



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA EN INFORMÁTICA

MedieVRal

Interacción en VR con HTC Vive

Autor

Juan Alberto Martínez López

Director

Marcelino José Cabrera Cuevas



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, 7 de septiembre de 2018



MedieVRal

Interacción en VR con HTC Vive

Autor

Juan Alberto Martínez López

Directores

Marcelino José Cabrera Cuevas

MedieVRal: Interacción en RV con HTC Vive

Juan Alberto Martínez López

Palabras clave: Realidad Virtual, Controlador, Kit de desarrollo de software, Assets, Game Object, Transform, Material, Textura, Prefabricado, Colisionador, Trigger, Grip y Touchpad.

Resumen

El propósito principal de este proyecto es desarrollar un estudio de las diferentes maneras que el dispositivo HTC Vive permite al usuario interactuar con el entorno virtual, a través de un videojuego en el cual se desarrollan diferentes pruebas ambientadas en la era medieval, en la que cada una desarrolla un aspecto de la interacción, viendo cuales implementan una mayor inmersión del jugador y cuales aportan factores negativos al mismo.

Para ello, se le dará a elegir al usuario entre siete pruebas, las cuales pueden ser accedidas desde una sala inicial y proporcionarán una puntuación al jugador dependiendo de la realización de las mismas. Dichas pruebas disponen de diferentes objetivos, permitiendo al usuario abandonarlas y volver a la sala inicial en cualquier momento.

Por último, en la sala principal se almacenará un seguimiento del jugador en dichas pruebas y cuando todas hayan sido realizadas se calificará al jugador en función de la puntuación adquirida.

MedieVRal: Interaction in VR with HTC VIVE

Juan Alberto Martínez López

Keywords: Virtual Reality, Controller, Software Development Kits, Assets, Game Object, Transform, Material, Texture, Prefabricated, Collider, Trigger, Grip and Touchpad

Abstract

The main purpose of this project is to develop a study of the different ways that the HTC Vive device allows the user to interact with the virtual environment, through a videogame in which different tests, which is based in medieval age, are developed. Each one develops an aspect of the interaction, seeing which implement a greater immersion of the player and which contribute negative factors to it.

To that end, the user will be given the choice between seven tests which can be accessed from an initial room and provide a score to the player depending on the performance of the same. These tests have different objectives allowing the user to abandon them and returning him to the initial room at any time.

Finally, in the main room will be stored a tracking of the player in these tests and when all have been made the player will be scored based on the score acquired.

Yo, **Juan Alberto Martínez López**, alumno de la titulación TITULACIÓN de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, con DNI 77145788Q, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Juan Alberto Martínez López

Granada a 18 de junio de 2018.

D. **Marcelino José Cabrera Cuevas**, Profesor del Área de Lenguajes y Sistemas Informáticos del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***MedieVRal, Interacción en VR con HTC Vive***, ha sido realizado bajo su supervisión por **Juan Alberto Martínez López**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 7 de septiembre de 2018.

El director:

Marcelino José Cabrera Cuevas

Índice general

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Entorno Operacional	2
1.3.1. Hardware	2
1.3.2. Software	7
2. Especificación de requisitos	11
2.1. Introducción	11
2.1.1. Definiciones, acrónimos y abreviaturas	11
2.1.2. Visión general del documento	12
2.2. Descripción del sistema	13
2.2.1. Selección de niveles	13
2.2.2. Niveles	13
2.3. Especificación de casos de uso	14
2.3.1. Casos de Uso	14
2.4. Especificación de Requisitos	20
2.4.1. Requisitos Funcionales	21
2.4.2. Requisitos No Funcionales	23
3. Planificación	25
3.1. Metodología Ágil	25
3.2. Diagrama de Gantt	28
3.3. Herramientas utilizadas	28
3.3.1. Github	28
3.3.2. Trello	29
3.4. Elección del Motor Gráfico	30
3.5. Restricciones del sistema	31
3.5.1. Restricciones Hardware	31
3.5.2. Restricciones Software	31

4. Análisis	33
4.1. Factores Inmersivos	33
4.1.1. Factores inmersivos directos	33
4.1.2. Factores inmersivos indirectos	34
5. Diseño	37
5.1. Diseño del juego: MediVRal	37
5.1.1. Resumen	37
5.1.2. Jugabilidad	37
5.1.3. Desarrollo del juego	38
5.2. Diagramas de clases	38
5.2.1. Selector de niveles	38
5.2.2. Menú de Usuario	39
5.2.3. Niveles	40
6. Implementación	53
6.1. Inicialización de HTC Vive	53
6.2. Mecanismos de Interacción	55
6.3. Contenido y funcionalidades	56
6.3.1. Selector de niveles	56
6.3.2. Throwing	58
6.3.3. Catapult	59
6.3.4. Shooting	61
6.3.5. Jousting	63
6.3.6. Bow	64
6.3.7. Climbing	66
6.3.8. Shield	68
6.4. Problemas resueltos	69
6.4.1. Colisiones en movimiento	69
6.4.2. Cambiar y repetir el nivel	71
7. Pruebas de usabilidad	73
7.1. Pruebas realizadas	73
7.2. Conclusiones sobre las pruebas realizadas	74
8. Conclusiones y trabajos futuros	77
8.1. Conclusiones	77
8.2. Trabajos futuros	78
Bibliografía	81

Índice de figuras

1.1.	Distintos componentes de las gafas HTC Vive	3
1.2.	Distintos botones de los mandos HTC Vive	4
1.3.	Zona de juego recomendada por HTC.	5
1.4.	Marcas de RV.	6
1.5.	Gafas Oculus Rift.	6
1.6.	Gafas PlayStation VR.	7
2.1.	Diagrama de casos de uso de selección de nivel.	15
2.2.	Diagrama de casos de uso de cada nivel.	16
3.1.	Diagrama de Gantt.	28
3.2.	Utilización de la aplicación Github.	29
3.3.	Utilización de la aplicación Trello en el sprint VG_1.	30
5.1.	Diagrama de clases de la escena de Selección de nivel.	39
5.2.	Diagrama de clases del menú de usuario.	40
5.3.	Diagrama de clases de la primera prueba (Trowing).	42
5.4.	Diagrama de clases de la segunda prueba (Catapult).	43
5.5.	Diagrama de clases de la tercera prueba (Shooting).	45
5.6.	Diagrama de clases de la cuarta prueba (jousting).	46
5.7.	Diagrama de clases de la quinta prueba (Bow).	48
5.8.	Diagrama de clases de la sexta prueba (Climbing).	50
5.9.	Diagrama de clases de la séptima prueba (Shield).	51
6.1.	Configuración del script <i>VRTK_SDK Setup</i>	54
6.2.	Configuración del script <i>VRTK_SDK Manager</i>	55
6.3.	Jerarquía de GameObjets final.	55
6.4.	Escena de selección de nivel.	56
6.5.	Teletransportación a un destino.	57
6.6.	Esfera de selección de nivel.	57
6.7.	Prueba Throwing.	58
6.8.	Objetivo alcanzado por daga.	59
6.9.	Prueba Catapult.	59
6.10.	Catapulta cargada y apuntando al objetivo.	60

6.11. Explosión del objetivo alcanzado.	61
6.12. Prueba Shooting.	61
6.13. Objetivo alcanzado por bala.	62
6.14. Objeto gunpowder cargando el arma.	63
6.15. Prueba Jousting.	63
6.16. Objetivo de Jousting.	64
6.17. Prueba Bow.	65
6.18. Arco tensado.	66
6.19. Objetivo alcanzado por flecha.	66
6.20. Prueba Climbing.	67
6.21. Objetos BombHip y Bomb.	67
6.22. Objetos BombSlot y ClimbableBricks.	68
6.23. Prueba Shield.	68
6.24. Objeto ArrowProjectile clavada en el escudo.	69
6.25. Lanzamiento fallido.	71
6.26. Lanzamiento en el blanco.	71
6.27. Menú del usuario.	72
1. SteamVR en el menú herramientas.	84
2. Pantalla de Inicio en Unity.	85
3. Asset de SteamVR.	86
4. Puntos de destino bloqueados.	87
5. Menú del usuario.	87
6. Prueba de lanzamiento de objetos.	88
7. Prueba de catapulta.	89
8. Prueba de armas de fuego.	90
9. Disparando el arma.	90
10. Prueba de disparo con arco.	91
11. Disparando a un objetivo.	91
12. Prueba de escalada.	92
13. Objetos escalables.	92
14. Prueba de bloqueo con escudo.	93
15. Señal de ataque.	93
16. Prueba de justa.	94
17. Ejemplo de globos.	94

Índice de cuadros

1.1. Tabla Resumen	7
2.1. Actor ACT-1.	14
2.2. Caso de uso CU-01.	16
2.3. Caso de uso CU-02.	17
2.4. Caso de uso CU-03.	17
2.5. Caso de uso CU-04.	17
2.6. Caso de uso CU-05.	18
2.7. Caso de uso CU-06.	18
2.8. Caso de uso CU-07.	18
2.9. Caso de uso CU-08.	19
2.10. Caso de uso CU-09.	19
2.11. Caso de uso CU-10.	19
2.12. Caso de uso CU-11.	20
2.13. Caso de uso CU-12.	20
2.14. Requisito funcional RF-01.	21
2.15. Requisito funcional RF-02.	21
2.16. Requisito funcional RF-02.	21
2.17. Requisito funcional RF-04.	21
2.18. Requisito funcional RF-05.	22
2.19. Requisito funcional RF-06.	22
2.20. Requisito funcional RF-07.	22
2.21. Requisito funcional RF-08.	22
2.22. Requisito funcional RF-09.	22
2.23. Requisito no funcional RNF-01.	23
2.24. Requisito no funcional RNF-02.	23
2.25. Requisito no funcional RNF-03.	23
2.26. Requisito no funcional RNF-04.	23
2.27. Requisito no funcional RNF-05.	24
3.1. Aprendizaje sobre la librería VRTK.	26
3.2. Análisis de Requisitos y elección de motor gráfico.	26
3.3. Diseño, implementación y pruebas del videojuego.	27

3.4. Realización de la memoria.	28
7.1. Resultados de las pruebas sobre los usuarios.	74

Capítulo 1

Introducción

1.1. Motivación

Hoy en día aun se está consolidando la tecnología Realidad Virtual al uso diario de los usuarios y aunque todo lo que se ha desarrollado en RV permite un gran avance en la utilización de la tecnología aun no se ha alcanzado una gran parte del potencial que tiene.

