



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Sistema de visión
estereoscópica altarmenta
inmersiva
Documentación Técnica**



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

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	8
Apéndice B Especificación de Requisitos	13
B.1. Introducción	13
B.2. Objetivos generales	13
B.3. Catalogo de requisitos	14
B.4. Especificación de requisitos	16
Apéndice C Especificación de diseño	25
C.1. Introducción	25
C.2. Diseño de datos	25
C.3. Diseño arquitectónico	26
C.4. Diseño de interfaces	27
Apéndice D Documentación técnica de programación	35
D.1. Introducción	35
D.2. Estructura de directorios	35
D.3. Manual del programador	36

D.4. Instalación y ejecución del proyecto	42
D.5. Builds del proyecto	43
D.6. Pruebas del sistema	46
Apéndice E Documentación de usuario	47
E.1. Introducción	47
E.2. Requisitos de usuarios	47
E.3. Instalación	48
E.4. Manual del usuario	50
Bibliografía	59

Índice de figuras

A.1. Gráfica Burndown del Sprint 0	2
A.2. Gráfica Burndown del Sprint 1	3
A.3. Gráfica Burndown del Sprint 2	3
A.4. Gráfica Burndown del Sprint 3	4
A.5. Gráfica Burndown del Sprint 4	5
A.6. Gráfica Burndown del Sprint 5	5
A.7. Gráfica Burndown del Sprint 6	6
A.8. Gráfica Burndown del Sprint 7	7
A.9. Gráfica Burndown del Sprint 8	7
 B.1. Diagrama de casos de uso	 16
 C.1. Diseño arquitectónico	 26
C.2. Diseño de pantallas	27
C.3. Escena inicial de la aplicación	28
C.4. Inspector Menu	29
C.5. Escena con un pequeño manual de funcionamiento para el usuario	29
C.6. Inspector UserManual	30
C.7. Escena del emisor de imágenes	30
C.8. Inspector Host	31
C.9. Escena del receptor de imágenes	32
C.10. Inspector Client	33
 D.1. Aplicación Unity Hub	 37
D.2. Configuración del entorno de programación	38
D.3. Configuración del entorno de programación	39
D.4. Página de descarga de Node.js	40
D.5. Instalador de Node.js	40

D.6. Instalación y primera ejecución de NodeDSS	41
D.7. Añadir proyectos a Unity Hub	42
D.8. Entorno de desarrollo Unity	44
D.9. Ventana Build Settings	45
E.1. Configuración del espacio de interacción	50
E.2. Escena inicial de la aplicación: Menú	51
E.3. Escena con un pequeño manual de funcionamiento para el usuario	52
E.4. Escena del receptor de imágenes	53
E.5. Vista con el botón HideButtons activado	54
E.6. Vista con el botón EnableVr activado	54
E.7. Vista con el botón ExternalCam activado	55
E.8. Vista de dos imágenes diferentes con el botón ExternalCam activado	55
E.9. Escena del emisor de imágenes	56
E.10. Escena del emisor de imágenes en el framework	57
E.11. Escena del emisor de imágenes en funcionamiento	57

Índice de tablas

A.1. Costes de personal	8
A.2. Costes hardware	9
A.3. Costes software	9
A.4. Costes de personal	9
A.5. Costes de personal	11
B.1. RF-1.1: Ejecutar el NodeDSS para que hagas las funciones de servidor local.	17
B.2. RF-2.1: Conseguir una conexión entre dos máquinas.	17
B.3. RF-2.2: Conseguir el envío y recepción de imágenes a través de la conexión.	18
B.4. RF-2.2.1: Reconocer la webcam en Unity.	18
B.5. RF-2.2.2: Mostrar la imagen de la webcam en Unity.	19
B.6. RF-2.2.3: Enviar la imagen.	19
B.7. RF-2.2.4: Recibir la imagen.	20
B.8. RF-2.3: Conseguir el envío de coordenadas a través de la conexión.	20
B.9. RF-2.3.1: Tomar las coordenadas de rotación de las gafas de realidad virtual.	21
B.10. RF-2.3.2: Enviar las coordenadas.	21
B.11. RF-2.3.3: Recibir las coordenadas.	22
B.12. RF-2.4: Reproducir las imágenes recibidas en un dispositivo de realidad virtual.	22
B.13. RF-2.4.: Reconocer el dispositivo en <i>Unity</i>	23
B.14. RF-2.4.2: Enviar cada conjunto de imágenes a un ojo.	23
D.1. Requisitos mínimos de HTC VIVE	42
E.1. Requisitos mínimos de HTC VIVE	48

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado trataré de dejar claros los procedimientos organizativos para realizar el desarrollo, y las tareas concretas, con fechas concretas y mecanismos concretos de control y seguimiento.

A.2. Planificación temporal

Este proyecto se ha dividido en *sprints*, medida propia de la metodología *SCRUM*[1] mencionada ya en la memoria. La duración de cada sprint es de 5 días de trabajo (lunes-viernes), con retoques mínimos en los días de fin de semana (sábado y domingo). Por esto, el proyecto cuenta con una duración de 10 *sprints*.

Estos *sprints*, a su vez, tendrán asignadas diferentes tareas que cuentan con un sistema de puntuación que evalúa la complejidad y el tiempo estimado necesario de las mismas. Este sistema consta de los siguientes valores: 1,2,3,5,8,13,21,40. Siendo 1 el valor que implica menor dificultad y tiempo y 40 la magnitud que implica la estimación máxima de ambos.

En cada uno de los subapartados de la planificación temporal se exponen las tareas planeadas para ese sprint y una imagen con una gráfica *textitBurndown*[22] que muestra el transcurso de la realización o cierre de las mismas.

En ciertos *sprints*, la necesidad de acabar las tareas en tiempo hizo que se cerrasen varias (sino todas) de golpe, por ello hay imágenes en las que la gráfica es un único escalón.

Sprints realizados

Sprint 0 (08/02/2021-12/02/2021)

- Elección de repositorio
- Elección de la herramienta de gestión del proyecto
- Elección de herramientas de desarrollo
- Elección del entorno de desarrollo integrado (IDE)
- Elección de la herramienta de documentación
- Toma de contacto con *Unity*
- Establecer los objetivos principales del proyecto

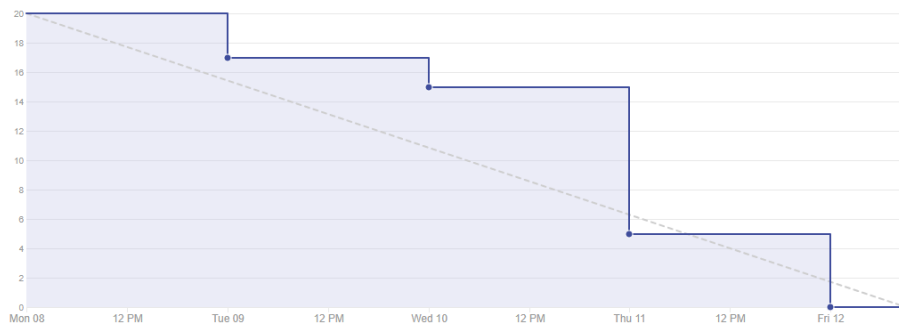


Figura A.1: Gráfica Burndown del Sprint 0

Sprint 1 (15/02/2021-19/02/2021)

- Instalación del sistema de Realidad Virtual *HTC VIVE*
- Documentar las funcionalidades del sistema *HTC VIVE*
- Utilización de RV en *Unity*

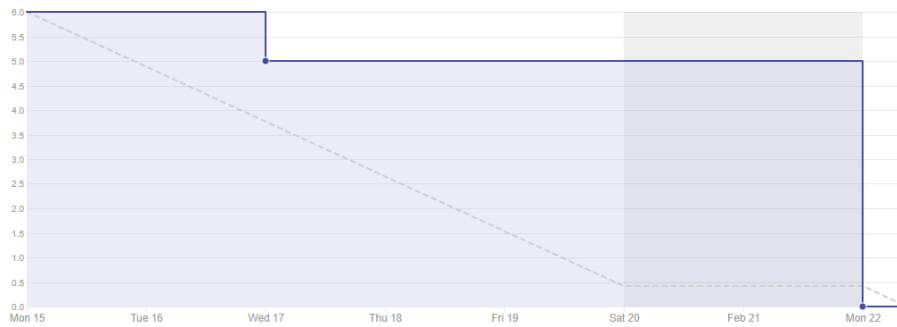


Figura A.2: Gráfica Burndown del Sprint 1

Sprint 2 (22/02/2021-26/02/2021)

- Comenzar el tutorial de RV + *HTC VIVE*
- Documentar en *GitHub* los objetivos principales del proyecto
- Recopilar información acerca de cámaras estereoscópicas
- Estudio del protocolo TCP/IP para la transmisión de imágenes en *Unity*
- Reconocer *WebCam* en *Unity*
- Investigar el uso de *WebCamTexture* en *Unity*

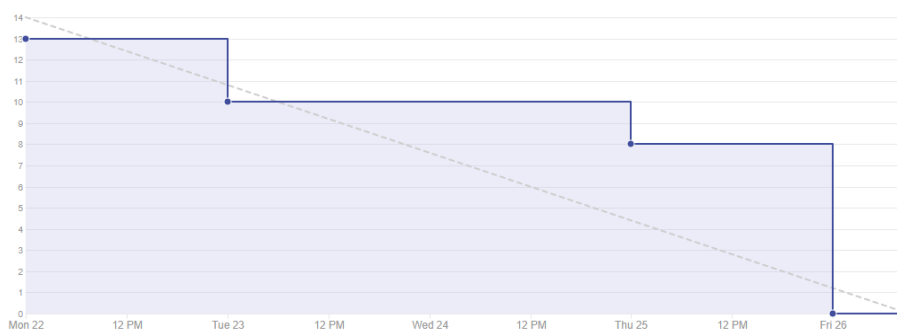


Figura A.3: Gráfica Burndown del Sprint 2

Sprint 3 (01/03/2021-05/03/2021)

- Recibir imagen de la *WebCam* en tiempo real
- Estudio de las conexiones servidor-cliente mediante TCP/IP en *Unity*
- Envío de la imagen recibida en la *WebCam* a las gafas de realidad virtual
- Solucionar *bug*: *Could not connect pins()*
- Recibir 2 *WebCams* simultáneamente en la misma escena de *Unity*
- Comenzar la redacción de Anexos de documentación en Látex
- Recibir cada *WebCam* en un ojo diferente a través de las gafas VR

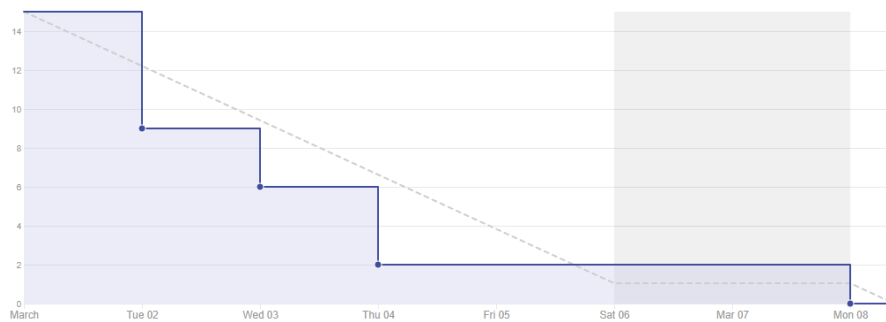


Figura A.4: Gráfica Burndown del Sprint 3

Sprint 4 (08/03/2021-12/03/2021)

- Estudiar acerca del *plugin Photon Engine*
- Comprobar cómo actúan las coordenadas de los objetos que son regidos por las gafas de RV
- Realizar conexión *Host-Cliente* en *Unity*
- Enviar vídeo a través de la TCP/IP
- Investigar acerca del protocolo UDP

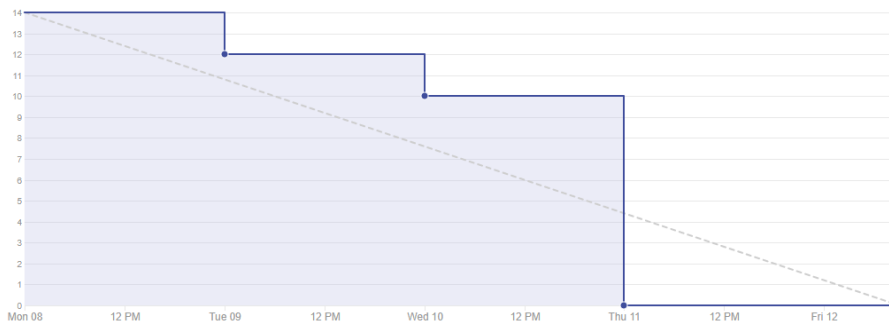


Figura A.5: Gráfica Burndown del Sprint 4

Sprint 5 (15/03/2021-19/03/2021)

- Implementar el intercambio de información vía protocolo UDP
- Optimizar el envío de imágenes mediante el protocolo TCP/IP
- Revisar la implementación del *byte* ACK en el protocolo UDP
- Estudiar el uso del *Asset FMETP STREAM*
- Recabar información acerca del *NodeDSS*
- Diseñar la interfaz a la que daremos funcionalidad en nuestra *app*

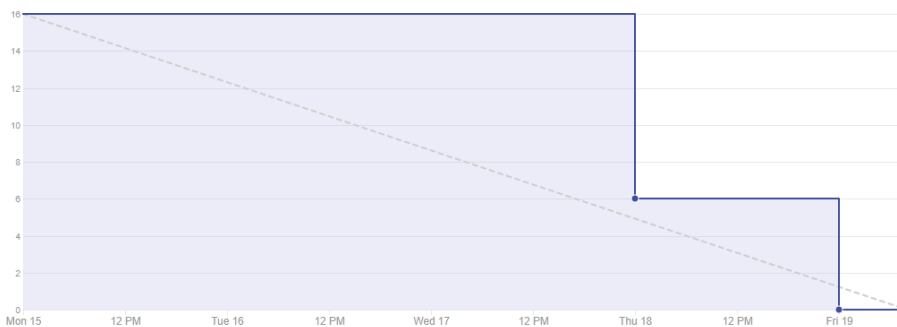


Figura A.6: Gráfica Burndown del Sprint 5

Sprint 6 (22/03/2021-26/03/2021)

