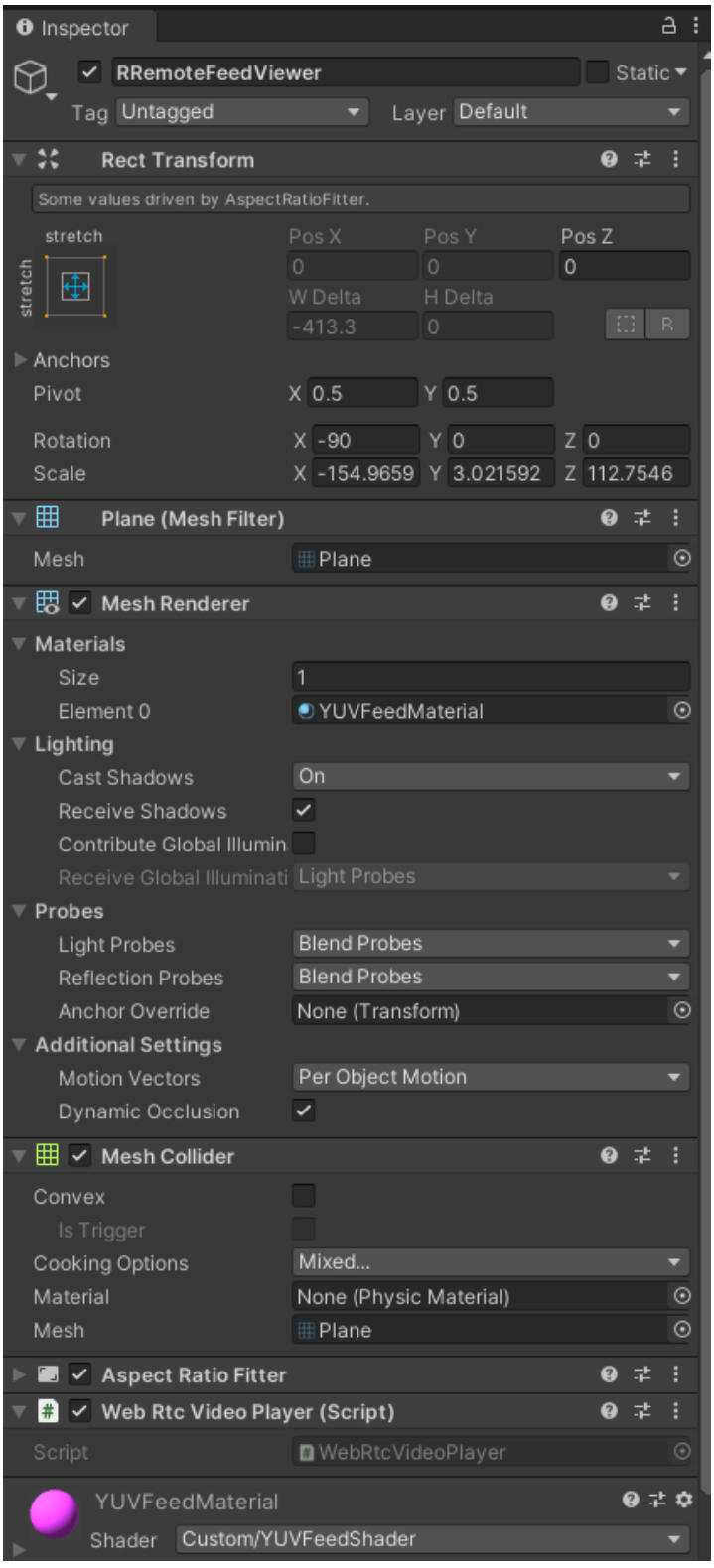

Figura 5.6: Inspector de los componentes del plano

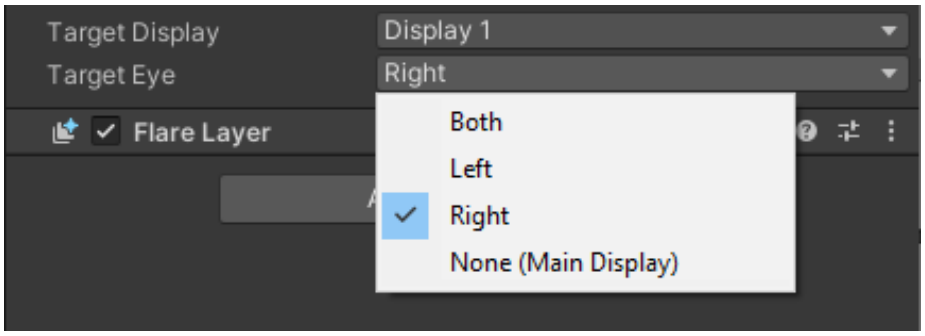

Figura 5.7: Selector de opciones para elegir pantalla de las gafas RV queremos enviar la imagen