Si pudiéramos ponernos unas gafas y poder viajar a otro mundo, un lugar remoto donde rigieran las mismas leyes de la gravedad, un sitio donde nos moviéramos de manera exacta a como lo hacemos aquí y pudiésemos interactuar virtualmente con todo lo que nos rodea. HTC Vive consigue sacar todos los mundos virtuales que vemos a través de la pantalla del ordenador y nos permite vivir en vivo esos mundos. Ser capaz de transportarnos a otros tiempos y lugares, permitiéndonos explorarlos como si estuviéramos nosotros ahí. Y no solo eso sino que incluso nos permite alcanzar situaciones inalcanzables por el hombre promedio, como sería andar por un transbordador que órbita la tierra, o viajar a realidades totalmente desconocidas con posibilidades ilimitadas.

Para ello las gafas HTC Vive cuentan con una variedad de sensores para que el usuario interaccione con el dispositivo entre los cuales se encuentran: el giroscopio de las gafas y de los mandos que nos permiten saber la dirección a la que apuntan; los link box que nos permiten saber la posición de las gafas y de los mandos; y la vibración de las gafas y de los mandos.

Gracias a estos componentes se nos permite diseñar e implementar mundos virtuales desde cero en los que nosotros dictamos las reglas de ese mundo. Por eso la propuesta de videojuego a desarrollar en este proyecto pretende

dar rienda suelta a la creatividad para conseguir un mundo inmersivo en el que el jugador se sienta cómodo con las interacciones y físicas que ofrece.

1.2. Objetivos

En este proyecto se quiere hacer un estudio de la inmersividad con el equipo de RV HTC Vive pretendiendo introducir al usuario en las distintas interacciones que se pueden realizar a través de distintas pruebas en las que el propio usuario interactuará con distintos objetos directa o indirectamente aplicando físicas y respuestas ante el medio para resolver situaciones de manera que sea inmersivo y realista. Los objetivos concretos que persigue este proyecto son:

1. Realizar un estudio de la herramienta Unity y de las funcionalidades del paquete VRTK 8.2.
2. Experimentar, con la herramienta Unity, los distintos ejemplos de la librería VRTK.
3. Hacer un estudio sobre los efectos inmersivos en la Realidad Virtual.
4. Diseñar un videojuego basado en el estudio realizado.
5. Desarrollar un prototipo del videojuego propuesto.
6. Hacer una prueba de usabilidad del videojuego sobre un grupo de usuarios para comprobar el efecto inmersivo sobre los mismos.
7. Crear una documentación técnica y memoria final del proyecto.

1.3. Entorno Operacional

Antes de comenzar con el proyecto en sí se definirán, los componentes Hardware del dispositivo HTC Vive, una introducción a los diferentes assets y software, que se usarán a lo largo del proyecto.

1.3.1. Hardware

HTC VIVE

HTC Vive[1] son unas gafas de realidad virtual fabricadas por HTC y Valve. Vive proporciona una solución a la realidad virtual en la que offre-

ce una experiencia inmersiva desde un punto físico móvil (nos detecta el movimiento en una habitación mientras andamos por ella) y permitiendo utilizar los controladores (con los distintos botones) para que las manos formen parte del juego en tiempo real, facilitando al usuario ver reflejados los movimientos, rotaciones y acciones.

Componentes

HTC Vive[1] cuenta con un sistema de periféricos (1.1) que conectan las gafas al ordenador y que permiten detectar su posición. Los distintos componentes son:

- Sensores de posición (*Base Station*).
- Gafas de RV.
- Controladores.



Figura 1.1: Distintos componentes de las gafas HTC Vive

Especificaciones

Tanto las gafas como los controladores cuentan con acelerómetro, giroscopio, doble sistema de posición láser además de las siguientes especificaciones:

- Las Gafas HTC[1] Vive cuenta con 36 sensores instalados, permitiendo un seguimiento de las mismas sin ángulos muertos, una cámara frontal con un campo de visión de 110°, una tasa de refresco de imagen de 90 fps (*frames per second*) y una resolución de 2160 x 1200 píxeles.

- Los Controladores (1.2) cuentan con 24 sensores cada uno permitiendo un *feedback* en tiempo real del movimiento y de los distintos botones del mando: Trigger (el gatillo), Touchpad (el panel tactil), Grip (los botones laterales), el botón de menú de aplicación y el botón de menú de sistema.

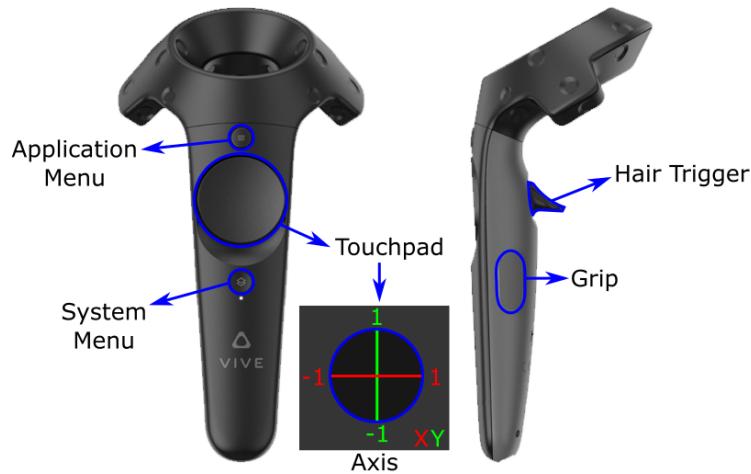


Figura 1.2: Distintos botones de los mandos HTC Vive

- Los Sensores de posición (*Base stations*) son dos cubos que necesitan conexión a enchufe y visión directa entre ellos. Estos sensores deben cubrir la zona de juego de manera que cada uno debe estar en un extremo de la sala.

Requisitos

Para que las HTC Vive funcionen correctamente es necesario un ordenador de gama alta como mínimo con las siguientes prestaciones.

- Tarjeta gráfica: NVIDIA Asus Dual GTX 1060 3GB (para tener una experiencia fluida).
- CPU: Intel Core i5 4590 .
- RAM: 8 GB .
- Un puerto HDMI.

- Un puerto USB 2.0 .
- Sistema Operativo: Windows 7 o superior.

Valve proporciona un programa[2] que evalúa las características de un ordenador e informa de si el ordenador es capaz de soportar las HTC Vive o necesita ampliar un componente en concreto.

Configuración

Para que las gafas HTC Vive funcionen correctamente[3] necesitarán de 3 enchufes (2 para las *Base Stations* y 1 para las gafas). Primero hay que colocar las *Base Stations* en un lugar alto y que se vean entre ellas y conectar las gafas al ordenador a partir de la *Link Box*.

Usando el **Setup** proporcionado por Vive[3] nos permite crear una *zona de juego*[3], la cual sera el área delimitadora que se nos mostrará, dentro del juego, si nos acercamos a los límites de la misma e instalará steamVR para poder sincronizar las gafas con el ordenador.

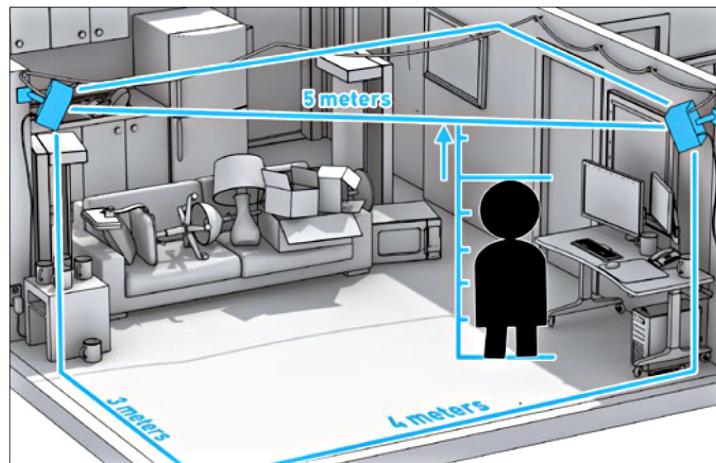


Figura 1.3: Zona de juego recomendada por HTC.

Por último, una vez todo está conectado y sincronizado, podemos hacer un tutorial que nos ofrece *Steam* para introducirnos al uso de las HTC Vive.

Mercado Actual sobre la RV

Cada vez hay más opciones en el mercado que nos permiten disfrutar de videojuegos y contenidos inmersivos, que nos hacen vivir la experiencia como si estuviéramos allí. Ya tengamos una consola, ya tengamos un ordenador, o incluso a través de nuestro smartphone. Actualmente la competencia en el terreno de la realidad virtual viene dada por tres grandes marcas: HTC Vive, Oculus y PlayStation VR.



Figura 1.4: Marcas de RV.

Oculus [4] fue la primera empresa en empezar a desarrollar prototipos de RV. Oculus ronda el mismo precio que las HTC Vive y es parecido en componentes, los mandos y los sensores, pero requiere de un ordenador menos potente para funcionar por lo que a usuarios con un ordenador de peores características les puede interesar mas. Además, Oculus permite trabajar en espacios mas cerrados gracias a los sensores que tiene y permite utilizar un mando, pero en espacios amplios HTC Vive es una mejor opción.



Figura 1.5: Gafas Oculus Rift.