- Estudio de otras alternativas para la retransmisión de vídeo en tiempo real
- Recibir las coordenadas de rotación de las gafas de realidad virtual
- Conversión de *Quaternion* a *Vector3*

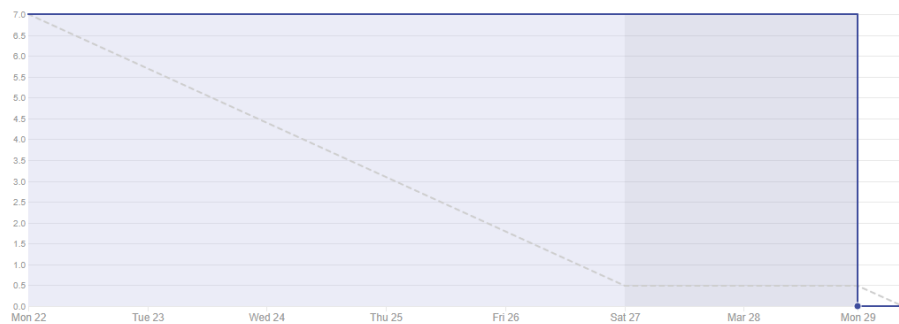


Figura A.7: Gráfica Burndown del Sprint 6

Sprint 7 (29/03/2021-02/04/2021)

- Creación de un canal de comunicación por protocolo RTC
- Comprobar la diferencia entre usar las coordenadas del *GameObject* de una escena o las del espacio real de las gafas
- Crear un nodo que haga de servidor local para ayudar al RTC *Data Channel*

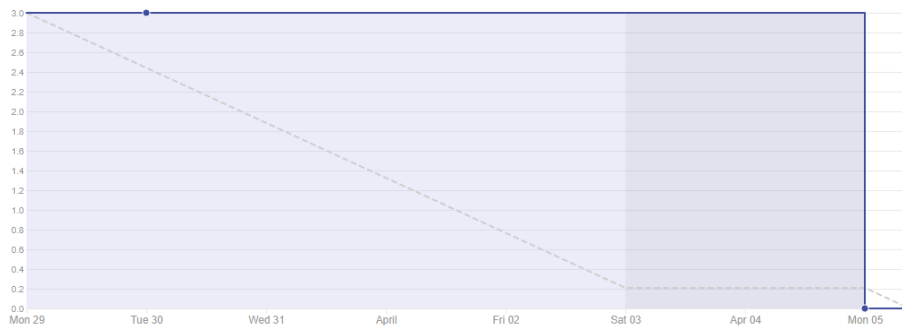


Figura A.8: Gráfica Burndown del Sprint 7

Sprint 8 (05/04/2021-09/04/2021)

- Conectar 2 equipos en red local a través del *NodeDSS*
- Enviar información a través de un canal RTC
- implementar el *plugin* de RTC en *unity*
- Comenzar la creación de la interfaz para el uso de la aplicación
- Crear botón que active y desactive la RV cuando se quiera

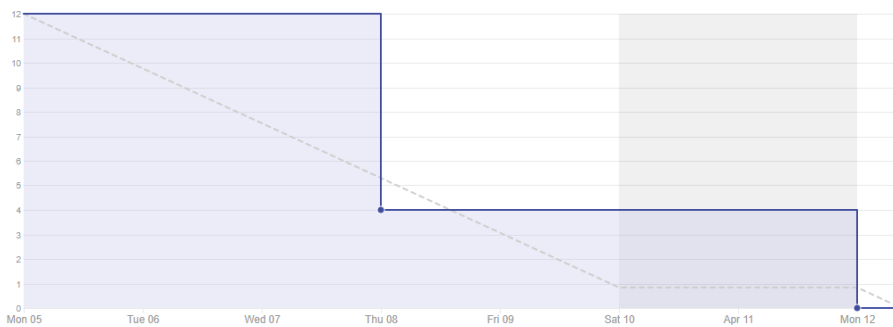


Figura A.9: Gráfica Burndown del Sprint 8

Sprint 9 (12/04/2021-16/04/2021)

Este sprint conlleva la errata de no haber cerrado hasta un mes después las últimas dos tareas. Por lo que la gráfica tiene una forma rectangular.

- Enviar vídeo a través del protocolo RTC
- Conseguir intercambio de mensajes mediante el *NodeDSS*

A.3. Estudio de viabilidad

Viabilidad económica

En el siguiente apartado se realizará un análisis de costes y beneficios hipotéticos, poniendo como escenario ficticio un desarrollo en un entorno empresarial real.

Costes

La estructura de costes del proyecto se segmentará de la siguiente manera:

Costes de personal

En este apartado se revisarán los costes referentes al personal que ha participado en el desarrollo del proyecto.

El proyecto ha sido desarrollado por un único desarrollador durante 10 semanas a tiempo completo, es decir, 40 horas semanales. Se considerará el sueldo medio en el sector, las retenciones debidas al IRPF y el valor del pago que la empresa hace a la seguridad social debido a la contratación y tiempo de trabajo que el desarrollador realice bajo la condición de empleado.

Costes de personal	
Salario mensual neto	3.870,5€
Coste de retención IRPF(26.67 %)	3733,8€
Coste de Seguridad social (28.3 %)	3962€
Salario mensual bruto	5600€
Total: 17372.05€	

Tabla A.1: Costes de personal

Costes de hardware

En este apartado se revisan los gastos referentes al hardware utilizado en el desarrollo del proyecto.

Costes <i>hardware</i>	
PC	1200€
<i>HTC VIVE</i>	700€
<i>Webcams</i>	40€
Total: 1940€	

Tabla A.2: Costes hardware

Costes de software

En este apartado se revisan los costes referentes a licencias no gratuitas del software utilizado en el desarrollo del proyecto.

Costes <i>software</i>	
Licencia Pro de <i>Unity</i>	1476.42€
Licencia <i>Enterprise</i> de <i>GitKraken</i>	88.18€
Licencia de <i>Rider</i>	339€
Total: 1913.6€	

Tabla A.3: Costes software

Costes totales

Sumatorio de todos los costes.

Costes totales	
Costes de personal	17372.05€
Costes <i>software</i>	1703.6€
Total: 19215.65€	

Tabla A.4: Costes de personal

Beneficios

En este caso se considera que el *hardware* lo paga el cliente ya que la empresa únicamente lo configura, por lo que no es necesario amortizarlo e irá incluido en el precio de venta. De esta forma, únicamente sería necesario amortizar el sumatorio de los costes de personal y los costes software. Por lo

tanto, se cree que con un precio de 3800€ sería suficiente para llegar a una relación razonable de prototipos vendidos - costes amortizados, ya que con un beneficio de 1860€, una vez vendidas 11 unidades, se habría conseguido una amortización completa.

Viabilidad legal

Hay que tener claro que al tratarse de un proyecto para desarrollo de un prototipo con fines académicos, todas las pautas legales son notablemente leves, puesto que a los estudiantes por lo general se nos permite el uso de herramientas con un menor número de restricciones.

Aún así, este prototipo puede llegar a ser la base de uno de los elementos de un producto final, por lo que, si ITCL lo considera necesario, será la empresa la que imponga las condiciones de comercialización que crea pertinentes.

Software utilizado

En cuanto a las licencias del *software* utilizado, se destacan 3 programas: *Unity*, *Rider* y *GitKraken*.

La licencia utilizada en ***Unity***[12] corresponde a una licencia de uso personal ya que a la hora de instalarme en mi puesto de trabajo me dijeron que al tratarse de un prototipo no habría problemas de licencias. Advirtiéndome también, que si en algún momento tuviera que dar el salto de prototipo a producto, tendrían que poner como principal desarrollador a ITCL puesto que la empresa sí que tiene una licencia pro para desarrollo de aplicaciones.

El uso de ***Rider***[6] a se dio a partir de la mitad del proyecto, como se menciona en la memoria, y en ese momento se tuvo que hacer una petición a *GitHub* de una licencia estudiantil que permitía el uso de un grupo amplio de programas, entre los que se encontraba el primeramente mentado. Por ello todo lo referente a la licencia de *Rider* está cubierto desde el punto de vista legal, siempre y cuando se trate de este prototipo. Volveríamos a lo mismo si se decidiera convertir en producto, la empresa pasaría a ponerse como principal desarrollador, indicando también el uso de licencia de *Rider* o *VisualStudio* en función del gusto del desarrollador.

Por último, ***GitKraken***[3] es utilizado gracias a la licencia general que recibimos los estudiantes y una que tiene ITCL, de tal manera que llegaríamos a la misma situación, bajo el nombre de prototipo no habría

problema con la licencia de estudiante, pero una vez llegado a producto habría que modificarla por la perteneciente a ITCL.

Resumen de licencias utilizadas:

Licencias usadas	
Licencia Pro	<i>Unity</i>
Licencia <i>Enterprise</i>	<i>GitKraken</i>
Licencia <i>Rider</i>	<i>JetBrains</i>

Tabla A.5: Costes de personal

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado se buscará identificar correctamente los dos tipos de requisitos que componen nuestro proyecto, de tal manera que queden definidas por un lado, las funciones del sistema software o de cualquiera de sus componentes, y por otro, las características de funcionamiento del mismo/s.

De esta manera, diferenciaremos entre requisitos funcionales[10][20], para todo aquello referente a los servicios que ofrece el sistema software, y requisitos no funcionales[21] para todo lo referente a los atributos de calidad del propio sistema.

B.2. Objetivos generales

Como se ha comentado en la memoria, el objetivo principal de este proyecto es la creación de un sistema de comunicación audiovisual en la que un extremo emita imágenes a través de dos dispositivos y el otro extremo tenga la capacidad de reproducirlas por separado en un equipo de realidad virtual para conseguir visión estereoscópica.

Como funcionalidad añadida, además, el extremo receptor de la imagen deberá devolver unas coordenadas que indican el movimiento de rotación del dispositivo de realidad virtual para, en un futuro, ser interpretadas como el movimiento de la cabeza humana de cara a mover las propias cámaras retransmisoras.

B.3. Catalogo de requisitos

Los requisitos derivados de los objetivos generales serán los siguientes:

Requisitos funcionales

- RF 1 Lanzar un servidor local para permitir la conexión
 - RF 1.1 Ejecutar el *NodeDSS* para que hagas las funciones de servidor local
- RF 2 Utilizar el sistema de visión
 - RF 2.1 Conseguir una conexión entre dos máquinas
 - RF 2.2 Conseguir el envío y recepción de imágenes a través de la conexión
 - RF 2.2.1 Reconocer la *webcam* en *Unity*
 - RF 2.2.2 Mostrar la imagen de la *webcam* en *Unity*
 - RF 2.2.3 Enviar la imagen
 - RF 2.2.4 Recibir la imagen
 - RF 2.3 Conseguir el envío de coordenadas a través de la conexión
 - RF 2.3.1 Tomar las coordenadas de rotación de las gafas de realidad virtual
 - RF 2.3.2 Enviar las coordenadas
 - RF 2.3.3 Recibir las coordenadas
 - RF 2.4 Reproducir las imágenes recibidas en un dispositivo de realidad virtual
 - RF 2.4.1 Reconocer el dispositivo en *Unity*
 - RF 2.4.2 Enviar cada conjunto de imágenes a un ojo

Requisitos no funcionales

- **RNF 1 - Inmersividad:** Este requisito es imprescindible ya que se puede considerar el distintivo principal de todo el proyecto porque será lo que lo hará llamativo a la hora de considerarse una opción real para acoplar a cualquier sistema externo.

- **RNF 2 - Rendimiento:** Otro de los requisitos de mayor importancia puesto que al tratarse de vídeo, si el rendimiento que se consigue no es óptimo, la calidad de la transmisión tampoco lo será y el sistema de visión perderá eficacia y utilidad.
- **RNF 3 - Adaptabilidad:** Cuando se habla de adaptabilidad se refiere a la posibilidad de que la aplicación sea utilizada con diferentes dispositivos, tanto de realidad virtual, como *webcams*, como ordenadores.
- **RNF 4 - Escalabilidad:** Con la escalabilidad se busca que el sistema permita su desarrollo para rangos de comunicación mayores, es decir, no sólo en conexiones a través de servidores locales.
- **RNF 5 - Usabilidad:** Imprescindible para cualquier tipo de prototipo de cara a mostrar a cliente o posibles inversores, un esquema de uso simple y sencillo que permita captar la idea principal del desarrollo.

Especificaciones de cliente

En el apartado de especificaciones del cliente expondré las pautas dadas por el tutor empresarial, ya que en ningún momento ha habido contacto real con el cliente.

- Es obligatorio el desarrollo de la aplicación con *Unity* y toda la tipología de herramientas e implementaciones que provee al usuario. Dejando total libertad para elegir el procedimiento completo, es decir, *assets*, *plugins*, implementaciones externas si fuera necesario, etc.
- La conexión se limitará a red local, puesto que primero se quiere comprobar si es una solución viable y de verdad se obtienen los resultados esperados, y de querer hacerlo a través de conexiones de mayor rango, requeriría de mayor cantidad de recursos y tiempo.
- La primera versión deberá ser utilizable tanto en *HTC VIVE* como en *Oculus Quest 2*, ya que el primer equipo de dispositivos se posee actualmente en la empresa y el segundo tiene prevista su compra en un margen de tiempo relativamente corto.
- No hay especificación de diseño gráfico, ya que a pesar de conocer el poco nivel de experiencia del alumno en ese campo, se prioriza la funcionalidad del proyecto antes que la imagen.