5.5. Forma de obtener coordenadas

Conseguir recibir en tiempo real las coordenadas tanto de posición como de rotación del casco RV fue otra de las tareas a tener en cuenta en este proyecto.

Para la obtención de coordenadas se valoraron 2 posibilidades: cogerlas directamente del componente *Transform* del objeto cámara que adquiere las propiedades de las gafas de realidad virtual en tiempo de ejecución, o mediante una serie de funciones implementadas en *Unity* que que calculan las coordenadas en el espacio real que generan las estaciones en la habitación en la que se encuentre el usuario.

Finalmente se optó por la selección de obtención a través del componente *Transform*, debido a que facilitaría futuras operaciones llegada la hora de tener que valorar reseteos de posición, rotación, guardados de puntos fijos, etc.

Trabajos relacionados

En cuanto a los trabajos relacionados, desde la empresa se me dio a conocer que, por un lado, contábamos con escasas referencias anteriores de gente o empresas que hubieran intentado mezclar el uso de visión estereoscópica con realidad virtual, y por otro, que la idea no venía de una propuesta de proyecto por parte de otra empresa o de un proyecto europeo, sino que pertenecía a un empleado de ITCL, más concretamente a mi tutor empresarial.

Por esto mismo hablaremos de otros proyectos que han inspirado el lanzamiento de este:

Stereolabs

Stereolabs[21] es una empresa que cuenta con avances punteros en el campo de la visión, puesto que se dedican a la producción de cámaras estereoscópicas en distintos formatos según las necesidades del cliente, y también a la implementación de software en distintas plataformas para así poder sacar el máximo rendimiento a sus productos.

Empresas de robótica

Generalizaremos un poco debido a que son varias las empresas mencionadas como punto de observación para la propuesta de nuestra idea, y es que compañías como *Boston Dynamics*[5], *Robotnik*[20] o *ASTI*[19] dan a conocer al mundo el potencial que la robótica posee y el abanico de posibilidades que ofrece, por esto, otro de los propósitos de este sistema de visión fue la posibilidad de acoplarlo en un futuro a otras soluciones robóticas si fuera posible.

OMRON Collaborative

Omron[15] es una empresa de innovación segmentada en 4 ramas, dónde destacaremos la industrial[17], ya que otro de los motivos por los que el sistema de visión llegó a ser considerado como una propuesta viable fue la futura colaboración con brazos articulados, en concreto, de esta empresa, ya que ITCL posee uno de proyectos anteriores.

El brazo al que se le pretendía dar utilidad y que yo tuve la suerte de manipular ligeramente fue el *OMRON TM Collaborative*[16].

Conclusiones y Líneas de trabajo futuras

Este sistema de visión estereoscópica es la fase inicial de un proyecto cuya complejidad requiere de la intervención de especialistas en distintos campos para poder ser llevado a cabo.

Aún así, gracias a la gran cantidad de información que internet ofrece a día de hoy, esta primera tarea ha podido ser desarrollada por un único participante, que, a pesar de haberlo hecho en lo que considero es un plazo de tiempo muy amplio para lo que un cliente podría exigir, ha podido llegar a una serie de conclusiones técnicas y personales que a su vez han derivado en en unas posibles líneas de trabajo futuras para la mejora de lo implementado hasta el momento.

7.1. Conclusiones técnicas del proyecto

En lo referente a las conclusiones técnicas del proyecto, enumeraré una serie de factores que considero imprescindible nombrar:

- **Control de versiones:** Es imprescindible el control de versiones en proyectos que dotan al empleado de una carga de trabajo notable para evitar errores fatales.

- **Diseño gráfico:** El diseño gráfico es una parte de este proyecto, que a pesar de no resultar complicada en cuánto a lógica, es laboriosa dado que requiere de más tiempo del esperado.