PlayStation VR [5] esta limitada a la PlayStation 4 lo que dificulta a los usuarios obtener los dispositivos necesarios, aunque las gafas son mas

económicas que las Oculus y las HTC Vive y solo necesitaría una PlayStation 4 a la que conectar el casco, la cual es mas barata que un ordenador potente. En cuestión de componentes no llega a alcanzar a sus competidores pero dado que es mas económica que las otras dos puede llegar a captar a usuarios que no quieran gastarse una gran cantidad de dinero.



Figura 1.6: Gafas PlayStation VR.

	Pros	Contras
Oculus Rift	Casi las mismas características por menos precio. Requiere de un ordenador menos potente. Permite trabajar en espacios mas cerrados.	En espacios amplios no reconoce bien la posición del jugador. Los controladores son menos precisos.
PlayStation VR	Son mucho mas económicas. Permiten acceder a contenido exclusivo de PlayStation 4.	Necesita la PlayStation 4 para funcionar. Los componentes no alcanzan a sus competidores. Utiliza el mando de la PlayStation en vez de controladores.

Cuadro 1.1: Tabla Resumen.

1.3.2. Software

SteamVR

Steam Virtual Reality [6] es una plataforma de Realidad Virtual desarrollada por Valve como una extensión de la plataforma Steam. La extensión de

SteamVR para Unity proporciona la SDK y un conjunto de librerías para el correcto funcionamiento de las HTC Vive. La configuración del paquete SteamVR para su uso en una escena con HTC Vive se encuentra en el manual del usuario 8.2.

VRTK

Virtual Reality ToolKit [7] es una librería orientada al desarrollo de aplicaciones en VR. Contiene una serie de scripts y ejemplos que facilitan la creación de escenarios e interacciones en Unity con VR. Esta librería está preparada para ser ejecutada y compilada en el lenguaje de C# compatible con el motor de juego Unity 3D.

Conceptos

En VRTK existen una serie de conceptos básicos:

- Play Area: Zona donde el jugador se puede mover sin usar mecanismos de locomoción.
- Controller Events: Son los eventos generados al pulsar un botón de los controladores.
- Grabbable Object: Son los objetos que pueden ser agarrados por el usuario.
- Usable Object: Son los objetos que pueden ser utilizados y realizan alguna acción.
- Pointers: Son algunos de los mecanismos usados para establecer un destino a la que el jugador se puede teletransportar e interaccionar con objetos.
- UI (User Interface): Son interfaces de usuario accesibles por medio de los controladores.
- Grab Attach Mechanics: Son los distintos mecanismos para heredar el movimiento de un objeto padre.
- Locomotion: Son los distintos métodos de movilidad creados para extender la Play Area del jugador.

Assets Utilizados

Para este proyecto se ha dispuesto de varios *assets* tanto proporcionados por Unity, como Unity community o páginas de diseño 3D. Assets que se encuentran en la web store de Unity:

- Free Demo Rock: Modelo de roca. Licence owner Grumpntug. Se utiliza en la prueba de catapulta como modelo de munición de roca.
- Medieval Barrows and Wagons: Modelos de carretas. Licence owner Kiria. Se utiliza en todas las escenas como carretas de madera de decoración.
- Steel Dagger PBR: Modelo de cuchillo. Licence owner Johnny's Place. Se utiliza en la prueba de lanzamiento como los objetos lanzables.
- UF Creator LITE: Modelo de campamento. Licence owner Aquarius Max. Se utiliza como decoración de un campamento en todas las escenas.
- Medieval Stone Keep : Modelo de muros y torres. Licence owner Lylek Games. Se utiliza como decoración de un castillo en la mayoría de las escenas.
- Water : Modelo de agua. Licence owner Unity. Se utiliza como modelo de agua en la mayoría de pruebas. Este es un paquete básico gratuito que la herramienta pone a disposición del usuario a través de la pestaña assets ¿import package¿environment.

Assets utilizados de paginas de diseño 3D:

- Medieval catapult 3d model [8]. Licence owner axe_93. Se utiliza como modelos de catapulta en la prueba de la catapulta.
- Flintlock pistol 3d model [9]. Licence owner Arms Museum. Se utiliza como modelo de pistola en la prueba de disparo con arma.
- Cannon Ball 3d model [10]. Licence owner ZacLinMac. Se utiliza como modelo de bala en la prueba de disparo con arma.
- Horse with Saddle 3D Model [11]. Se utiliza como modelo de caballo en la prueba de justa.
- Rings [12]. Se utiliza como modelos de objetivos en la prueba de justa.

- 3D Simple Lance [13]. Licence owner Vertici. Se utiliza como modelo de lanza en la prueba de justa.
- Bomb 3D model [14]. Licence owner evilvoland. Se utiliza como modelo de bomba en la prueba de escalada.
- Old Damaged Shield [15]. Licence owner Pino4et. Se utiliza como modelo de escudo en la prueba de escudo.
- Quiver and arrow [16]. Licence owner 3dregenerator. Se utiliza como modelo de flechas y carcaj en las pruebas de tiro con arco y escudo.
- Short rope 3D model [17]. Licence owner svyart. Se utiliza como modelo de cuerda en la prueba de escalada.
- Sand Pile [18]. Licence owner CC Attribution. Se utiliza como modelo de pólvora para el jarrón en la prueba de disparo con arma.

Capítulo 2

Especificación de requisitos

Esta especificación de requisitos contiene los **requisitos iniciales** apor-tados desde el inicio del proyecto y conforme ha ido avanzando este proyecto se han añadido más.

2.1. Introducción

2.1.1. Definiciones, acrónimos y abreviaturas

- **ERS:** Especificación de Requisitos Software.
- **API** (*Application Programming Interface*): Conjunto de funciones y procedimientos que ofrece alguna librería como capa de abstracción.
- **SDK** (*Software Development Kit*): Conjunto de herramientas que per-miten ejecutar uno o varios software en un sistema concreto.
- **VRTK** (*Virtual Reality Toolkit*): Librería de interacción con las gafas HTC Vive que se usará en el proyecto.
- **Escena:** Una escena que contiene los GameObject y menús de un juego.
- **PlayArea:** Zona designada por el jugador en la habitación física como jugable.
- **Blueprint:** Interfaz basada en nodos, que nos permite manejar los elementos del juego sin tener que programar código.
- **GameObject:** Objeto creado en una escena con una serie de compo-nentes y una transformación.
- **Component:** Elementos que se le pueden asociar a un GameObject.

- **Transformación:** Conjunto de datos en los que para cada objeto se tiene una posición, una rotación y un escalado en XYZ con un valor determinado.
- **Prefab:** GameObject almacenado que conserva todos los componentes y propiedades.
- **Collider:** Componente que detecta cuando intersectan la malla de colisión asociada a un objeto y la de otro.
- **Material:** Componente utilizado para renderizar texturas, sistemas de partículas o una malla.
- **Textura:** Es un archivo de imagen que se aplica en un material para asociar dicha textura a una malla.
- **IEnumerator:** Son corrutinas que se ejecutan paralelamente al script que las a lanzado.
- **Assets:** Representación de cualquier objeto que puede ser usado en un proyecto.
- **Ray casting:** Técnica de *Direct Volume Rendering* utilizada para la visualización de volúmenes. En este proyecto se utilizará como ayuda en las físicas para detectar la colisión de objetos.
- **GameManager:** Programa que se encarga de gestionar el cambio entre pruebas y guarda la puntuación del player para cada prueba.
- **GameController:** Programa que se encarga de gestionar el funcionamiento de una prueba del juego.

2.1.2. Visión general del documento

Este capítulo consta de tres secciones:

- En la primera sección se incluirá una descripción del sistema.
- En la segunda sección se realizarán los diagramas de caso de uso.
- En la tercera sección se realizará la especificación de requisitos.

2.2. Descripción del sistema

2.2.1. Selección de niveles

Esta escena consiste en un campamento donde nos podremos teletransportar a varios destinos donde vamos encontrando una serie de esferas de cristal con imágenes, que cuando las cogemos y nos las acercamos a las gafas nos llevan a las distintas pruebas.

2.2.2. Niveles

- Lanzamiento de objetos: Consiste en lanzar un objeto, agarrible por el player, a distintas dianas desde una zona del mapa. Al lado del player se encontrará una caja de munición la cual le permite coger munición pulsando un botón del controlador. Los objetos dianas se encuentran a una distancia donde el jugador no puede llegar para darles con el objeto en la mano y tiene que lanzarlo. Cuando dichas dianas son golpeadas por el objeto o pasa un tiempo límite desaparecen. Cuantas más dianas golpee el usuario más puntos conseguirá.
- Catapulta: La tarea consiste en lanzar munición con una catapulta a una zona determinada para derribar diferentes objetivos. En la zona de la catapulta el usuario tiene 4 objetos interactuables: una manivela para recargarla, otra manivela para rotarla, una palanca para accionarla y munición que se puede colocar en la lanzadera de la misma. Los objetivos a derribar son murallas, torres y casas pertenecientes a un castillo, los objetos casas dan 3 puntos si son golpeados por la munición pero hay murallas y torres que las protegen que proporcionan 1 y 2 puntos respectivamente y hay que derribarlas primero.
- Armas de fuego: Se imita el disparo de un mosquete con el que hay que acertar en diferentes objetivos. Al lado del player se encontrará una caja de munición la cual le permite coger munición pulsando un botón del controlador. Se coloca pólvora y munición para dispararla. Los objetos dianas, distantes del jugador, no podrán ser golpeados si no es disparando la munición. Cuando dichas dianas son golpeadas por la munición o pasa un tiempo límite desaparecen. Cuantas más dianas acierte el usuario más puntos conseguirá.
- Disparo con arco: Se simula el uso de un arco para alcanzar dianas o diferentes objetivos. Al lado del player se encontrará una caja de flechas que le permite cogerlas pulsando un botón del controlador. Se coge el arco con un controlador y con el otro se coloca la flecha. Los objetos dianas, distantes del jugador, solo serán golpeados lanzando las

flechas sin el arco. Cuando dichas dianas son golpeadas por las flechas o pasa un tiempo límite desaparecen. Cuantas más dianas acierte el usuario más puntos conseguirá.