B.4. Especificación de requisitos

Diagrama de casos de uso

En el siguiente diagrama B.4 mostraremos todos los casos de uso prácticos de la aplicación en función del tipo de usuario. Este tipo de usuario se verá diferenciado por el hardware que posea, ya que en este proyecto, para poder ejecutar una u otra funcionalidad es necesario un hardware específico.

No se contempla el caso de acceso al manual de uso puesto que únicamente es un salto entre escenas sin ninguna funcionalidad adicional más allá de una ventana de texto.

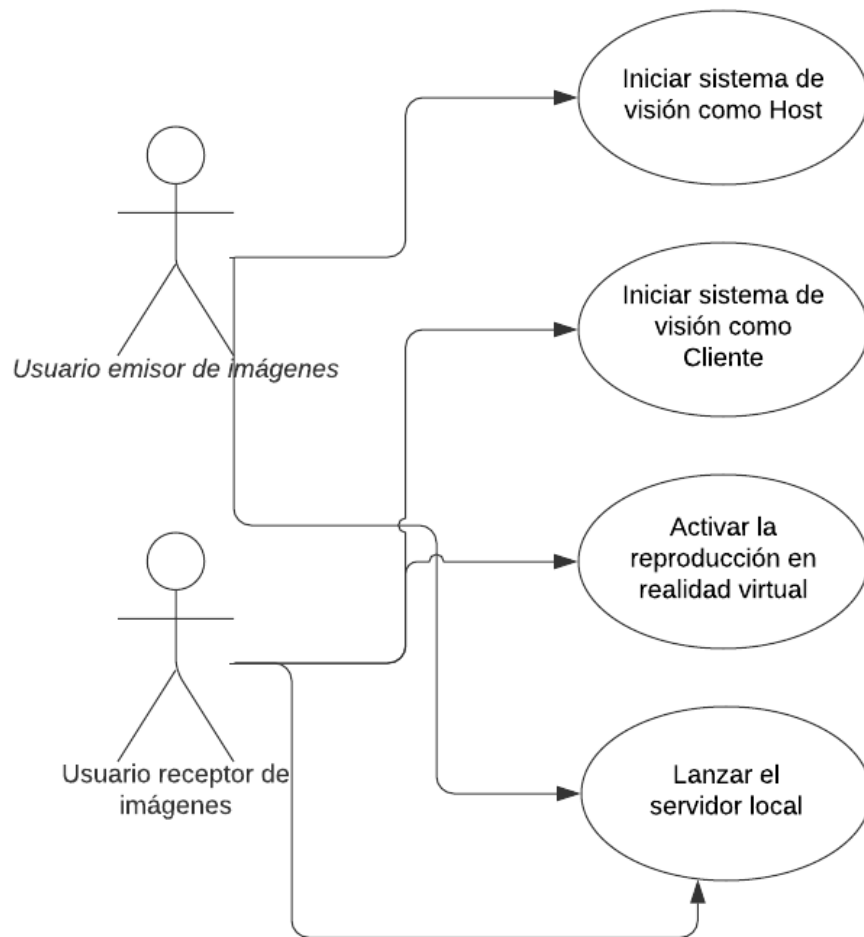


Figura B.1: Diagrama de casos de uso

Requisitos

Los requisitos del sistema de visión se exponen en las siguientes tablas:

RF-1.1: Ejecutar el <i>NodeDSS</i> para que hagas las funciones de servidor local.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se debe poner en marcha el nodo que hará las funciones de servidor local.
Precondiciones	<i>Node.js</i> y <i>NodeDSS</i> deben estar instalados.
Acciones	Uno de los dos usuarios del sistema de visión ejecuta desde la consola de comandos el <i>NodeDSS</i> .
Postcondiciones	El servidor queda lanzado y muestra <i>pings</i> de confirmación cada x milisegundos (intervalo elegido por el usuario).
Excepciones	Ninguna.
Importancia	Muy alta

Tabla B.1: RF-1.1: Ejecutar el NodeDSS para que hagas las funciones de servidor local.

RF-2.1: Conseguir una conexión entre dos máquinas.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se debe establecer una conexión a través de un servidor local entre 2 ordenadores.
Precondiciones	El servidor local debe estar lanzado.
Acciones	Ambos usuarios han iniciado sus respectivas escenas y establecido la conexión.
Postcondiciones	Se puede confirmar que los ordenadores están conectados.
Excepciones	Ninguna.
Importancia	Muy alta

Tabla B.2: RF-2.1: Conseguir una conexión entre dos máquinas.

RF-2.2: Conseguir el envío y recepción de imágenes a través de la conexión.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se debe conseguir una emisión y recepción de imágenes óptima y fluida.
Precondiciones	Los ordenadores deben estar conectados.
Acciones	Un usuario inicia una escena de emisión; Otro usuario inicia una escena de recepción.
Postcondiciones	Se debe poder ver el <i>streaming</i> en el dispositivo pertinente del extremo del cliente.
Excepciones	Ninguna.
Importancia	Muy alta

Tabla B.3: RF-2.2: Conseguir el envío y recepción de imágenes a través de la conexión.

RF-2.2.1: Reconocer la <i>webcam</i> en <i>Unity</i> .	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se debe reconocer la <i>webcam</i> como dispositivo disponible en <i>Unity</i> .
Precondiciones	La cámara debe estar conectada al ordenador.
Acciones	Un usuario inicia una escena de emisión.
Postcondiciones	<i>Unity</i> reconoce la cámara.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.4: RF-2.2.1: Reconocer la webcam en Unity.

RF-2.2.2: Mostar la imagen de la <i>webcam</i> en <i>Unity</i> .	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se retransmite en la escena de <i>Unity</i> la imagen que capta la <i>webcam</i> .
Precondiciones	La cámara debe ser reconocida por <i>Unity</i> ; Debe haber una superficie de reproducción en la escena.
Acciones	Un usuario inicia una escena de emisión.
Postcondiciones	Se ve en <i>Unity</i> lo que la <i>webcam</i> reproduce.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.5: RF-2.2.2: Mostrar la imagen de la webcam en Unity.

RF-2.2.3: Enviar la imagen.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se envía la imagen a través de la conexión establecida previamente.
Precondiciones	Se debe obtener la imagen de la <i>webcam</i> para poder comprimirla y enviarla.
Acciones	Un usuario inicia una escena de emisión; La conexión está establecida.
Postcondiciones	Se podría comprobar si la imagen se está enviando correctamente.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.6: RF-2.2.3: Enviar la imagen.

RF-2.2.4: Recibir la imagen.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se recibe la imagen a través de la conexión establecida previamente.
Precondiciones	Se debe haber enviado la imagen a través de la conexión.
Acciones	Un usuario inicia una escena de emisión; Otro usuario una escena de recepción; La conexión está establecida.
Postcondiciones	Se podría comprobar si la imagen se está enviando correctamente.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.7: RF-2.2.4: Recibir la imagen.

RF-2.3: Conseguir el envío de coordenadas a través de la conexión.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se debe conseguir un envío íntegro y fiable de las coordenadas de rotación de las gafas de RV desde el extremo del receptor de imágenes.
Precondiciones	Se debe haber reconocido el dispositivo en <i>Unity</i> ; Debe estar la conexión PC-PC establecida.
Acciones	El usuario que recibe imágenes activa la escena de realidad virtual.
Postcondiciones	Las coordenadas se muestran en la escena del emisor.
Excepciones	Ninguna.
Importancia	Muy alta

Tabla B.8: RF-2.3: Conseguir el envío de coordenadas a través de la conexión.

RF-2.3.1: Tomar las coordenadas de rotación de las gafas de realidad virtual.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se debe conseguir guardar en una variable las coordenadas de rotación de las gafas de realidad virtual.
Precondiciones	Se debe haber reconocido el dispositivo en <i>Unity</i> .
Acciones	El usuario que recibe imágenes activa la escena de realidad virtual.
Postcondiciones	Se podría comprobar si las coordenadas se están guardando correctamente.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.9: RF-2.3.1: Tomar las coordenadas de rotación de las gafas de realidad virtual.

RF-2.3.2: Enviar las coordenadas.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se envían las coordenadas a través de la conexión establecida previamente.
Precondiciones	Se deben haber guardado correctamente las coordenadas.
Acciones	Un usuario inicia una escena de emisión de imágenes; La conexión está establecida; El usuario que recibe imágenes activa la escena de realidad virtual.
Postcondiciones	Se podría comprobar si las coordenadas se están enviando correctamente.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.10: RF-2.3.2: Enviar las coordenadas.

RF-2.3.3: Recibir las coordenadas.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se reciben las coordenadas a través de la conexión establecida previamente.
Precondiciones	Se deben haber enviado correctamente las coordenadas a través de la conexión.
Acciones	Un usuario inicia una escena de emisión de imágenes; El usuario que recibe imágenes activa la escena de realidad virtual; La conexión está establecida.
Postcondiciones	Las coordenadas se muestran en la escena del emisor.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.11: RF-2.3.3: Recibir las coordenadas.

RF-2.4: Reproducir las imágenes recibidas en un dispositivo de realidad virtual.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Se reproducen las imágenes recibidas en las gafas de realidad virtual.
Precondiciones	Se debe haber producido un correcto envío y recepción de las imágenes y una correcta activación de la escena de realidad virtual.
Acciones	El usuario que está en el extremo de recepción de imágenes activa el botón de realidad virtual.
Postcondiciones	Las imágenes se reproducen en las gafas de realidad virtual.
Excepciones	Ninguna.
Importancia	Muy alta

Tabla B.12: RF-2.4: Reproducir las imágenes recibidas en un dispositivo de realidad virtual.

RF-2.4.1: Reconocer el dispositivo en <i>Unity</i> .	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	<i>Unity</i> reconoce las gafas como dispositivo de realidad virtual.
Precondiciones	El proyecto debe estar configurado para soportar RV y las gafas reconocidas por el ordenador a través de <i>SteamVR</i> .
Acciones	Un usuario debe activar la aplicación.
Postcondiciones	Las gafas de <i>Unity</i> deben mostrar la pantalla de carga.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.13: RF-2.4.: Reconocer el dispositivo en *Unity*.

RF-2.4.2: Enviar cada conjunto de imágenes a un ojo.	
Versión	1.0
Autor	Pablo Seoane Fuente
Descripción	Cada una de las pantallas de las gafas muestran un conjunto de imágenes de cada <i>webcam</i> respectivamente.
Precondiciones	Toda la retransmisión de vídeo debe funcionar correctamente; La escena de realidad virtual debe funcionar correctamente.
Acciones	Un usuario debe activar la escena de realidad virtual.
Postcondiciones	Las gafas reproducirán las imágenes recibidas del otro extremo de la conexión simulando la vista del ojo humano.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.14: RF-2.4.2: Enviar cada conjunto de imágenes a un ojo.

Apéndice C

Especificación de diseño

C.1. Introducción

En este apartado [19] en el que se podrían tratar diversos aspectos, puesto que abarca un incontable número de componentes (algunos ya tratados debido a su importancia, como los requisitos), en este anexo se centrará más en la organización de los datos.

C.2. Diseño de datos

Durante todo el desarrollo, testeo y uso de la aplicación que se implementa en este proyecto, únicamente se trabaja con información en *Bytes*. De tal manera que tanto las ofertas de conexión que se realizan entre los dos extremos de la conexión pasando por el servidor, como las imágenes que van a ser retransmitidas y recibidas, como las coordenadas que van a ser retransmitidas de vuelta, van a hacerlo en forma de *Bytes*.

El proceso va a ser de obtención de datos, codificación de los mismos convirtiendo su tipo al anteriormente mentado para un envío óptimo, eficiente e íntegro, recepción, revertir la codificación para conocer cuál era el dato real e interpretarlo como tal. Este proceso se realizará principalmente con imágenes y coordenadas.

Como punto a nombrar, cabe decir que a la hora de establecer la conexión, tendrá lugar una conversación en la que un extremo hace una oferta al otro, y si es de gusto, es decir, si el *id* enviado en la oferta coincide con el propio, enviará un paquete de aceptación de vuelta para establecer la conexión. En caso negativo, ignorará la oferta y no enviará respuesta.

C.3. Diseño arquitectónico

La única particularidad de esta aplicación va a ser acudir a los métodos implementados en *Unity* para la gestión de escenas, que recaen en el *Scene-Manager*[13] el cuál nos permitirá hacer transiciones de una escena a otra. El diagrama C.3 que mostrará la simplicidad del diseño arquitectónico de la aplicación será el siguiente:

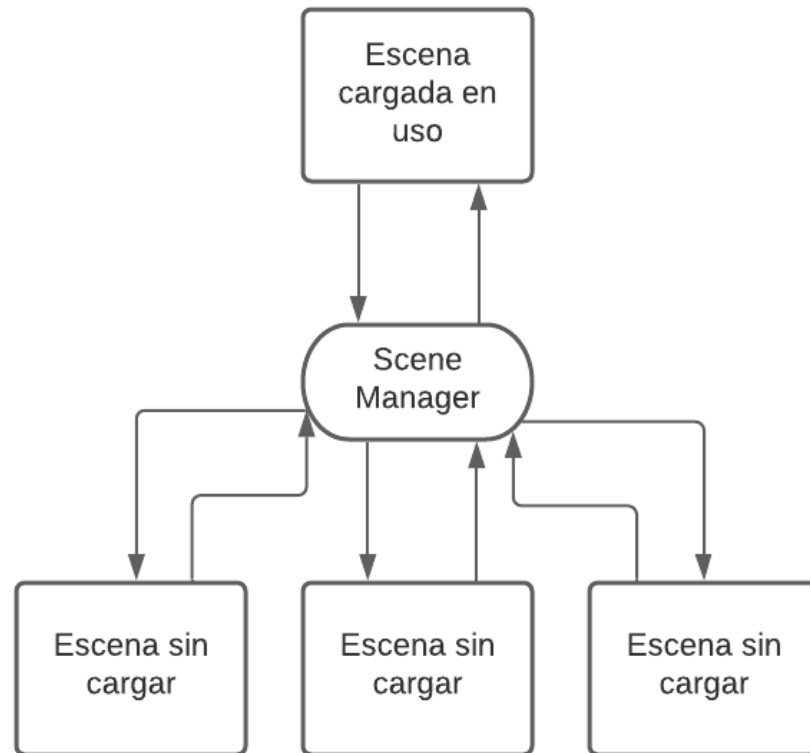


Figura C.1: Diseño arquitectónico

En la aplicación la transición entre escenas está regulada por el uso de botones, lo cuál permitirá desactivar la escena actual y dar la orden de carga de una de las que no estaba siendo utilizada en ese momento.