- **Conectividad:** Resulta realmente complicada la consecución de una buena conectividad teniendo que conocer primero todas las peculiaridades de una red local si esta tiene un tamaño considerable.
- **Optimización:** Tanto en la realización de operaciones como en la transmisión de información, es un factor determinante que en este caso ha requerido varios cambios estructurales en el proyecto.
- **Investigación:** Las áreas de desarrollo requieren invertir mucho tiempo en investigar y recabar información, y a veces no tanto en ponerla en práctica. Como ha ocurrido con este proyecto en ciertos momentos del desarrollo.

7.2. Líneas de trabajo futuras

Está claro que al ser un proyecto desarrollado por una persona sin experiencia hay muchos elementos con amplia capacidad de mejora, y más si a esto le añadimos que se trata de un prototipo. Por ello hablaremos de los siguientes aspectos como posibles mejoras de cara al futuro:

- **Seguridad de la conexión:** Hasta el momento la única seguridad que tiene la conexión entre los dos extremos del sistema de visión es verificar que el ID de un lado corresponde con lo que busca el otro. Este es un aspecto que, de llegar a utilizarse como producto, considero que requeriría de un mayor grado de seguridad a través de distintos factores de autenticación de la conexión.
- **Diseño gráfico:** El diseño gráfico de la interfaz de la aplicación, a pesar de, como ya he dicho, necesitar más tiempo del esperado, en un futuro podría ser un trabajo para delegar a un departamento de arte o diseño 3D.
- **Optimización:** Otro punto mentado en las conclusiones técnicas, que considero, no tiene límite de mejora puesto que es probable que gente con un mayor grado de conocimiento sea capaz de lograr una transmisión de datos (tanto imágenes como coordenadas) con una velocidad e integridad de la información mayores a las conseguidas por mí.
- **Añadido de funcionalidades:** Actualmente el sistema es únicamente de visión, pero en un futuro, se podría intentar implementar un sistema de sonido por ejemplo, o de visión artificial, etc.

7.3. Conclusiones personales

En cuanto a las conclusiones personales haré referencia a las sensaciones o aspectos propios que creo haber descubierto, fortalecido o identificado como debilidad durante el desarrollo del proyecto.

- He descubierto el mundo de la realidad virtual en persona, y no sólo a través de vídeos en distintas plataformas o de noticias.
- He podido conocer *Unity* como herramienta de trabajo, pero también como *hobby* o pasatiempo, ya que el desarrollo en esta plataforma puede implicar trabajo pero también entretenimiento.
- Agradezco profundamente haber tenido acceso a ciertos aspectos del mundo laboral, al día a día de un trabajador y no tanto de un estudiante, y al inicio en el mundo empresarial que tanto ITCL como sus empleados me han proporcionado.
- Por otro lado, he conocido la dificultad de trabajar en el mundo del desarrollo, puesto que al tratarse muchas veces de campos desconocidos o sin tratamiento anterior, los momentos de estudio e investigación pueden ser algo más ásperos que los trabajo puramente práctico.
- A nivel personal he sido capaz de fortalecer la capacidad de organización, punto que nunca he considerado una fortaleza.
- Me ha sido dada una visión fundamental (bajo mi punto de vista) de cuáles son los factores principales de cara a la producción de elementos orientados a necesidades del cliente, que en la universidad no son tratados con la importancia que se merecen.

Bibliografía

[1] Encarna Abellán. Scrum: qué es y cómo funciona esta metodología. https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html, 2020.

[2] Scott Chacon. Acerca del control de versiones. https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones, 2014.

[3] Scott Chacon. *Pro Git*. Apress, 2014.

[4] Discord. Página oficial de discord: Discord | imagina un lugar. https://discord.com/, 2021.

[5] Boston Dynamics. Página oficial de boston dynamics. https://www.bostondynamics.com/, 2021.

[6] Facebook. Oculus quest 2. https://www.oculus.com/quest-2/, 2021.

[7] GitHub. Página oficial de github. https://github.com/, 2021.

[8] GitKraken. Página oficial de gitkraken. https://www.gitkraken.com/, 2021.

[9] Iberdrola. Motores gráficos y de juego: definición, tipos y modelos de negocio. https://www.hyperhype.es/motores-graficos-y-de-juego-definicion-tipos-y-modelos-de-negocio/, 2019.

[10] Iberdrola. Realidad virtual, la tecnología del futuro. https://www.iberdrola.com/innovacion/realidad-virtual, 2019.