- Escalada: Se simula con los mandos el agarre de distintos elementos pudiendo alternar ambas manos provocando que el jugador avance. Se simula la escalada por un castillo. El usuario tiene que escalar mientras coloca unas bombas, situadas en la cintura del mismo, en algunos puntos del muro. Cuando se coloca correctamente una bomba aumenta su puntuación.
- Bloqueo con escudo: En esta escena nos disparan flechas y tendremos que cubrirnos con el escudo para que no nos alcancen. Se empieza con una puntuación máxima y por cada impacto recibido se disminuirá la puntuación.
- Justa: El objetivo es simular una justa de caballeros imitando el movimiento de la lanza para alcanzar un objetivo mientras vamos montados a caballo. Cuando se alcanza aumenta la puntuación del jugador.

2.3. Especificación de casos de uso

El actor en este proyecto será el jugador. 2.1 2.1

Actor	Player	ACT-1
Descripción	Persona que prueba el proyecto con las gafas HTC y los mandos.	
Características	Cualquier persona que tenga más de 12 años.	
Relaciones	El player interacciona con los objetos creados en cada prueba por el GameController y accede a las diferentes pruebas almacenando su puntuación por el GameManager .	
Referencias	CU-01 2.2, CU-02 2.3, CU-03 2.4, CU-05 2.6, CU-07 2.8, CU-08 2.9, CU-09 2.10	

Cuadro 2.1: Actor ACT-1.

2.3.1. Casos de Uso

En este apartado se mostrarán los diagramas de caso de uso y la descripción de cada uno.

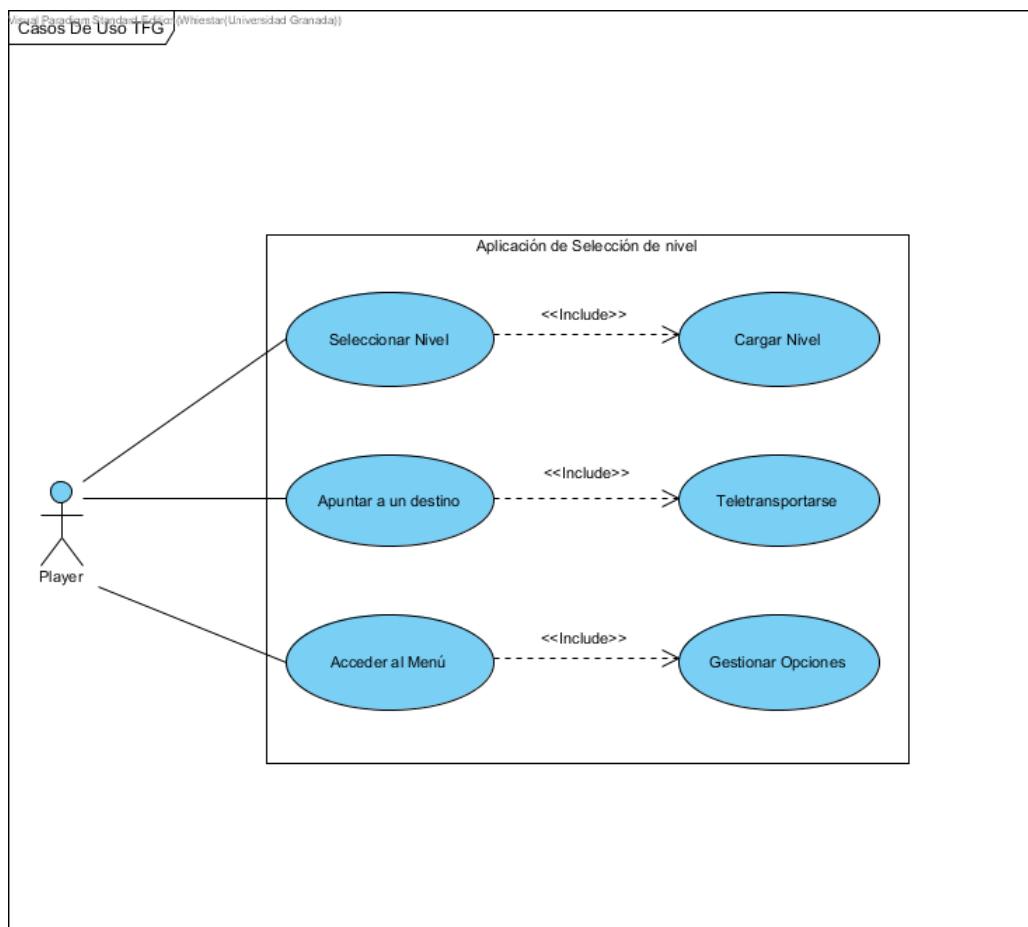


Figura 2.1: Diagrama de casos de uso de selección de nivel.

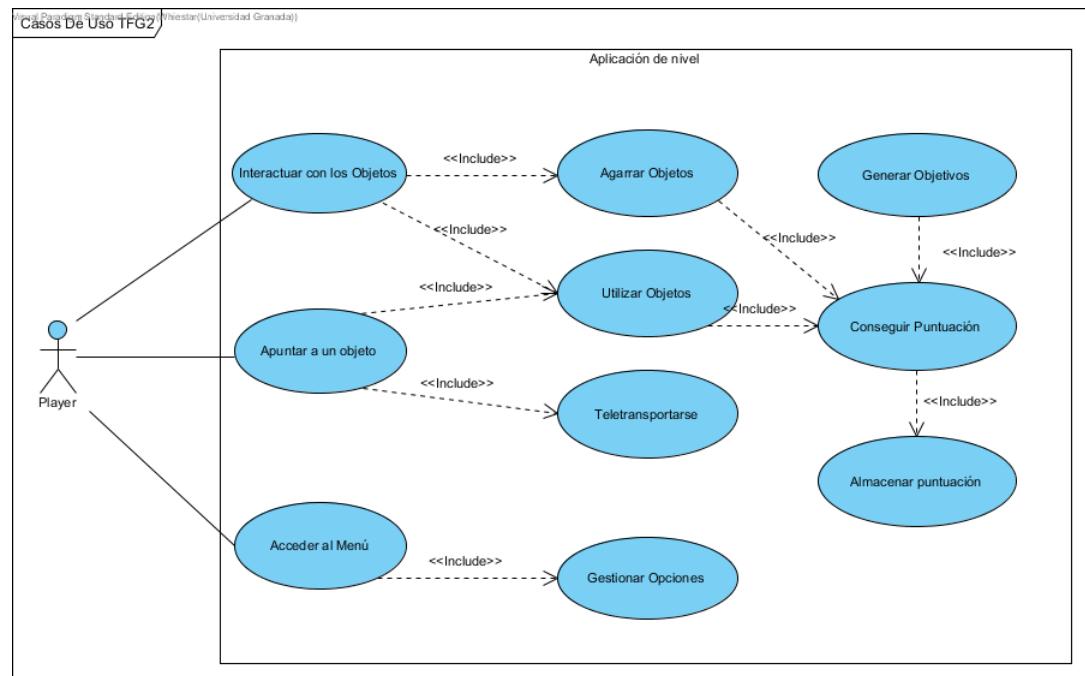


Figura 2.2: Diagrama de casos de uso de cada nivel.

Caso de Uso	Seleccionar Nivel	CU-01
Actor	Player	
Tipo	Principal — Real	
Referencias	RF-01 2.14, RF-02 2.15	CU-04 2.5
Precondiciones	El usuario debe estar en la escena de selección de nivel.	
Poscondiciones	Se informará al GameManager del nivel seleccionado.	
Propósito	Permitir al jugador acceder a los distintos niveles.	
Resumen	Las esferas de nivel están asociadas cada una a una prueba en concreto y el jugador debe coger una de esas esferas y acercarla a su hitbox en la cara.	

Cuadro 2.2: Caso de uso CU-01.

Caso de Uso	Apuntar a un destino u objeto	CU-02
Actor	Player	
Tipo	Principal — Real	
Referencias	RF-04 2.17, RF-05 2.18, RNF-01 2.23	CU-05 2.6, CU-09 2.10
Precondiciones	El puntero del player debe estar habilitado para esa escena, el player debe pulsar el botón Touchpad y el puntero debe colisionar con el destino u objeto.	
Postcondiciones	Se podrá interactuar con el objeto o destino.	
Propósito	Permitir al player interactuar con objetos lejanos.	
Resumen	Al utilizar el botón Touchpad se puede apreciar como un puntero, lineal o curva de Bezier, sale desde el extremo del controlador.	

Cuadro 2.3: Caso de uso CU-02.

Caso de Uso	Acceder al Menú	CU-03
Actor	Player	
Tipo	Principal — Real	
Referencias	RF-03 2.16,	CU-06 2.7
Precondiciones	El player debe pulsar el botón menú.	
Postcondiciones	Se podrán seleccionar las distintas opciones del menú.	
Propósito	Generar un mecanismo para que el player pueda volver al escenario de selección de nivel.	
Resumen	Al pulsar el botón menú el player puede seleccionar las distintas opciones del menú usando el otro controlador.	

Cuadro 2.4: Caso de uso CU-03.

Caso de Uso	Cargar Nivel	CU-04
Actor	Player	
Tipo	Principal — Real	
Referencias	RF-02 2.15	CU-01 2.2
Precondiciones	El player debe haber seleccionado un nivel.	
Postcondiciones	Se cargará el nivel escogido por el player.	
Propósito	Crear el mecanismo de transición de escenas.	
Resumen	Cuando el player seleccione una escena el GameManager sera el encargado de cambiarla.	

Cuadro 2.5: Caso de uso CU-04.