Como aclaración principal hay que decir que cuando se activa la realidad virtual en la escena *Streaming Client* no se desactiva la escena en ningún momento, únicamente se carga la configuración de las propiedades de reproducción de la RV en Unity.

C.4. Diseño de interfaces

Para este prototipo no se ha necesitado la ayuda de ningún especialista en diseño gráfico puesto que la prioridad era la funcionalidad. Por ello se ha llegado a un diseño muy simple que se expondrá en el siguiente diagrama C.4:

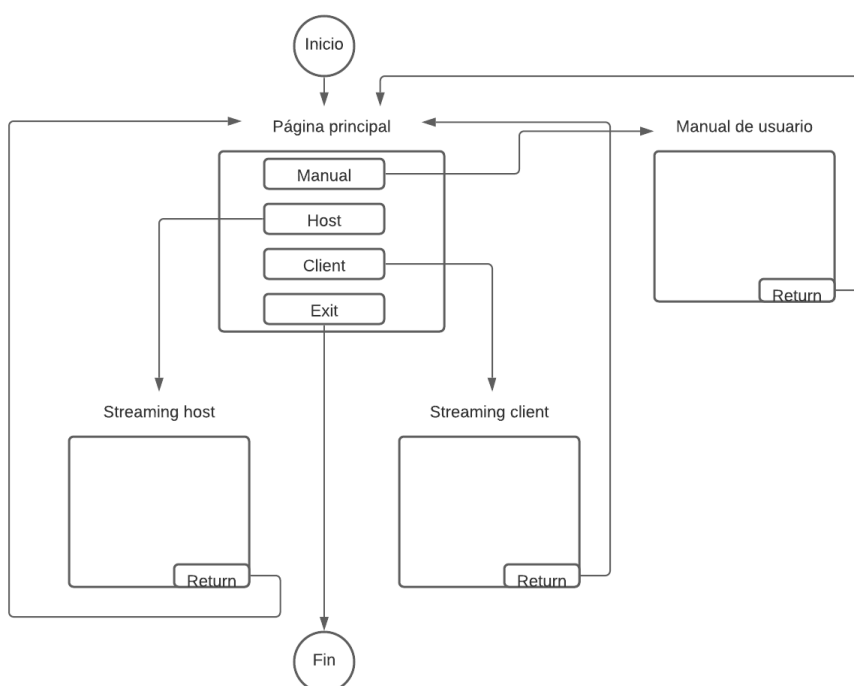


Figura C.2: Diseño de pantallas

Además, a continuación mostraremos capturas de cada una de las escenas que verá el usuario cuando lance la aplicación. Aunque se analizarán las funcionalidades en el apartado de manual de usuario, aquí las mostramos para apreciar cómo están creadas, por lo que irán acompañadas de los elementos que las componen.