[11] JetBrains. Página oficial de rider. https://www.jetbrains.com/es-es/rider/, 2021.

[12] Open JSFoundation. Página oficial de node.js. https://nodejs.org/es/, 2021.

[13] Microsoft. Microsoft azure. https://azure.microsoft.com/es-es/, 2021.

[14] Microsoft. Página oficial de visual studio. https://visualstudio.microsoft.com/es/, 2021.

[15] Omron. Omron españa. https://omron.es/es/home, 2021.

[16] Omron. Robots colaborativos | omron, españa. https://industrial.omron.es/es/products/collaborative-robots#, 2021.

[17] Omron. Robótica | omron, españa. https://omron.es/es/home, 2021.

[18] PhotonEngine. Photon unity networking. https://www.photonengine.com/PUN, 2021.

[19] ASTI Mobile Robotics. Robotics company | agvs and amrs | asti mobile robotics. https://www.astimobilerobotics.com/, 2021.

[20] Robotnik. Página oficial de robotnik - robótica y manipulación móvil. https://robotnik.eu/es/, 2021.

[21] Stereolabs. Stereolabs - capture the world in 3d. https://www.stereolabs.com/, 2021.

[22] FMETP Stream. Fmetp stream asset. https://assetstore.unity.com/packages/templates/packs/fmetp-stream-143080, 2021.

[23] Unity. About openvr (desktop). https://docs.unity3d.com/Packages/com.unity.xr.openvr.standalone@2.0/manual/index.html, 2021.

[24] Unity. Administrador del tiempo y framerate. https://docs.unity3d.com/es/530/Manual/TimeFrameManagement.html, 2021.

[25] Unity. Canvas. https://docs.unity3d.com/es/2019.4/Manual/UICanvas.html, 2021.

[26] Unity. Escena. https://docs.unity3d.com/es/530/Manual/CreatingScenes.html, 2021.

[27] Unity. GameObject. https://docs.unity3d.com/es/2018.4/Manual/class-GameObject.html, 2021.

[28] Unity. Guía rápida para la asset store de unity. https://unity3d.com/es/quick-guide-to-unity-asset-store, 2021.

[29] Unity. Objetos primitivos. https://docs.unity3d.com/es/530/Manual/PrimitiveObjects.html, 2021.

[30] Unity. Pagina oficial de unity. https://unity.com/es, 2021.

[31] Unity. Plugins. https://docs.unity3d.com/es/530/Manual/Plugins.html, 2021.

[32] Valve. Página oficial de steamvr. https://www.steamvr.com/es/, 2021.

[33] VIVE. Vive support. https://www.vive.com/eu/support/vive/, 2021.

[34] WebRTC. Página oficial de webrtc. https://webrtc.org/, 2021.

[35] Wikipedia. Steam vr — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Steam_VR, 2019.

[36] Wikipedia. Control de versiones — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Control_de_versiones, 2021.

[37] Wikipedia. Datagrama — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Datagrama, 2021.

[38] Wikipedia. Desarrollo ágil de software — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Desarrollo_ágil_de_software, 2021.

[39] Wikipedia. Fps — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Fotogramas_por_segundo, 2021.

[40] Wikipedia. Física de juego — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Física_de_juego, 2021.

[41] Wikipedia. Htc vive — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/HTC_Vive, 2021.

[42] Wikipedia. Metodologías ágiles — wikipedia, la enciclopedia libre. https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/, 2021.

[43] Wikipedia. Modelo tcp/ip — wikipedia, la enciclopedia libre. https://https://es.wikipedia.org/wiki/Modelo_TCP/IP, 2021.

[44] Wikipedia. Motor de videojuego — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Motor_de_videojuego, 2021.

[45] Wikipedia. Motor físico — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Motor_físico, 2021.

[46] Wikipedia. Protocolo de datagramas de usuario — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Protocolo_de_datagramas_de_usuario, 2021.

[47] Wikipedia. Unity — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego), 2021.

[48] Wikipedia. Visual studio — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Microsoft_Visual_Studio, 2021.

[49] Wikipedia. Webrtc — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/WebRTC, 2021.

[50] Yúbal Fernández, Xataka. Qué son los fps o fotogramas por segundo, y para qué sirven en los videojuegos. https://www.xataka.com/basics/que-fps-fotogramas-segundo-sirven-videojuegos, 2020.

[51] ZenHub. Página oficial de zenhub. https://www.zenhub.com/, 2021.