Caso de Uso	Teletransportarse	CU-05
Actor	Player	
Tipo	Principal — Real	
Referencias	RF-04 2.17, RF-05 2.18	CU-02 2.3,
Precondiciones	Se tiene que haber apuntado a un destino y el destino debe estar habilitado.	
Postcondiciones	El player se teletransportará al destino.	
Propósito	Crear un mecanismo para mover la PlayArea del player.	
Resumen	El player se moverá a la posición marcada por el puntero ésta está habilitada.	

Cuadro 2.6: Caso de uso CU-05.

Caso de Uso	Gestionar Opciones	CU-06
Actor	Player	
Tipo	Principal — Esencial	
Referencias	RF-03 2.16	CU-03 2.4
Precondiciones	El player debe haber accedido al menú y haber seleccionado una opción.	
Postcondiciones	Se procesará la opción elegida por el player.	
Propósito	Crear el procesamiento del menú del player.	
Resumen	Cuando el player elija una de las opciones del menú el GameManager cambiara la escena según la elección realizada.	

Cuadro 2.7: Caso de uso CU-06.

Caso de Uso	Interactuar con los Objetos	CU-07
Actor	Player	
Tipo	Principal — Real	
Referencias	RF-05 2.18, RF-06 2.19, RNF-01 2.23, RNF-02 2.24, RNF-03 2.25, RNF-04 2.26	CU-08 2.9, CU-09 2.10, CU-11 2.12, CU-12 2.13
Precondiciones	El player debe tocar el objeto, pulsar el botón Grip y el objeto debe ser interactuable.	
Postcondiciones	Dependiendo de la función del objeto se agarrará o se utilizará.	
Propósito	Permitir al jugador interactuar con el medio.	
Resumen	Se permite al jugador interactuar con los objetos de la escena tocándolos con los controladores y usando el botón Grip.	

Cuadro 2.8: Caso de uso CU-07.

Caso de Uso	Agarrar Objetos	CU-08
Actor	Player	
Tipo	Secundario — Real	
Referencias	RF-05 2.18, RF-06 2.19, RNF-01 2.23, RNF-02 2.24, RNF-03 2.25	CU-07 2.8, CU-11 2.12, CU-12 2.13
Precondiciones	Los objetos deben estar en contacto con el controlador, deben ser interactuables y se debe pulsar el botón Grip.	
Postcondiciones	El objeto heredará los movimientos del controlador.	
Propósito	Permitir al player sostener objetos para cumplir las pruebas.	
Resumen	El player podrá agarrar objetos que sean interactuables y usarlos para completar las pruebas.	

Cuadro 2.9: Caso de uso CU-08.

Caso de Uso	Utilizar Objetos	CU-09
Actor	Player	
Tipo	Secundario — Real	
Referencias	RF-04 2.17, RF-05 2.18, RNF-01 2.23, RNF-03 2.25	CU-02 2.3, CU-07 2.8, CU-11 2.12, CU-12 2.13
Precondiciones	Los objetos deben estar en contacto con el controlador o agarrados, deben ser interactuables y se debe pulsar el botón Trigger.	
Postcondiciones	El objeto realizará la acción que tenga asociada.	
Propósito	Permitir accionar objetos con distintas funciones.	
Resumen	El player podrá hacer que los objetos, que sean interactuables y usables, realicen su acción para completar las pruebas.	

Cuadro 2.10: Caso de uso CU-09.

Caso de Uso	Generar Objetivos	CU-10
Actor	Player	
Tipo	Principal — Real	
Referencias	RF-08 2.21, RF-09 2.22, RNF-03 2.25, RNF-04 2.26, RNF-05 2.27	
Precondiciones	Se debe haber seleccionado una de las escenas de pruebas.	
Postcondiciones	El GameController de dicha prueba generará los objetivos y controlará los objetos de la prueba.	
Propósito	Tener un controlador que instancie los objetos de la prueba y comprobar los aciertos del jugador en dicha prueba.	
Resumen	Cuando se inicia una prueba el GameController instancia los objetivos y controla sus acciones.	

Cuadro 2.11: Caso de uso CU-10.

Caso de Uso	Conseguir Puntuación	CU-11
Actor	Player	
Tipo	Principal — Esencial	
Referencias	RF-07 2.20	CU-02 2.3, CU-07 2.8, CU-08 2.9, CU-09 2.10, CU-12 2.13
Precondiciones	El player debe haber conseguido una puntuación al realizar alguno de los objetivos de la prueba.	
Postcondiciones	Se aumentará la puntuación conseguida por el player en ese intento.	
Propósito	Dar al player una puntuación por sus acciones realizadas durante la prueba.	
Resumen	Cuando el player cumple un objetivo se aumenta su puntuación en dicha prueba para ese intento.	

Cuadro 2.12: Caso de uso CU-11.

Caso de Uso	Almacenar Puntuación	CU-12
Actor	Player	
Tipo	Principal — Real	
Referencias	RF-07 2.20	CU-02 2.3, CU-07 2.8, CU-08 2.9, CU-09 2.10, CU-11 2.12,
Precondiciones	Debe haber finalizado una prueba.	
Postcondiciones	Se guardará la puntuación final conseguida por el player en esa prueba.	
Propósito	Mantener una puntuación de las pruebas de manera que si se cambia de escena no se pierda la puntuación.	
Resumen	El GameController guarda la puntuación obtenida para un intento de una prueba, si esa puntuación es mayor que la mejor conseguida hasta ese punto, guardada por el GameManager, sobrescribiéndose.	

Cuadro 2.13: Caso de uso CU-12.

2.4. Especificación de Requisitos

Los requisitos pueden clasificarse como funcionales o no funcionales. Mientras que los requisitos funcionales describen el funcionamiento específico que debe tener el sistema, los requisitos no funcionales especifican características del funcionamiento.

2.4.1. Requisitos Funcionales

Requisito Funcional	RF-01	Fuente	CU-01		
Nombre	Selección de nivel				
Descripción	Se puede seleccionar el nivel a partir de unos objetos interactivos por el player.				
Prioridad	Alta	Necesidad	Esencial	Estabilidad	Estable

Cuadro 2.14: Requisito funcional RF-01.

Requisito Funcional	RF-02	Fuente	CU-04		
Nombre	Cambio de Escenas				
Descripción	Un objeto tiene que encargarse de realizar los cambios de escenas cuando sea necesario.				
Prioridad	Alta	Necesidad	Esencial	Estabilidad	Estable

Cuadro 2.15: Requisito funcional RF-02.

Requisito Funcional	RF-03	Fuente	CU-03		
Nombre	Menú de gestión de juego				
Descripción	El menú debe integrar opciones que permitan al player volver a la escena de selección de nivel y reiniciar la prueba.				
Prioridad	Alta	Necesidad	Esencial	Estabilidad	Estable

Cuadro 2.16: Requisito funcional RF-02.

Requisito Funcional	RF-04	Fuente	CU-02		
Nombre	Configuración del puntero				
Descripción	El puntero debe tener: unos parámetros (Longitud máxima y grosor entre otros), una forma (lineal o curva de bezier) y configuración de colisión (con destino y objetos interactivos).				
Prioridad	Media	Necesidad	Esencial	Estabilidad	Estable

Cuadro 2.17: Requisito funcional RF-04.

Requisito Funcional	RF-05	Fuente	CU-07		
Nombre	Interacción con Objetos				
Descripción	El player debe de tener la capacidad de interactuar con objetos colindantes a la hora de superar las pruebas.				
Prioridad	Alta	Necesidad	Esencial	Estabilidad	Estable

Cuadro 2.18: Requisito funcional RF-05.

Requisito Funcional	RF-06	Fuente	CU-07		
Nombre	Detección de colisión entre objetos interactivos				
Descripción	En las pruebas que requieran acertar con un objeto interactuable en otro objeto se deben comprobar las colisiones entre estos mismos.				
Prioridad	Alta	Necesidad	Esencial	Estabilidad	Inestable

Cuadro 2.19: Requisito funcional RF-06.

Requisito Funcional	RF-07	Fuente	CU-12		
Nombre	Base de datos de puntuación				
Descripción	La puntuación obtenida por el jugador debe ser guardada y no se debe perder en el cambio de escenas.				
Prioridad	Alta	Necesidad	Esencial	Estabilidad	Estable

Cuadro 2.20: Requisito funcional RF-07.

Requisito Funcional	RF-08	Fuente	CU-10		
Nombre	Generación de Objetivos				
Descripción	Un objeto debe encargarse de generar en una prueba los objetivos y reglas definidas para completar dicha prueba.				
Prioridad	Alta	Necesidad	Esencial	Estabilidad	Estable

Cuadro 2.21: Requisito funcional RF-08.

Requisito Funcional	RF-09	Fuente	CU-10		
Nombre	Comportamiento de Objetivos				
Descripción	En las pruebas que los objetivos requieran de un comportamiento, se debe definir para cada objetivo el comportamiento que debe seguir.				
Prioridad	Media	Necesidad	Esencial	Estabilidad	Inestable

Cuadro 2.22: Requisito funcional RF-09.

2.4.2. Requisitos No Funcionales

Requisito No Funcional	RNF-01	Fuente	CU-07		
Nombre	Capacidad de interacción del controlador				
Descripción	El player solo podrá interactuar con un objeto a la vez por cada controlador.				
Prioridad	Alta	Necesidad	Esencial	Estabilidad	Estable

Cuadro 2.23: Requisito no funcional RNF-01.

Requisito No Funcional	RNF-02	Fuente	CU-07		
Nombre	Detección precisa de colisiones				
Descripción	La detección entre las colisiones debe de ser los más precisa posible.				
Prioridad	Media	Necesidad	Deseable	Estabilidad	Inestable

Cuadro 2.24: Requisito no funcional RNF-02.

Requisito No Funcional	RNF-03	Fuente	CU-07		
Nombre	Inmersividad del juego				
Descripción	La interacción con los objetos y los gráficos de los mismos deben ser lo mas reales posibles para conseguir una inmersividad del player.				
Prioridad	Media	Necesidad	Deseable	Estabilidad	Estable

Cuadro 2.25: Requisito no funcional RNF-03.