Menú

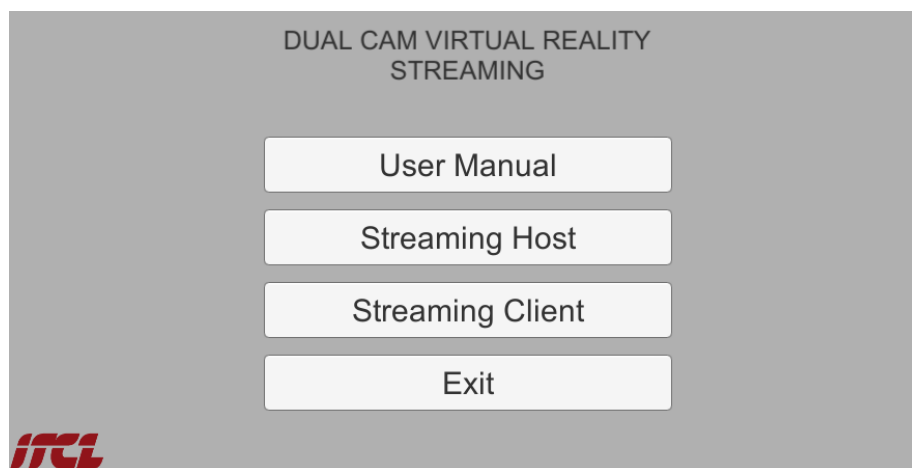


Figura C.3: Escena inicial de la aplicación

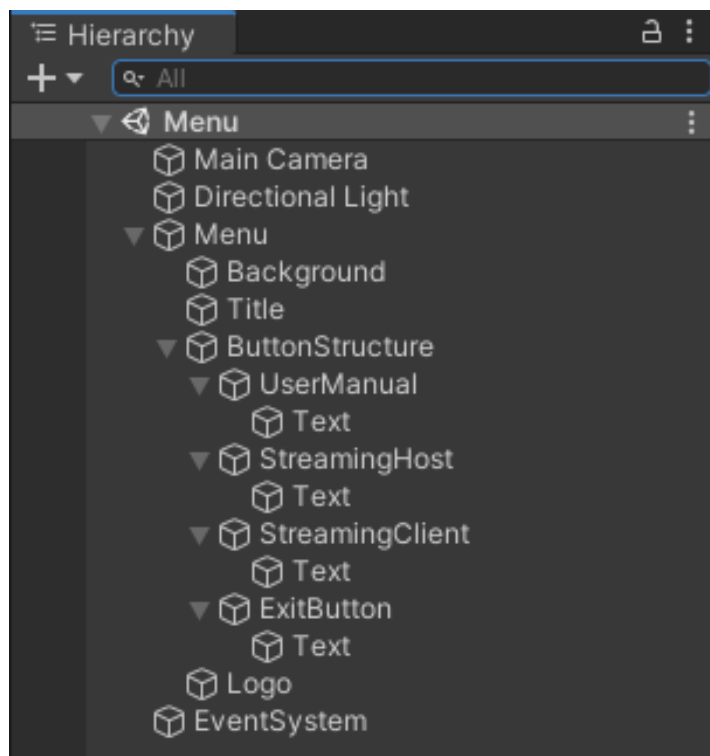


Figura C.4: Inspector Menu

Manual

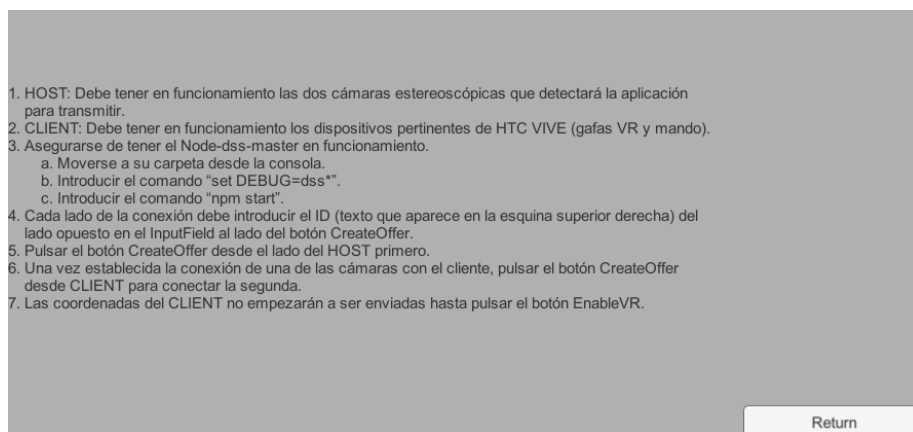


Figura C.5: Escena con un pequeño manual de funcionamiento para el usuario

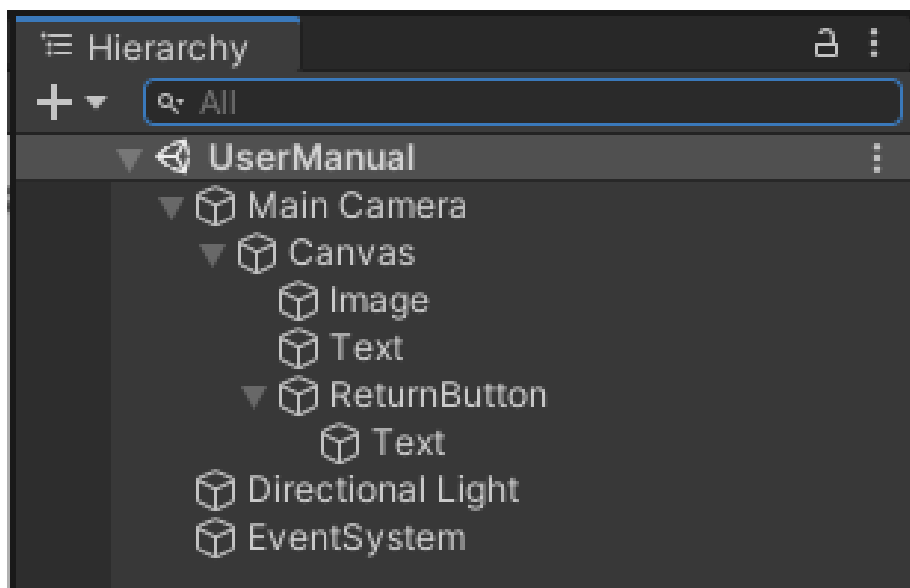


Figura C.6: Inspector UserManual

Streaming Host

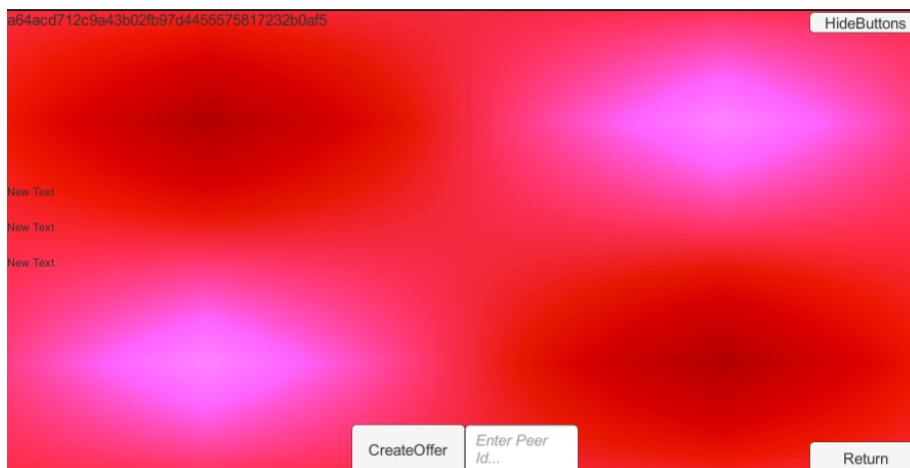


Figura C.7: Escena del emisor de imágenes

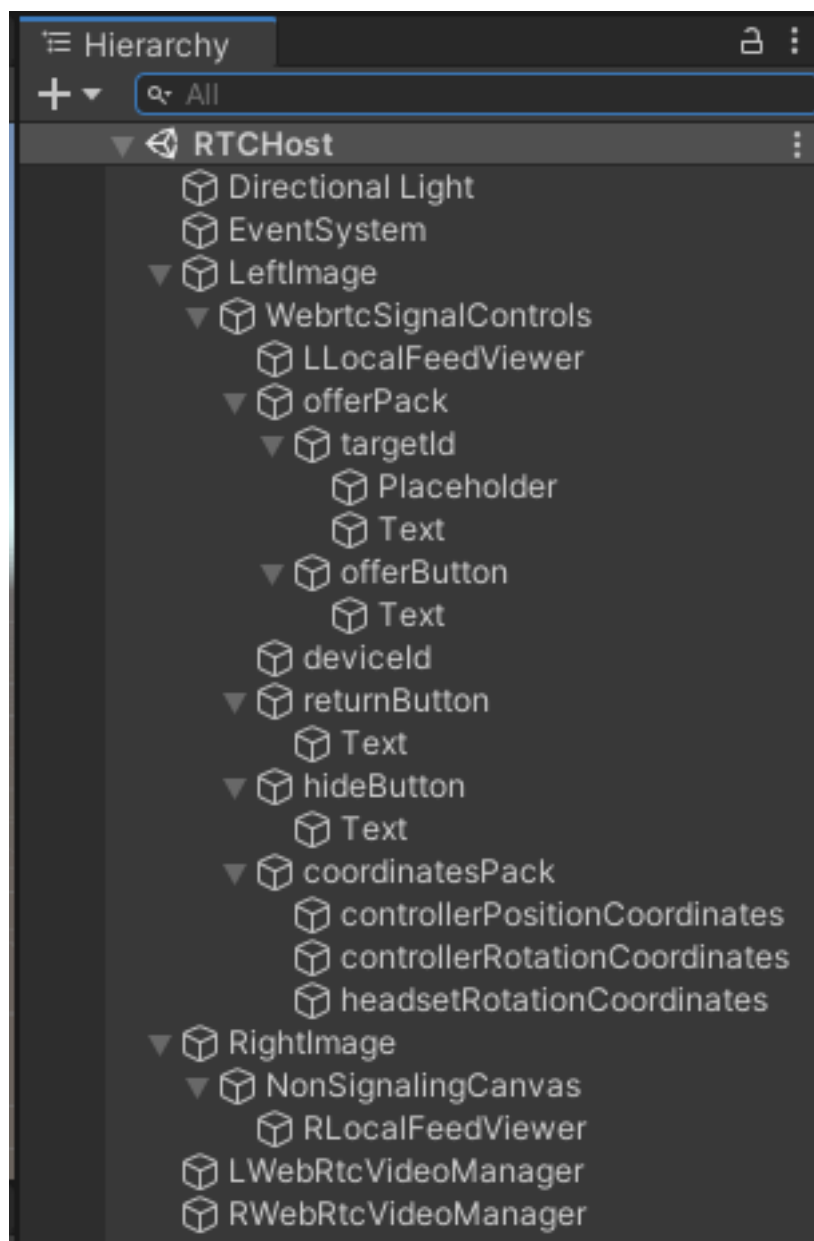


Figura C.8: Inspector Host

Streaming Client

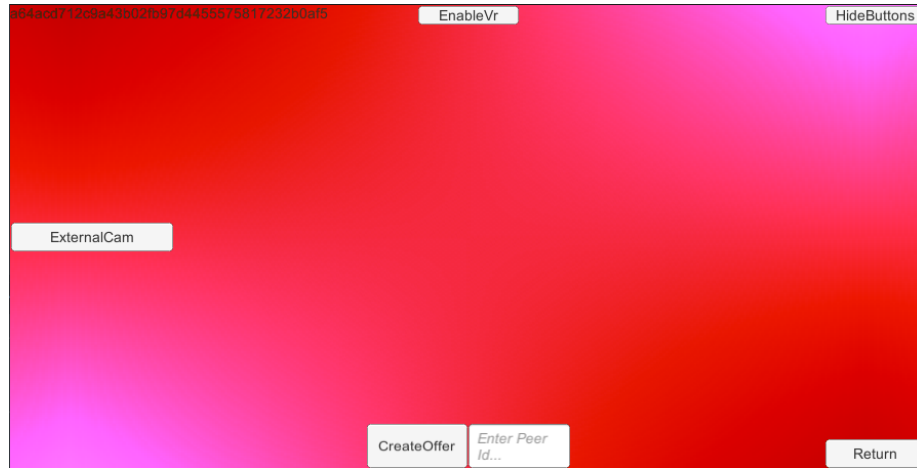


Figura C.9: Escena del receptor de imágenes

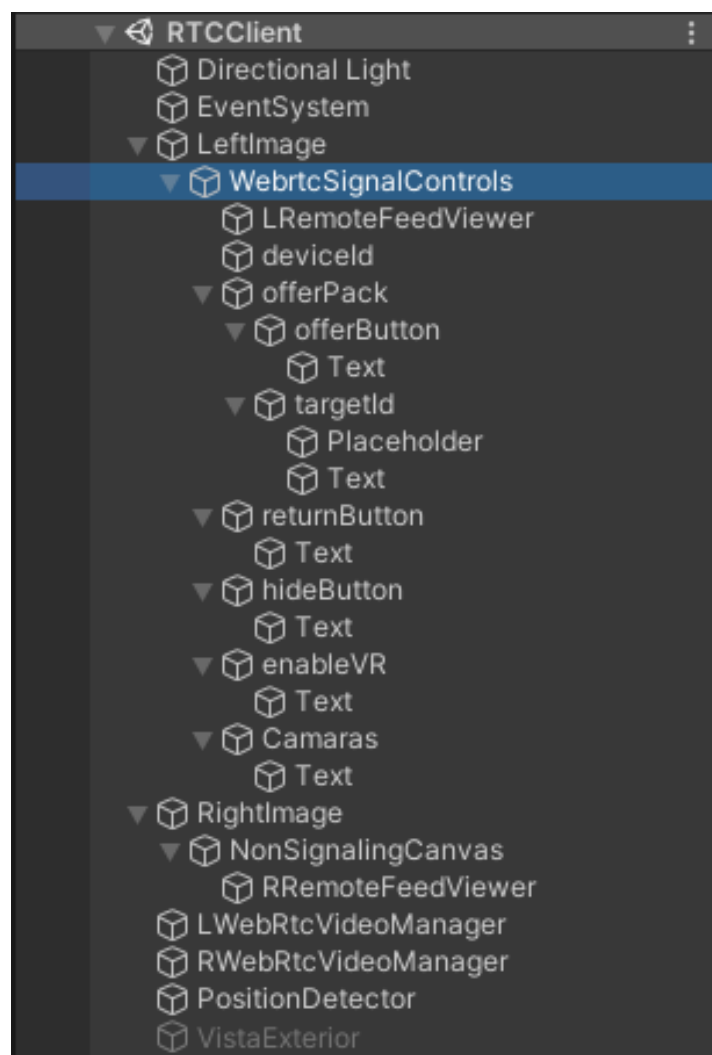


Figura C.10: Inspector Client

Apéndice *D*

Documentación técnica de programación

D.1. Introducción

La documentación técnica es muy importante en el desarrollo *software*[2]. Dejar el proceso por escrito, sirve para entender el desarrollo del proyecto, cómo funciona y cómo se utiliza.

Es verdad que una vez se tiene la carpeta que contiene el proyecto, sólo será necesario tener *Unity*[12] para poder verlo en un IDE y poder crear *builds*, pero si queremos realizar cualquier tipo de modificación, serán necesarios conocimientos de uso de *Unity* y de programación en lenguaje *C#*.

Por otro lado, para visualizar todo lo relacionado con la realidad virtual, independientemente de si se quiere hacer desde dentro del IDE o desde una versión de la aplicación, será necesario tener instalados y configurados correctamente por un lado el hardware de RV (*HTC VIVE*[17][18]) y por otro el software que permite su uso (*Steam*[14] y *SteamVR*[15])

D.2. Estructura de directorios

A continuación se detalla la organización de los directorios más importantes del proyecto:

- **/Assets:** Carpeta que guarda los variedad de elementos del proyecto.

- **/Assets/Plugins:** Directorio que guarda carpetas correspondientes a la versión del sistema operativo de windows.
 - **/Assets/Resources:** Directorio que guarda un archivo *.json* con información de *Android*.
 - **/Assets/Scenes:** Directorio que almacena las 4 escenas que tiene la aplicación.
 - **/Assets/UnityWebRTC:** Directorio que almacena el *plugin* de *Unity WebRTC*.
- **/Library:** Carpeta que almacena las librerías de *Unity*.
 - **/Logs:** Carpeta que contiene un documento de texto con información sobre los paquetes introducidos en el proyecto.
 - **/Packages:** Carpeta que contiene los paquetes del proyecto.
 - **/ProjectSettings:** Carpeta que contiene las opciones del proyecto.

D.3. Manual del programador

El siguiente manual[5] tiene como objetivo servir de referencia a futuros programadores que trabajen en la aplicación. En él se tratarán aspectos propios de un desarrollo software, entre los que se pueden destacar los requisitos técnicos necesarios del sistema, contenido adicional de terceros que se haya añadido, conceptos relativos a la modificación o ampliación del proyecto, etc.

En nuestro caso, para poder tratar el proyecto bajo el rol de desarrollador o programador será necesario tener instalados en nuestro ordenador elementos software y hardware específicos:

Requisitos Software

El requisito *software* principal tiene que ver con tener la versión de *Unity* en la que ha sido desarrollado el proyecto. En este caso se trata de la versión 2019.4, sin importar la subversión, ya que se ha abierto el archivo del propio proyecto en diferentes y no ha dado problemas con ninguna.

Instalación de Unity

Para poder tener la versión de *Unity* necesaria tendremos que instalarla, y para ello lo haremos desde una aplicación que la propia empresa *Unity*[12]

ha desarrollado llamada *Unity Hub*[11]. Desde ella podremos descargar las versiones más novedosas o aquellas que tienen están definidas como LTS (*Long Time Support*) y acceder a la página web de *Unity* que contiene todas las versiones disponibles.

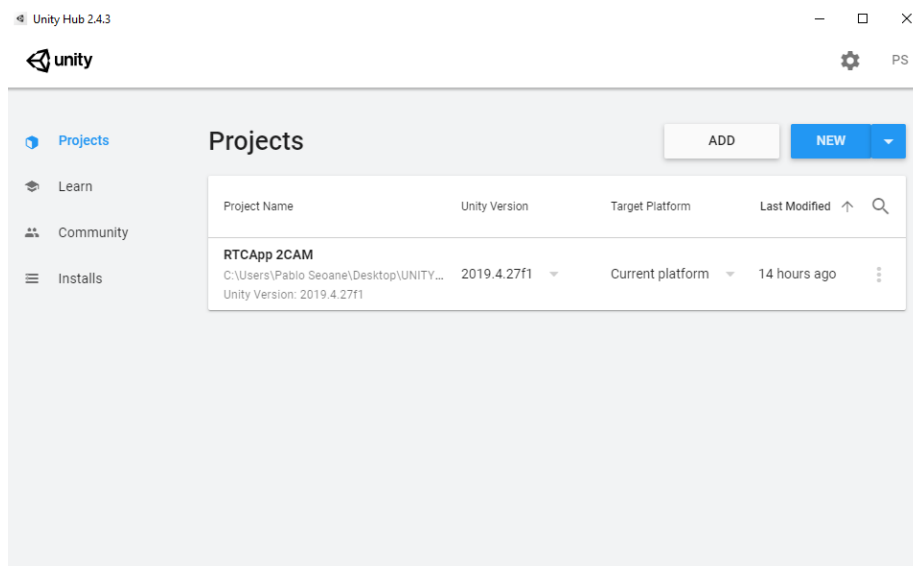


Figura D.1: Aplicación Unity Hub

Una vez instalada la versión deseada, podremos crear proyectos o acceder mediante ruta a carpetas que ya los contengan, pudiéndolos iniciar en la versión que se desea siempre que se tenga instalada. Si las versiones no coinciden, *Unity* avisará de que se pueden producir errores ya que *Unity* genera librerías y recursos necesarios la primera vez que abre un proyecto.

Instalación del entorno de programación

Este punto no será tratado de manera extensa ya que *Unity* de por sí, durante su instalación, procedería a instalar *VisualStudio*[9], ya que es el entorno predeterminado.

Si se deseara utilizar otro entorno de programación (*Rider*[6] en nuestro caso) se debería instalar primero, indicando luego en el *framework* de *Unity* que se desea que tenga preferencia a *VisualStudio*[9]. Accedemos a las preferencias del proyecto, más concretamente a la pestaña de herramientas externas (*External Tools*).

Edit	Assets	GameObject	Component	Window
Undo				Ctrl+Z
Redo				Ctrl+Y
Select All				Ctrl+A
Deselect All				Shift+D
Select Children				Shift+C
Select Prefab Root			Ctrl+ Shift+R	
Invert Selection				Ctrl+I
Cut				Ctrl+X
Copy				Ctrl+C
Paste				Ctrl+V
Duplicate				Ctrl+D
Rename				
Delete				
Frame Selected				F
Lock View to Selected				Shift+F
Find				Ctrl+F
Play				Ctrl+P
Pause			Ctrl+ Shift+P	
Step			Ctrl+Alt+P	
Sign in...				
Sign out				
Selection				>
Project Settings...				
Preferences...				
Shortcuts...				
Clear All PlayerPrefs				
Graphics Tier				>
Grid and Snap Settings...				

Figura D.2: Configuración del entorno de programación

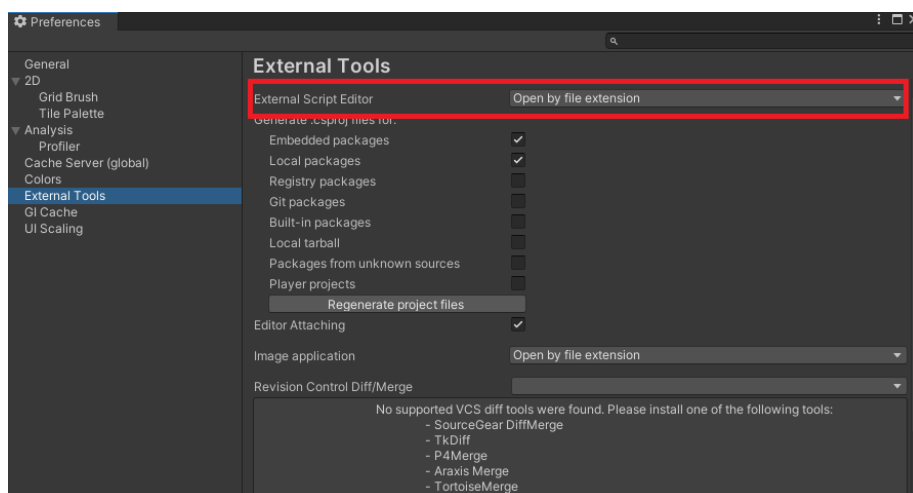


Figura D.3: Configuración del entorno de programación

En la pestaña de *External Tools* [D.3](#) seleccionaríamos el editor de *scripts* externo que deseemos.

Instalación de Steam y SteamVR

La instalación de *Steam* y *SteamVR* que el programador debe realizar quedará explicada en el manual de usuario, puesto que considero que para usuarios con conocimientos medios puede resultar más complicado que para un desarrollador, de tal manera que sólo se explique una vez en este conjunto de anexos. Aún así se puede adelantar que esta instalación se delega en un *software* que simplifica todas las descargas bajo un instalador.

Instalación de Node.js

La instalación de Node.js [\[7\]](#) no supone ninguna dificultad ya que el desarrollador únicamente tendrá que acceder a la página de descargas, elegir el instalador que más se adecúe a las características de su sistema operativo y ejecutarlo una vez descargado para instalar el programa. A continuación de muestra la página de descarga [D.3](#) dónde seleccionar la versión:

LTS

Recomendado para la mayoría

Actual

Últimas características



Instalador Windows

node-v14.17.0-x64.msi



Instalador macOS

node-v14.17.0.pkg



Código Fuente

node-v14.17.0.tar.gz

Instalador Windows (.msi)

Binario Windows (.zip)

Instalador macOS (.pkg)

Binario macOS (.tar.gz)

Binario Linux (x64)

Binario Linux (ARM)

Código Fuente

32-bit	64-bit
32-bit	64-bit
64-bit	
64-bit	
64-bit	
ARMv7	ARMv8
node-v14.17.0.tar.gz	

Figura D.4: Página de descarga de Node.js

Una vez descargado el instalador únicamente debemos ejecutarlo y seguir los pasos que indica.