Requisito No Funcional	RNF-04	Fuente	CU-07		
Nombre	Diseño dinámico del mapa de juego				
Descripción	El diseño del mapa de juego debe ser consecuente a la visión del player sin excederse en objetos para intentar ahorrar la mayor cantidad de memoria posible.				
Prioridad	Baja	Necesidad	Deseable	Estabilidad	Estable

Cuadro 2.26: Requisito no funcional RNF-04.

Requisito No Funcional	RNF-05	Fuente	CU-10					
Nombre	Generación aleatoria de objetivos							
Descripción	Los objetivos para una prueba deben generarse de manera procedural para generar una dificultad en las pruebas.							
Prioridad	Baja	Necesidad	Deseable	Estabilidad	Estable			

Cuadro 2.27: Requisito no funcional RNF-05.

Capítulo 3

Planificación

En este capítulo se expondrá la planificación seguida a lo largo del proyecto y la metodología usada. En este proceso se ha dividido el trabajo a realizar en tareas, de mayor o menor complejidad, y se mostrará el tiempo que han llevado realizarse cada una de ellas.

3.1. Metodología Ágil

La metodología utilizada en este proyecto ha sido la metodología ágil [19] Scrum [20], la cual implementa interacciones cortas llamadas sprints, que son fáciles de estimar en tiempo, en las que se introducen tareas a todos los niveles, desde concepto y diseño hasta prueba y optimización. Cada varias iteraciones generan una versión del juego en la que se alcanzan objetivos globales. Esta metodología permite adaptar el proyecto a medida que se realiza facilitando la implementación para alcanzar un mejor resultado.

Los sprint definidos para este proyecto no deben superar las dos semanas. Aproximadamente se estiman unos 15 sprints de 20/30 horas de desarrollo cada uno.

Al aplicar la metodología Scrum el proyecto sera funcional desde las primeras etapas de la implementación permitiendo corregir errores y modificar el proyecto desde una etapa temprana.

Las tareas a realizar en este proyecto serían:

- Aprendizaje sobre la librería VRTK.
- Análisis de Requisitos y elección de motor gráfico.
- Diseño, implementación y pruebas del videojuego.

- Realización de la memoria.

Aprendizaje sobre la librería VRTK

Sprint	Inicio	Fin	Descripción	Estimado	Realizado
VRTK_0	11/10/17	25/10/17	Se documentaron los ejemplos 1-5.	14 días	14 días
VRTK_1	26/10/17	8/11/17	Se documentaron los ejemplos 6-10.	14 días	14 días
VRTK_2	10/11/17	29/11/17	Se documentaron los ejemplos 11-15.	14 días	19 días
VRTK_3	30/11/17	13/12/17	Se documentaron los ejemplos 16-20.	14 días	15 días
VRTK_4	14/12/17	29/12/17	Se documentaron los ejemplos 21-25.	15 días	15 días
VRTK_5	3/1/18	17/1/18	Se documentaron los ejemplos 26-30.	14 días	14 días
VRTK_6	18/1/18	1/2/18	Se documentaron los ejemplos 31-35.	15 días	14 días
VRTK_7	2/2/18	19/2/18	Se documentaron los ejemplos 36-40.	15 días	17 días
VRTK_8	20/2/18	6/3/18	Se documentaron los 4 últimos ejemplos.	16 días	14 días

Cuadro 3.1: Aprendizaje sobre la librería VRTK.

Análisis de Requisitos y elección de motor gráfico

Sprint	Inicio	Fin	Descripción	Estimado	Realizado
AR_0	12/3/18	26/3/18	Se estableció la especificación de requisitos.	14 días	14 días
AR_1	27/3/18	8/4/18	Se estableció Unity como motor gráfico.	14 días	12 días

Cuadro 3.2: Análisis de Requisitos y elección de motor gráfico.

Diseño, implementación y pruebas del videojuego

Sprint	Inicio	Fin	Descripción	Estimado	Realizado
VG_0	9/4/18	23/4/18	Diseño e implementación de la prueba de lanzamiento de objetos. Fuera del tiempo de diseño e implementación se ha sometido a algunos cambios y actualizaciones.	14 días	14 días
VG_1	26/4/18	10/5/18	Diseño e implementación de la prueba de catapulta. Fuera del tiempo de diseño e implementación se ha sometido a algunos cambios y actualizaciones.	16 días	15 días
VG_2	14/5/18	21/5/18	Diseño e implementación de la prueba de arma de fuego y el selector de niveles. Fuera del tiempo de diseño e implementación se ha sometido a algunos cambios y actualizaciones.	7 días	7 días
VG_3	20/6/18	27/6/18	Diseño e implementación de la prueba de tiro con arco. Fuera del tiempo de diseño e implementación se ha sometido a algunos cambios y actualizaciones.	10 días	8 días
VG_4	2/7/18	10/7/18	Diseño e implementación de la prueba de justa de caballeros. Fuera del tiempo de diseño e implementación se ha sometido a algunos cambios y actualizaciones.	7 días	9 días
VG_5	11/7/18	20/7/18	Diseño e implementación de la prueba de escalada. Fuera del tiempo de diseño e implementación se ha sometido a algunos cambios y actualizaciones.	14 días	9 días
VG_6	20/7/18	2/8/18	Diseño e implementación de la prueba de escudo. Fuera del tiempo de diseño e implementación se ha sometido a algunos cambios y actualizaciones.	14 días	13 días
BUG	27/8/18	6/9/18	Prueba del juego y búsqueda de bugs.	10 días	11 días

Cuadro 3.3: Diseño, implementación y pruebas del videojuego.

Realización de la memoria

Sprint	Inicio	Fin	Descripción	Estimado	Realizado
MEM_0	22/5/18	5/6/18	Recopilación de biografía y realización de los dos primeros capítulos de la memoria.	16 días	14 días
MEM_1	6/8/18	26/8/18	Realización de los 6 capítulos siguientes.	15 días	21 días

Cuadro 3.4: Realización de la memoria.

3.2. Diagrama de Gantt

Para la planificación de tiempo dedicado a cada sprint se ha realizado un diagrama de Gantt 3.1 utilizando la aplicación GanttProject [21]. El diagrama de Gantt muestra los Sprints desarrollados junto al tiempo invertido en cada una de ellos.

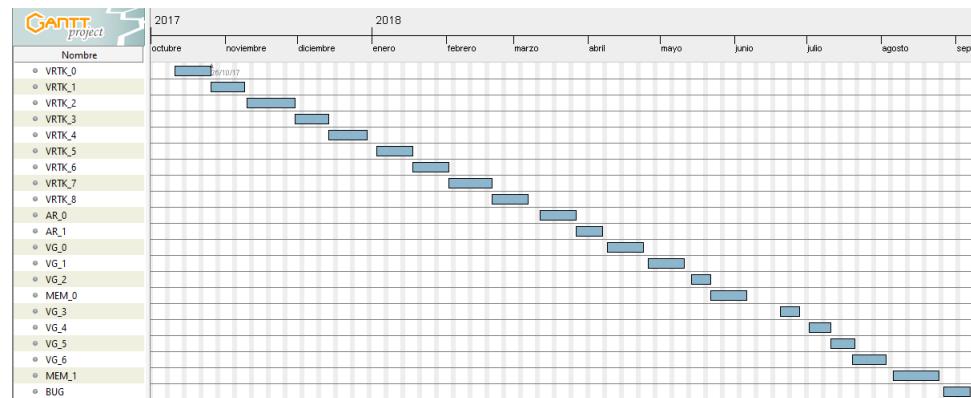


Figura 3.1: Diagrama de Gantt.

3.3. Herramientas utilizadas**3.3.1. Github**

Github [22] es web que implementa un sistema de control de versiones Git permitiendo a los desarrolladores subir sus proyectos manteniendo todas las versiones y modificaciones aplicadas al mismo. Esta misma aplicación incorpora también un sistema de 'issues' como un sistema de tareas a realizar en el proyecto.

Para este proyecto se ha creado un repositorio en Github 3.2 para realizar un control de versiones dado que en este tipo de proyectos es asiduo cometer errores de programación y se podría necesitar recuperar una versión anterior.

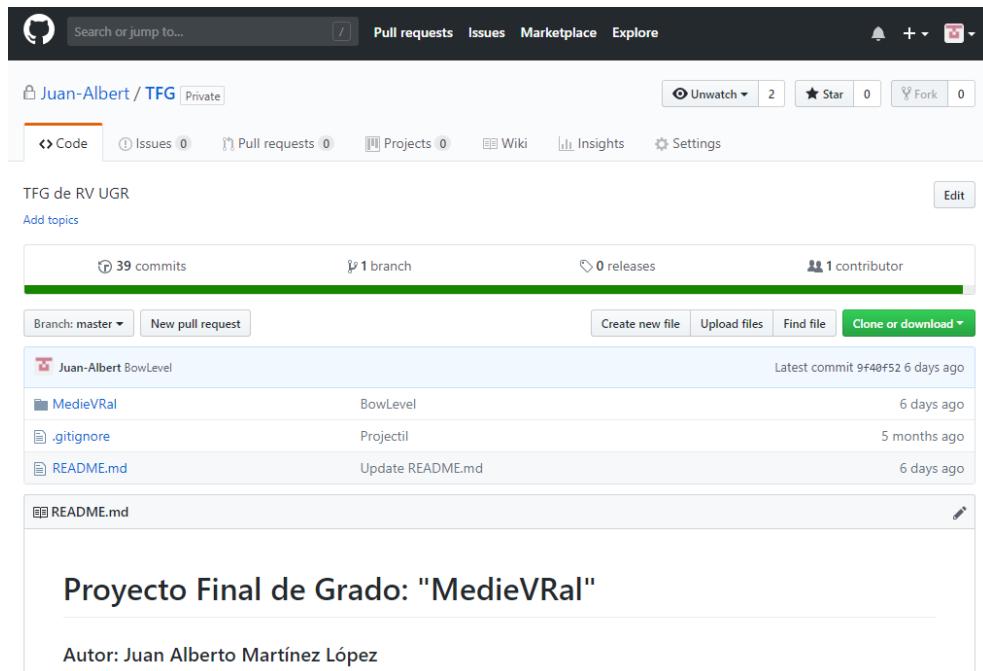


Figura 3.2: Utilización de la aplicación Github.

3.3.2. Trello

Trello [23] es una herramienta utilizada en el ámbito empresarial para la planificación de proyectos. La aplicación usa tableros en los cuales se incluye listas de tareas, permitiendo mover unas tareas de unos tableros a otros. Estas tareas pueden etiquetarse como realizadas, asignarse a miembros en concreto y se les puede asignar una fecha de vencimiento.

Para este proyecto 3.3 se ha creado un tablero 'To do list' para tareas genéricas de un sprint, de manera que para cada sprint hay un tablero para cada escena del proyecto, así que cada escenario tiene sus tareas permitiendo así que para cada sprint se puedan utilizar todos los tableros para hacer modificaciones; y un tablero 'Done' para las tareas finalizadas.

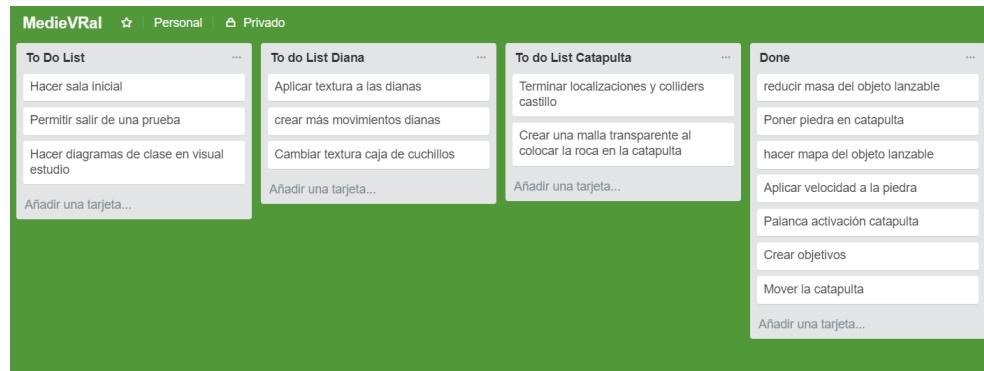


Figura 3.3: Utilización de la aplicación Trello en el sprint VG_1.

3.4. Elección del Motor Gráfico

Uno de los componentes más importantes en el desarrollo de un videojuego, es el motor gráfico. El motor gráfico provee un conjunto de componentes software para solucionar problemas tales como el renderizado de imágenes, compilación y procesado de los scripts, generador de físicas, solucionador de colisiones y creación de animaciones. En definitiva un motor gráfico permite al desarrollador abstraerse de la programación de dichos problemas con el uso de las distintas **API** que el motor proporciona.

A la hora de elegir el motor se ha utilizado una comparación que he realizado en la asignatura de *Programación Lúdica* entre distintos motores cuyas licencias estándar sean gratuitas, eficientes y compatibles con el software de HTC Vive y el sistema operativo Windows. Bajo esas premisas los motores gráficos escogidos son los siguientes:

- Unreal Engine[24] es un motor gráfico que usa C++ como lenguaje de scripting C++ y BluePrints. No es necesario programar en C++ dado que los blueprint se componen de diagramas de flujo que utilizan módulos, los cuales permiten realizar acciones de transformación y diseño básico de objetos. Posee una pequeña cantidad de documentación, que es poco extensa y dificulta el aprendizaje.
- Unity[25] es un motor gráfico gratuito para el uso no comercial que utiliza C# como lenguaje de scripting. Permite un alto grado de abstracción debido a la gran cantidad de **APIs** y Assets que incluye. Respecto al uso de memoria Unity genera un puntero para cada GameObject permitiendo mayor eficiencia y menor consumo de memoria.

cada vez que utilizamos dicho objeto, ya que evita manejarlo íntegramente, siendo más pesado debido a los componentes y materiales asociados al mismo. Unity posee una amplia cantidad de documentación, tutoriales y feedback activa, lo que permite hacer un proceso de Debug menos intenso.

Ambos motores son muy parecidos. Dado que el paquete VRTK y SteamVR tienen problemas de compatibilidad con Unreal Engine, he decidido desarrollar el proyecto con Unity que tiene mayor documentación y facilita el aprendizaje del mismo.

3.5. Restricciones del sistema

3.5.1. Restricciones Hardware

- Los controladores tienen una interacción limitada por el número de botones. A la hora de asignar acciones a los controladores hay que tener en cuenta esta limitación.
- El dispositivo necesita de un espacio mínimo para el Play Area por lo que se tiene que instalar los dispositivos en un entorno amplio.
- Las gafas van conectadas al ordenador por cable pues necesitan corriente.
- Los controladores tienen batería por lo que hay que cargarlos cada cierto tiempo.
- Es importante no trabajar con las gafas en un entorno con mucha iluminación ya que se interrumpe con mayor facilidad la detección de las base stations.

3.5.2. Restricciones Software

- Es necesario realizar el proyecto en el software Unity porque VRTK no es compatible con la mayoría de los motores gráficos.
- Es necesario desarrollar el proyecto en Windows dado que Unity no tiene soporte para Linux.
- Se trabajará en el lenguaje de programación C# ya que es el utilizado en Unity.

Capítulo 4

Análisis

En este capítulo se definen que factores inmersivos que afectan al jugador. Para el desarrollo del videojuego me he basado en varias mecánicas utilizadas en el juego 'The Lab' [26] de Steam que proporcionan inmersión en las pruebas que se realizan.

4.1. Factores Inmersivos

En el proceso de juego con VR existen diversos elementos relacionados con la inmersividad. Un equipo de realidad virtual inmersivo puede incluso tener efectos beneficiosos a la hora de producir factores analgésicos [27], ya que los equipos menos inmersivos pueden inducir a problemas como náuseas o mareos. Estos elementos o factores pueden modificar la experiencia de juego y lo real que siente el jugador que es, ya sea debido a factores directos con los que el jugador se relaciona en el entorno virtual o factores indirectos.

4.1.1. Factores inmersivos directos

Los factores que se han encontrado en relación directa con la inmersión son las físicas, la interacción con los objetos, la posición de la cámara, movilidad y los dispositivos utilizados.

Físicas

Cuando nos sumergimos en un mundo donde las físicas se corresponden con la idea que tenemos de ese entorno, la reacción suele ser atractiva para el usuario hacia ese mundo. Ya que una física de gravedad en el espacio, donde

la fuerza que atrae al usuario a un asteroide es menor que la gravedad de la tierra, es más inmersivo.

Interacción con Objetos

A la hora de interactuar con objetos una de las características mas inmersivas es el feedback. Cuando nosotros tensamos un arco y el controlador vibra, como vibraría la cuerda del arco al tensarse, crea una falsa sensación de realidad.

Posición de la cámara

La sensación de altura es una de los factores más determinantes en la inmersión. Si la posición de la cámara no se ajusta bien al entorno en el que nos encontramos el usuario detecta la ficción dentro del juego. Esto ocurre cuando se calcula mal la cámara y el campo visual no coincide con el campo de acción.

Movilidad

Al permitir al usuario moverse con libertad en una Play Area, que sus movimientos se vean reflejados en el mundo virtual y que mediante algún mecanismo el usuario pueda mover esa Play Area alrededor del entorno proporciona al usuario una sensación de libertad.

Calidad de los dispositivos de RV

La calidad del casco de Realidad Virtual en cuanto a características técnicas, como la resolución de las lentes o el peso, influye a la hora de inmergirse en el juego.

4.1.2. Factores inmersivos indirectos

Gráficos

Como en cualquier juego unos efectos gráficos bien definidos, tanto realistas como animados, permiten al jugador abstraerse del mundo real.

Profundidad

Cuando en un escenario se muestran fondos, creando la sensación de profundidad, el jugador siente que el mundo virtual no es finito.

Sonido

El sonido generado por el ambiente, como los pájaros o el viento, añadido al sonido generado por las acciones del jugador en el juego generan un feedback hacia el usuario.

Capítulo 5

Diseño

5.1. Diseño del juego: MediVRal

5.1.1. Resumen

MedieVRal, es un juego en el cual nos ponemos en la piel de un caballero en la edad medieval, el cual tiene que superar una serie de pruebas para conseguir ser el mejor caballero del reino. Estas representan competiciones reales de dicha era, como tiro con arco o una justa de caballeros.

5.1.2. Jugabilidad

El objetivo del juego consiste en obtener una puntuación lo suficientemente alta en una prueba como para poder acceder a la siguiente y al final haber superado todas las pruebas. Para conseguirlo el jugador puede acceder a los siguientes escenarios:

- Selector de niveles: en esta área el jugador tiene que teletransportarse, mediante los puntos de destino disponibles, y seleccionar un nivel a través de las bolas de cristal .
- Menú: El menú, que estará disponible para el jugador en cualquier prueba, permite volver al selector de niveles, reiniciar la prueba actual o consultar la puntuación.
- Pruebas: en cada prueba el jugador debe de demostrar su habilidad y superarla con los elementos disponibles. Dichos elementos son objetos con los que el jugador tendrá que interactuar para conseguir puntos.

5.1.3. Desarrollo del juego

Al comenzar el juego MedieVRal, aparecemos en el selector de niveles, el cual tiene apariencia de campamento, donde el jugador tiene que teletransportarse hacia uno de los destinos disponibles, agarrar una de las bolas de cristal, que representan los distintos niveles y acercarla a las gafas.

Una vez hecho esto comienza la prueba correspondiente a la bola de cristal elegida. En cualquier momento durante el juego el jugador puede abrir el menú mediante el botón "menú" del controlador y volver al selector de niveles.