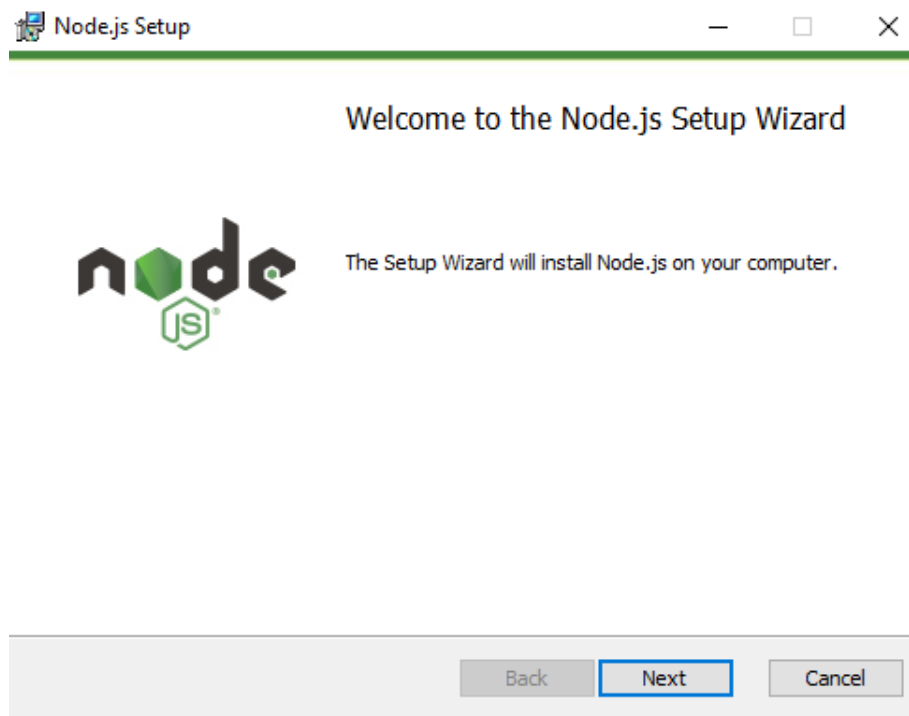
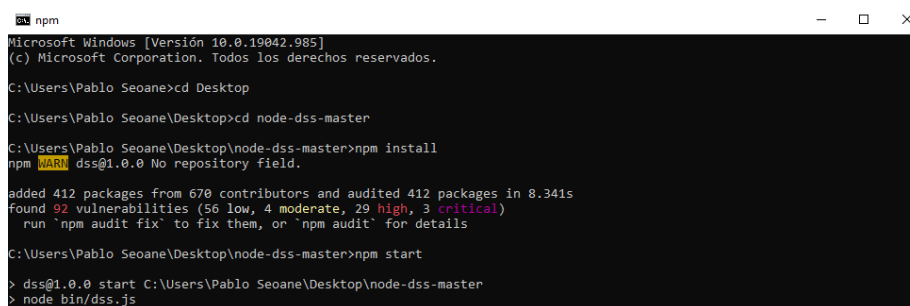


Figura D.5: Instalador de Node.js

Instalación de NodeDSS

- Para la instalación de NodeDSS se debe descargar primero el proyecto[4] que lo contiene.
- Una vez descargado, se mueve el proyecto a la ruta deseada por el usuario.
- Desde la consola de comandos se accede a la carpeta NodeDSS.
- Se ejecuta el comando `"npm install"`.
- Si se considera necesario se puede ejecutar el comando `"npm start"` para comprobar si funciona correctamente.



```
npm
Microsoft Windows [Versión 10.0.19042.985]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Pablo Seoane>cd Desktop
C:\Users\Pablo Seoane\Desktop>cd node-dss-master
C:\Users\Pablo Seoane\Desktop\node-dss-master>npm install
npm WARN dss@1.0.0 No repository field.

added 412 packages from 670 contributors and audited 412 packages in 8.341s
found 92 vulnerabilities (56 low, 4 moderate, 29 high, 3 critical)
  run 'npm audit fix' to fix them, or 'npm audit' for details

C:\Users\Pablo Seoane\Desktop\node-dss-master>npm start

> dss@1.0.0 start C:\Users\Pablo Seoane\Desktop\node-dss-master
> node bin/dss.js
```

Figura D.6: Instalación y primera ejecución de NodeDSS

Requisitos Hardware

Refiriéndome a los requisitos *hardware*, y teniendo en cuenta que las gafas *HTC VIVE* son dispositivos que necesitan un ordenador de características potentes para un uso óptimo, a continuación mostraremos una tabla E.1 con las especificaciones mínimas que debería tener nuestra máquina para que el dispositivo RV se acoplara a él y desarrollara su funcionalidad sin ningún tipo de contratiempo:

De igual manera que *Steam* y *SteamVR*, su instalación se explicará de manera detallada en el manual del usuario, dejando los *links* necesarios que servirán de ayuda. Esto se produce al mantener la opinión de que para un usuario medio, una instalación relativamente sencilla puede resultar más complicada que para un desarrollador.

Requisitos mínimos de <i>HTC VIVE</i> .	
Procesador	<i>Intel™ Core™ i5-4590</i> o <i>AMD FX™ 8350</i> , equivalente o superior.
Gráficos	<i>NVIDIA GeForce™ GTX 1060</i> o <i>AMD Radeon™ RX 480</i> , equivalente o superior.
Memoria	4GB RAM o superior.
Salida de vídeo	Un puerto HDMI 1.4 o <i>DisplayPort</i> 1.2 o superior.
USB	Un puerto USB 2.0 o superior.
Sistema operativo	<i>Windows 7</i> SP1, 8.1, 10 o superior.

Tabla D.1: Requisitos mínimos de HTC VIVE

D.4. Instalación y ejecución del proyecto

Para la instalación del proyecto volveremos a hablar de la aplicación *Unity Hub*, ya que, dando por hecho que poseemos el archivo en el que se almacenará la estructura del mencionado anteriormente, tendremos que crear una ruta para que *Unity* pueda abrirlo. Además tendremos que seleccionar la versión con la que se desee abrir y la plataforma para la que queramos hacerlo.

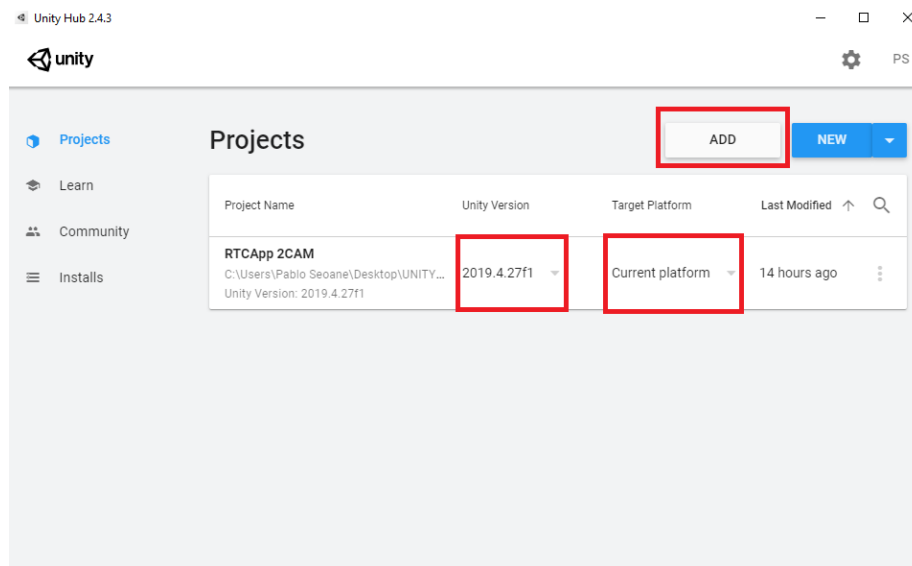


Figura D.7: Añadir proyectos a Unity Hub

Es importante recordar que se debe abrir el proyecto con la misma versión en la que está creado (obviando la subversión salvo que haya mucha

diferencia entre la de creación del proyecto y la deseada por el usuario) para evitar cualquier tipo de malentendido entre versiones que pueda provocar un fallo de alguna de las implementaciones realizadas durante del desarrollo. usar misma versión.

También cabe decir, que en la mayoría de ocasiones en las que un proyecto se abre con la misma versión y distinta subversión, *Unity* únicamente reconstruye algunas secciones de la estructura del proyecto y el funcionamiento sigue siendo correcto. Aún así, se recomienda que la diferencia sea lo menor posible.

D.5. Builds del proyecto

En *Unity*, las *builds* podrían ser las diferentes versiones de aplicación que se crean para un proyecto, ya que, a pesar de que el *framework* de trabajo ya permite poner en marcha las diferentes escenas con unas opciones de ejecución y parada de la escena, es interesante compilar la aplicación y utilizarla como un archivo ejecutable para comprobar si toda la funcionalidad implementada se comporta según lo esperado.

Tras haber mostrado el entorno con su posibilidad de ejecución de escenas [D.5](#), mostramos la ventana *Build Settings* [D.5](#) y las opciones que nos ofrece una vez abierta. Encontraremos varios campos para definir las características que tendrá nuestro proyecto. Como se puede ver en la imagen, se han señalado las que más se han utilizado. Para acceder a ella basta con ir a *File > Build Settings* en la barra de herramientas.

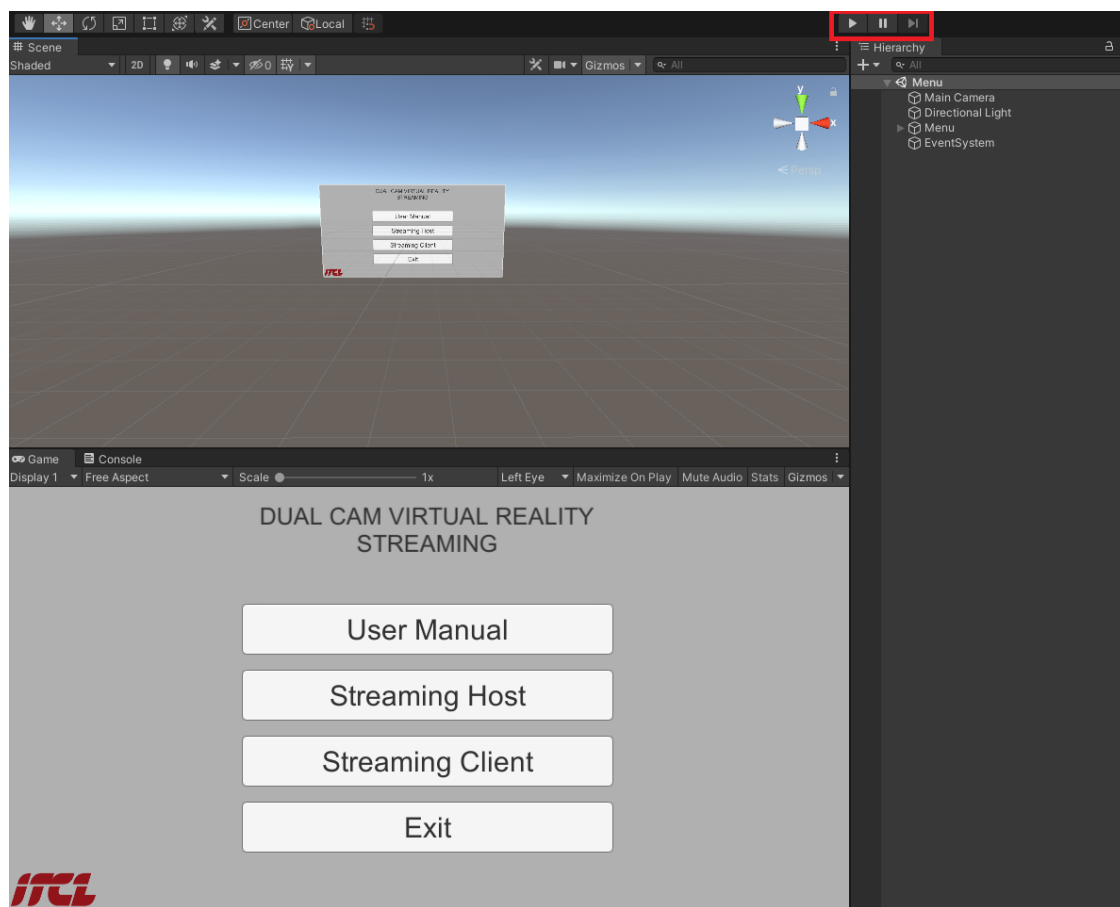


Figura D.8: Entorno de desarrollo Unity

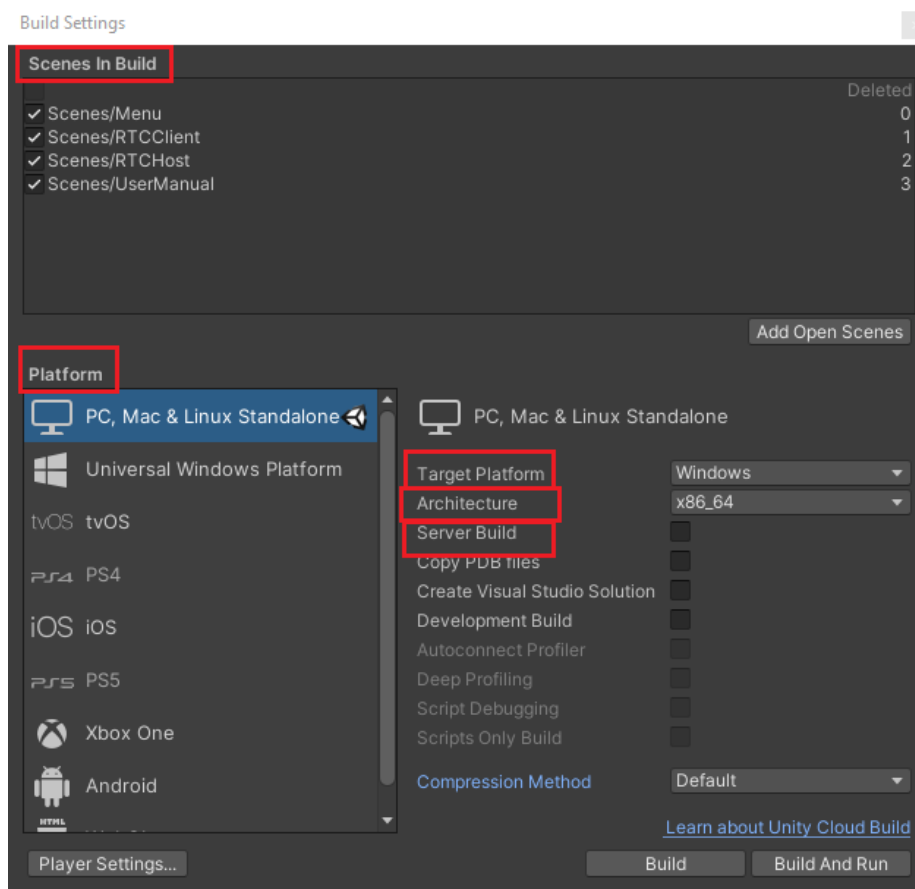


Figura D.9: Ventana Build Settings

- *Scenes In Build*: Campo en el que deberemos elegir las escenas que queremos incluir en la *build* de la aplicación.
- *Platform*: Lista de opciones que nos da a elegir la plataforma para la que queremos realizar la *build*.
- *Target Platform*: Dentro de la plataforma elegida, podremos elegir el tipo de la misma a través de este campo.
- *Architecture*: Tipo de arquitectura de sistema operativo que se desea dar a nuestra aplicación. En este caso siempre x86-64 o x64.
- *Server Build*: Campo que elimina la parte gráfica de una aplicación dejando sólo la funcionalidad que tenga implementada sin necesidad de acceder a objetos visuales. Muy útil para escenas que pretenden ser servidores de conexión. Reduce la cantidad de requisito que consume durante su ejecución.

D.6. Pruebas del sistema

Al hablar de pruebas, en este caso no ha habido una organización estricta puesto que la metodología principal que se ha seguido ha sido la de Ensayo-Error. Es decir, el único procedimiento al que se ha obedecido ha sido el de implementación de tareas y prueba de las mismas a medida que se ha ido creyendo conveniente, o que alguno de los objetivos se había logrado.

Apéndice *E*

Documentación de usuario

E.1. Introducción

La documentación de usuario es un elemento de consulta que irá dirigido a las personas que pretendan dar uso a la aplicación, o que quieran saber de antemano si ésta será capaz de satisfacer sus necesidades. Así mismo, es de utilidad para realizar un manejo básico o para profundizar si se posee cierto grado de conocimiento de uso de la herramienta.

E.2. Requisitos de usuarios

En cuanto a los requisitos de usuario, mencionaremos los programas o dispositivos que el usuario necesitará para poder explotar la funcionalidad de la aplicación desarrollada. Por esto diferenciaremos dos tipos de requisitos de usuario: Hardware y Software.

Requisitos Hardware

Como se ha indicado en el manual del programador, son necesarios una serie de requisitos para poder ejecutar el proyecto con las VIVE:

Por ende, el usuario debe poseer un dispositivo de realidad virtual *HTC VIVE* con todo su equipamiento e instalación pertinente. Además, será necesario poseer dos *webcams* de características idénticas que serán los ojos que retransmitirán las imágenes desde el lado del *host*.

Requisitos mínimos de <i>HTC VIVE</i> .	
Procesador	<i>Intel™ Core™ i5-4590</i> o <i>AMD FX™ 8350</i> , equivalente o superior.
Gráficos	<i>NVIDIA GeForce™ GTX 1060</i> o <i>AMD Radeon™ RX 480</i> , equivalente o superior.
Memoria	4GB RAM o superior.
Salida de vídeo	Un puerto HDMI 1.4 o <i>DisplayPort</i> 1.2 o superior.
USB	Un puerto USB 2.0 o superior.
Sistema operativo	<i>Windows</i> 7 SP1, 8.1, 10 o superior.

Tabla E.1: Requisitos mínimos de HTC VIVE

El hardware probablemente sea el factor limitante de uso de esta aplicación ya que el valor económico del dispositivo de realidad virtual, añadido al valor del hardware que requiere para su ejecución, es considerable.

Requisitos Software

El software necesario para poder disfrutar de la funcionalidad de la aplicación será el siguiente:

- Descarga e instalación de **Node.js**[8] de cara a poder ejecutar un señalizador[4] que también habrá que descargar para la conexión. Más concretamente, ambos extremos tendrán que instalar el *Node.js*, pero sólo uno de los dos (a elegir por el usuario) tendrá que abrir el señalizador que hará de servidor y permitirá la conexión.
- Será necesario instalar **Steam**[14] para poder descargar desde su aplicación de escritorio el programa que permitirá al ordenador reconocer el dispositivo de realidad virtual. Para ello será necesario poseer una cuenta de usuario de *Steam*.
- **Steam VR**[15] hará las funciones de configurador de las gafas *HTC VIVE* en el ordenador utilizado, por lo que su instalación también es imprescindible y se realizará desde la aplicación de escritorio de *Steam*.

E.3. Instalación

La instalación de la aplicación en un ordenador no conlleva ninguna complicación ni trabajo extra, como mucho la descompresión de la carpeta en

la que se encuentra. Después únicamente será necesario proceder a ejecutar el archivo con extensión .exe que se encuentre dentro de la carpeta y nuestra aplicación está en marcha.

No ocurrirá lo mismo con el hardware a utilizar en el extremo que pasará a adquirir el rol de cliente (o receptor de imágenes) puesto que este necesitará tener operativo todo el sistema de realidad virtual *HTC VIVE*, el cuál no sólo pide enchufar las gafas al ordenador.

Instalación de HTC VIVE

En este subapartado dejaremos una referencia a la página de soporte[18] de *VIVE* para realizar absolutamente todos los ajustes posibles que permite este equipo, pero procederé a explicar los ajustes esenciales que permiten disfrutar sin ningún problema de la funcionalidad completa de la aplicación:

- El primer paso a realizar será la instalación del software de *HTC VIVE*[16], a través del cual se instalarán otros programas de interés como son *Steam*[14] y dentro de éste *SteamVR*[15], agilizando notablemente las descargas necesarias para poner en funcionamiento las gafas. Además realizará cualquier tipo de actualización de *drives* o de las propias gafas si es necesario.
- Una vez puesto en regla el software necesario, se procede a la instalación de las estaciones de reproducción de espacio virtual, que tendrán la necesidad de estar enchufadas a la corriente, a 5 metros de distancia, enfocándose entre sí, y a 2 metros del suelo para poder realizar un buen seguimiento del terreno, las gafas y los controladores de RV. Cuando están instaladas, se procede a su conexión, que puede ser por cable, situación en la que será importante pulsar en ambas el botón trasero para que una tenga el canal **a** y la otra el canal **c**, o sin necesidad de cable, caso en el que pulsaremos el botón trasero en ambas estaciones de nuevo, hasta que una de ellas tenga el canal **b** y la otra el canal **c** y esperaremos hasta que se encuentren (encendiéndose una luz verde).
- Por último en lo referente a la instalación hardware sólo nos quedará conectar a una *linkbox* por un lado un cable de corriente, un cable *display-port* y un USB, estos dos últimos conectados al PC (y gráfica) por su otro extremo. Y a la otra parte de la *linkbox* le conectaremos los cables de las gafas, que serán otro USB, HDMI y un cable de sonido.

- Finalmente se ajustará el espacio que se va a utilizar, proceso que *SteamVR* nos facilitará mucho ofreciéndonos las opciones que se aprecian en la siguiente imagen E.3:

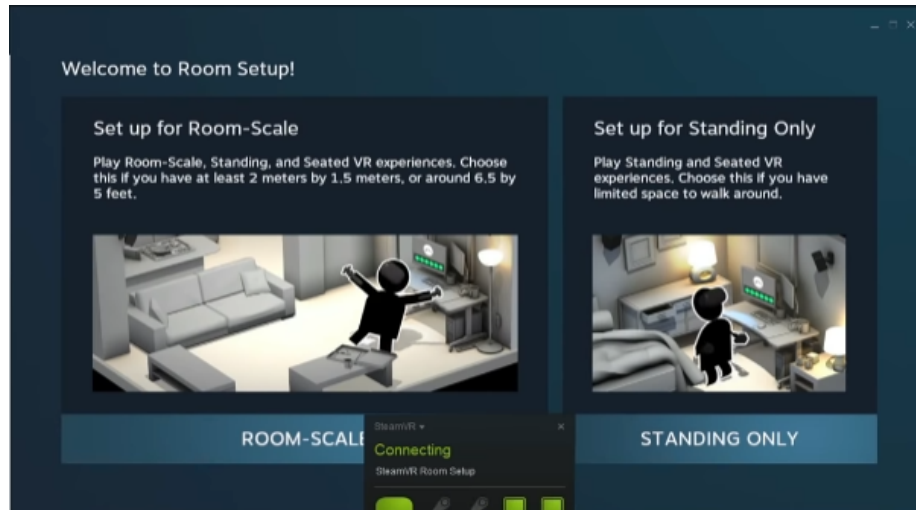


Figura E.1: Configuración del espacio de interacción

Instalación y ejecución de Node.js y NodeDSS

Este apartado ya está explicado en el manual del programador, por lo tanto dejaríamos como competencia del desarrollador, o en caso de que se comercializase, del instalador del sistema o de algún tipo de manual de ayuda la obligación de explicar los pasos necesarios para descargar e instalar estas dos herramientas[8] [4] que son imprescindibles para el correcto funcionamiento de la aplicación.

E.4. Manual del usuario

En este manual de usuario procederé a explicar de manera práctica qué se puede hacer y cómo dentro de la aplicación. De esta forma explicaremos procedimientos genéricos antes de entrar en cada una de las escenas propias de cada uno de los dos roles que se pueden desempeñar respectivamente: Emisor de imágenes y receptor de imágenes y coordenadas.

Primero el usuario (independientemente de su rol) se va a encontrar con una escena de menú E.4 con la siguiente estructura:

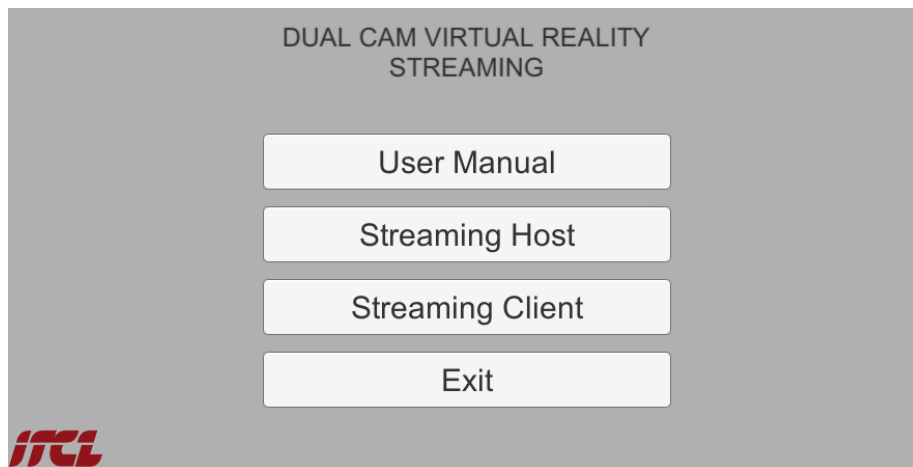


Figura E.2: Escena inicial de la aplicación: Menú

Aquí podremos acceder a:

- **User Manual:** Botón que activa una nueva escena con un pequeño manual de usuario que nos indica los pasos a seguir para poner en funcionamiento el sistema de visión.
- **Streaming Host:** Botón que activa una nueva escena que corresponde al emisor de imágenes. Explicaremos la funcionalidad en subapartados posteriores.
- **Streaming Client:** Botón que activa una nueva escena que corresponde al receptor de imágenes. Explicaremos la funcionalidad en subapartados posteriores.
- **Exit:** Botón que cierra la aplicación.

A continuación mostraremos los pasos a seguir indicados en el la escena *UserManual* E.4 para establecer de manera satisfactoria una conexión entre ordenadores y poner en marcha el sistema de visión estereoscópica:

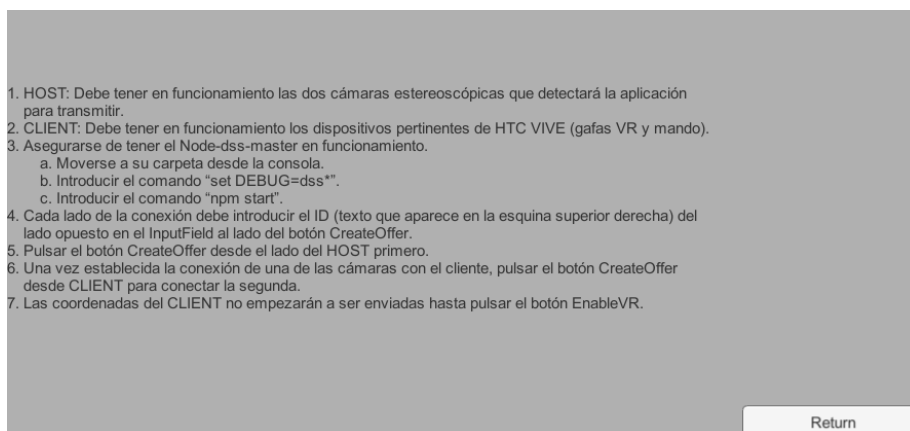


Figura E.3: Escena con un pequeño manual de funcionamiento para el usuario

Debido a que considero que está lo suficientemente claro en este pequeño manual, no repetiré el procedimiento por escrito.

Manual del receptor de imágenes

En el manual del receptor de imágenes mostraremos la escena en la que se realiza todo el proceso de recepción, muestra de imágenes, reproducción en realidad virtual y envío de coordenadas.

Una vez accedemos a esta escena nos encontramos con la siguiente imagen **E.4:**

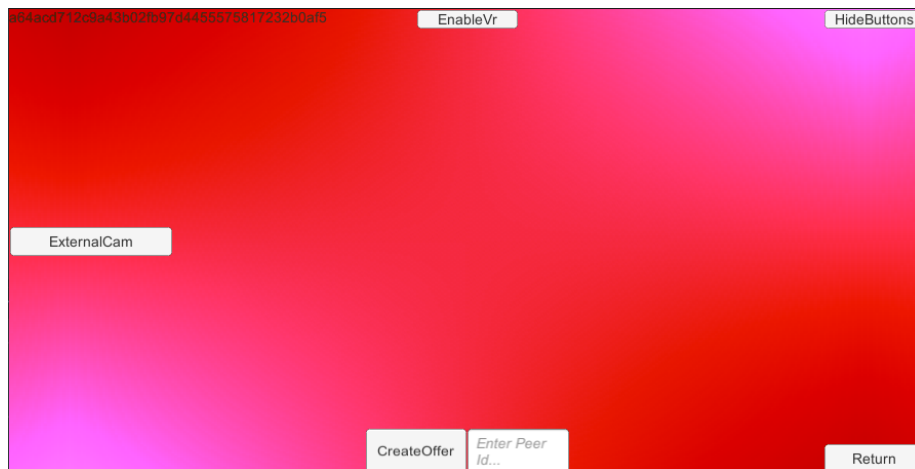


Figura E.4: Escena del receptor de imágenes

A partir de esta visión de la escena, destacamos varios puntos:

- En la esquina superior izquierda se encuentra el ID que deberá ser introducido en la escena del emisor para poder realizar la conexión.
- **Create Offer:** botón que lanzará un mensaje a través del servidor para proponer la retransmisión de imágenes.
- **Peer ID:** campo dónde se introducirá el ID de la escena del emisor.
- **Return:** botón que nos permite volver al menú inicial.
- **Hide Buttons:** botón que esconde todos los botones a excepción de él mismo para volver a mostrarlos de nuevo si se desea (es una implementación opcional).



Figura E.5: Vista con el botón HideButtons activado

- **EnableVr:** botón que activa el dispositivo de realidad virtual y todas sus funcionalidades. La apreciación en pantalla es prácticamente nula (sólo cambia el botón *ShowButtons*), pero aún así mostraremos la imagen. Para volver al estado anterior podemos pulsar el botón Escape.



Figura E.6: Vista con el botón EnableVr activado

- **ExternalCam:** botón que activa una vista externa de ambas cámaras para saber si se están reproduciendo dos imágenes diferentes. Para volver al estado anterior podemos pulsar el botón Escape.

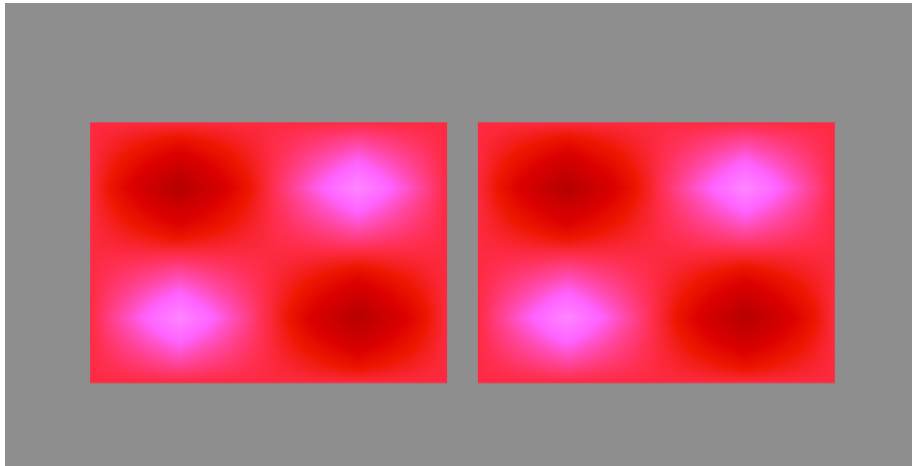


Figura E.7: Vista con el botón ExternalCam activado

Ahora que ya hemos analizado la escena sin vídeo entrante, procederé a mostrar un extracto del vídeo ?? de prueba dónde se aprecia la recepción de dos imágenes diferentes:



Figura E.8: Vista de dos imágenes diferentes con el botón ExternalCam activado

Manual del emisor de imágenes

De igual manera que en el subapartado anterior, en este manual mostraremos la escena con sus funcionalidades, pero en este caso se tratará de la correspondiente al emisor de imágenes.

Una vez accedemos a esta escena nos encontramos la siguiente imagen [E.4](#):

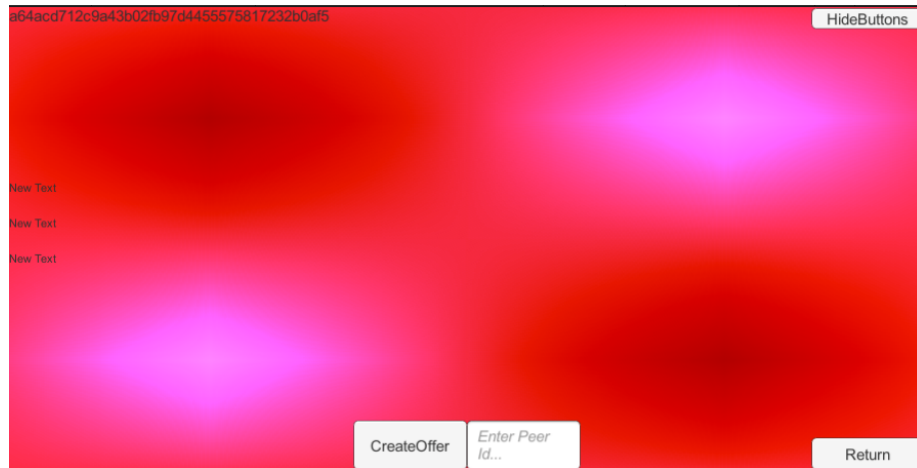


Figura E.9: Escena del emisor de imágenes

A partir de esta visión de la escena, destacamos varios puntos:

- En la esquina superior izquierda se encuentra el ID que deberá ser introducido en la escena del receptor para poder realizar la conexión. En este caso es el mismo porque ambas capturas están tomadas desde el mismo ordenador.
- En el lado izquierdo, a media altura, observamos 3 líneas con las palabras *New Text*. Estas tres líneas de texto son capaces de recibir de manera simultánea las coordenadas de posición de las gafas de RV, las coordenadas de rotación de las gafas de RV y las coordenadas de posición de uno de los controladores manuales (mando) del equipo *HTC VIVE*. En este ejemplo únicamente está activada la recepción de coordenadas de rotación, que son las deseadas por la empresa.
- **Create Offer:** botón que lanzará un mensaje a través del servidor para proponer la retransmisión de imágenes.
- **Peer ID:** campo dónde se introducirá el ID de la escena del receptor.
- **Return:** botón que nos permite volver al menú inicial.
- **Hide Buttons:** botón que esconde todos los botones a excepción de él mismo para volver a mostrarlos de nuevo si se desea(es una implementación opcional).

En este caso no hay un botón que confirme que ambas *webcams* estén siendo reconocidas por lo que habría que iniciar la escena teniendo una única cámara conectada, y después repetir el proceso con la otra, comprobando que se retransmite la imagen por pantalla. No obstante mostraremos un ejemplo de la escena reproduciendo imagen, pero primero enseñaremos una captura de la escena en el framework E.4 para confirmar que existen 2 paneles de retransmisión:

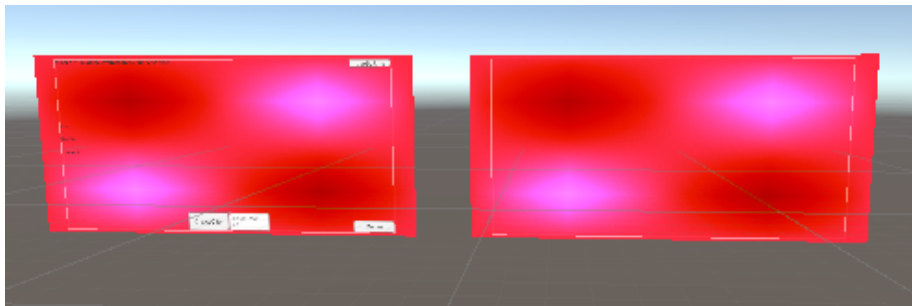


Figura E.10: Escena del emisor de imágenes en el framework

Por último, mostraremos un ejemplo de la misma escena captando una *webcam* y con la recepción de coordenadas de rotación activada. Esta imagen E.4 también es un extracto del vídeo de prueba realizado en ITCL:

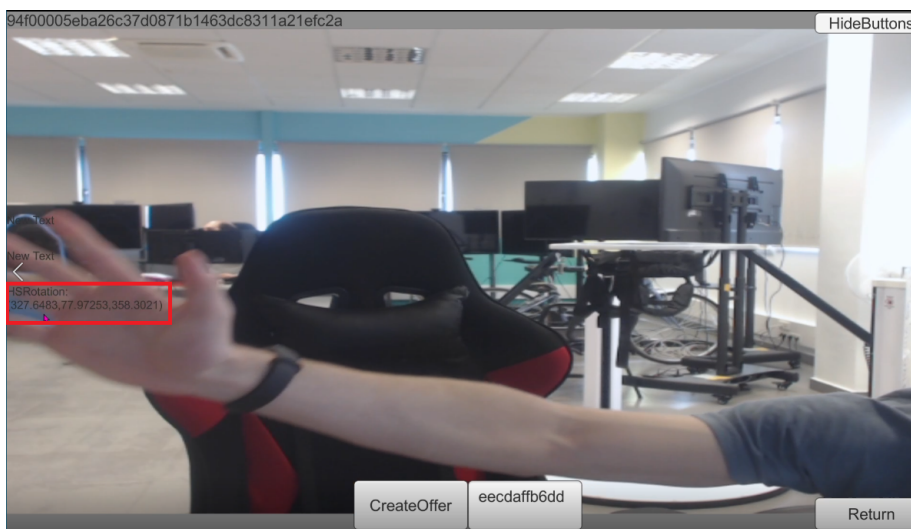


Figura E.11: Escena del emisor de imágenes en funcionamiento

Recomendaciones de uso

Basándome en la experiencia personal me permitiré el lujo de realizar un par de recomendaciones:

- Nunca utilizar el sistema de visión más de 30 minutos seguidos, puede alterar ciertos aspectos sensoriales del cuerpo humano que, a pesar de no mostrar ningún tipo de efecto durante el uso, posteriormente pueden generar malestar, dolor de cabeza, náuseas, etc.
- Si algún de los síntomas mencionados en el punto anterior comienza a aparecer en el usuario, por favor, no valore la posibilidad de seguir usándolo a la espera de que desaparezcan, porque aun fuera posible, la sintomatología suele empeorar en vez de ir a mejor.

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] apiumhub.com. Documentación técnica: herramientas y consejos. <https://apiumhub.com/es/tech-blog-barcelona/herramientas-documentacion-tecnica/>, 2021.
- [3] GitKraken. Página oficial de gitkraken. <https://www.gitkraken.com/>, 2021.
- [4] Ben Greenier. Node-dss. <https://github.com/bengreenier/node-dss>, 2019.
- [5] ITCA Fepade ingenieros técnicos. Manual del programador. https://virtual.itca.edu.sv/Mediadores/ads/136_manual_del_programador.html, 2021.
- [6] JetBrains. Página oficial de rider. <https://www.jetbrains.com/es-es/rider/>, 2021.
- [7] Open JSFoundation. Página oficial de node.js. <https://nodejs.org/es/download/>, 2021.
- [8] Open JSFoundation. Página oficial de node.js. <https://nodejs.org/es/>, 2021.
- [9] Microsoft. Página oficial de visual studio. <https://visualstudio.microsoft.com/es/>, 2021.

- [10] Google Sites. Técnicas para identificar irequisitos funcionales y no funcionales. <https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales>, 2021.
- [11] Unity. Pagina de descargas de unity. <https://unity3d.com/es/get-unity/download>, 2021.
- [12] Unity. Pagina oficial de unity. <https://unity.com/es>, 2021.
- [13] Unity. Unity - scripting api: Scenemanager. <https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.html>, 2021.
- [14] Valve. Página oficial de steam. <https://store.steampowered.com/>, 2021.
- [15] Valve. Página oficial de steamvr. <https://www.steamvr.com/es/>, 2021.
- [16] VIVE. Set up your vive | vive europe. <https://www.vive.com/eu/setup/>, 2021.
- [17] VIVE. Vive european union | discover virtual reality beyond imagination. <https://www.vive.com/eu/>, 2021.
- [18] VIVE. Vive support. <https://www.vive.com/eu/support/vive/>, 2021.
- [19] Wikipedia. Diseño de software — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Dise~no_de_software#Conceptos_de_dise~no, 2021.
- [20] Wikipedia. Requisito funcional — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Requisito_funcional, 2021.
- [21] Wikipedia. Requisito no funcional — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Requisito_no_funcional, 2021.
- [22] ZenHub. Track sprint progress with burndown charts. <https://help.zenhub.com/support/solutions/articles/43000010356-track-sprint-progress-with-burndown-charts>, 2021.