Una vez finalizada la prueba, que se denotará con el sonido de un cuerno de guerra, al volver al selector de niveles si el jugador ha conseguido la suficiente puntuación desbloqueará la siguiente prueba. Si el jugador ha superado todas las pruebas con la suficiente puntuación habrá ganado.

5.2. Diagramas de clases

En este apartado se mostrarán los diagramas de clases para cada escena del juego con sus elementos más importantes. Dado que se reutilizarán clases de escenas anteriores no se repetirá su definición. También se explicarán algunas de las clases utilizadas del paquete VRTK que tienen relación con las clases desarrolladas para el proyecto, el resto se explicarán en el siguiente capítulo.

5.2.1. Selector de niveles

En la figura 5.1 se muestra el diagrama de clases del Selector de niveles 'SelectLevel'. En dicho diagrama se muestran las siguientes clases:

- **GameManager**: Esta clase se encarga de gestionar toda la puntuación del jugador así como de desbloquear las distintas pruebas una vez el jugador ha superado las anteriores correspondientes. Es un *singleton* por lo que solo existe una instancia de esta clase y no se destruye en el cambio de escenas.
- **SoundManager**: Esta clase se encarga de gestionar toda la música y efectos de sonido del juego. Es un *singleton* por lo que solo existe una instancia de esta clase y no se destruye en el cambio de escenas.

- **Loader**: Es la encargada de instanciar los objetos correspondientes a las clases *GameManager* y *SoundManager*.
- **LoadLevel**: Es la encargada de detectar cuando el jugador acerca una bola de cristal de nivel a las gafas y de cargar el nivel correspondiente.
- **DestinationPoint**: Es la encargada de detectar cuando el jugador ha conseguido la suficiente puntuación para desbloquear el TeleportPoint a la siguiente prueba.
- **BallonBehaviour**: Controla cuando el jugador ha ganado y genera globos por el campamento.

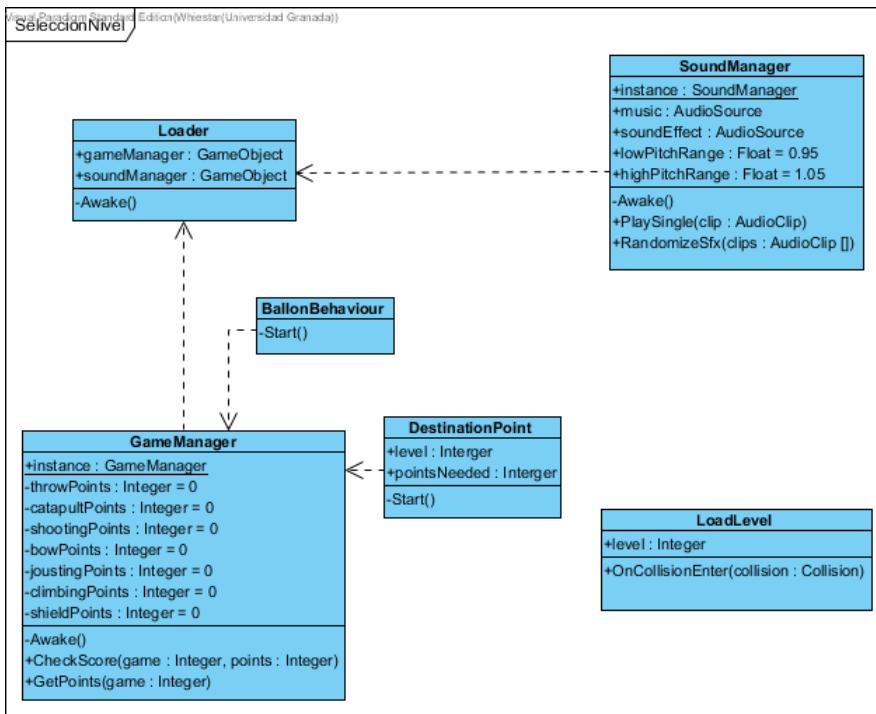


Figura 5.1: Diagrama de clases de la escena de Selección de nivel.

5.2.2. Menú de Usuario

En la figura 5.2 se muestra el diagrama de clases del menú al que el usuario puede acceder para conocer la puntuación, repetir el nivel o volver al selector de niveles. En dicho diagrama se muestran las siguientes clases:

- **MenuFunctions**: Esta clase se encarga de mostrar toda la puntuación del jugador en el menú del usuario así como de recargar el nivel actual o volver al selector de niveles.

- **MenuToggle:** Esta clase se encarga de gestionar el uso del botón *application menu* para abrir y cerrar el menú.
- **VRTK_ControllerEvents:** Es un script del paquete VRTK el cual controla cuando son usados los botones de los controladores.

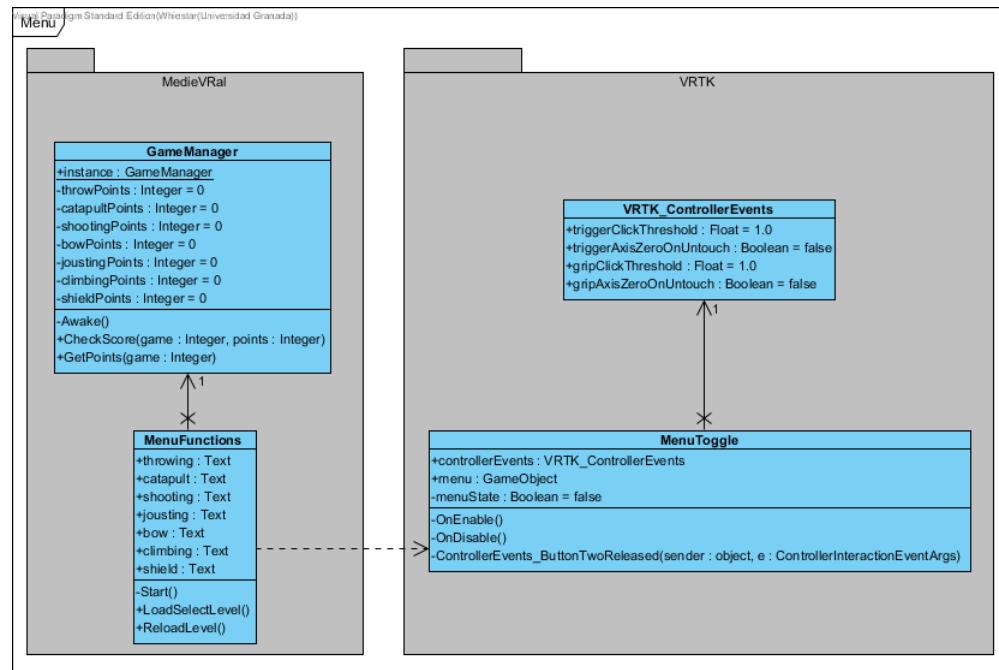


Figura 5.2: Diagrama de clases del menú de usuario.

5.2.3. Niveles

En la figura 5.3 se muestra el diagrama de clases de la primera prueba 'Throwing'. En dicho diagrama se muestran las siguientes clases:

- **ThrowManager:** Esta clase se encarga de instanciar los objetivos de la prueba 'Throwing', de manejar la puntuación antes de que finalice la prueba y de enviar la puntuación final a la clase *GameManager*.
- **Objectives:** Esta clase gestiona el comportamiento de los objetivos y de detectar cuando son alcanzados por el objeto lanzable y son destruidos. Utiliza la enumeración *Movement* para definir como se mueven los objetivos.
- **Throwable:** Gestiona la detección de colisión del objeto lanzable así como el quedarse clavado a los objetivos.

- **Spawner:** Clase que genera copias de un GameObject introducido por parámetro, cuando un controlador interactúa con el objeto que contiene la clase.
- **VRTK_InteractableObject:** Es un script del paquete VRTK el cual permite interaccionar solo con los objetos que queramos. Utiliza las enumeraciones *AllowedController* para definir con que controlador se puede interactuar con el objeto y *ValidDropTypes* para definir donde se puede soltarlo.
- **VRTK_InteractGrab:** Es un script del paquete VRTK el cual permite agarrar un objeto válido, es decir que contenga el script *VRTK_InteractableObject*, el cualhereda la rotación y posición del controlador mientras está agarrado y la velocidad y la dirección cuando se suelta el botón de agarre.
- **VRTK_InteractTouch:** Es un script del paquete VRTK en el que se aplica una malla de colisión para saber cuando el controlador está tocando algo, a los objetos que se les aplica este script también hay que asignarles el script *VRTK_InteractableObject* para que sean interactivables.

En la figura 5.4 se muestra el diagrama de clases de la segunda prueba 'Catapult'. En dicho diagrama se muestran las siguientes clases:

- **CatapultManager:** Esta clase se encarga de instanciar los objetivos de la prueba 'Catapult', de manejar la puntuación antes de que finalice y de enviar la puntuación final a la clase *GameManager*.
- **Lanzadera:** Esta clase gestiona el movimiento de la lanzadera de la catapulta así como de generar los GameObjects que contienen la clase *AmmoBehaviour* .
- **AmmoBehaviour:** Es la encargada de proporcionar al GameObject las propiedades físicas para que salga volando y pueda alcanzar los objetivos.
- **SlotRoca:** Es la clase encargada de detectar cuando se ha colocado munición en la lanzadera de la catapulta.
- **LeverControl:** Gestiona el funcionamiento de la palanca para activar la lanzadera de la catapulta.
- **WheelControlThrow:** Esta clase se encarga de recargar la lanzadera, conforme giramos la manivela, para poder volver a colocar munición.
- **WheelControlRotator:** Esta clase se encarga de girar la catapulta, conforme giramos la manivela, para apuntar a los objetivos.

- **VRTK_Control_UnityEvents:** Es un script del paquete VRTK el cual gestiona los eventos de Unity para detectar cuando cambia el valor de un objeto que le hemos asignado.

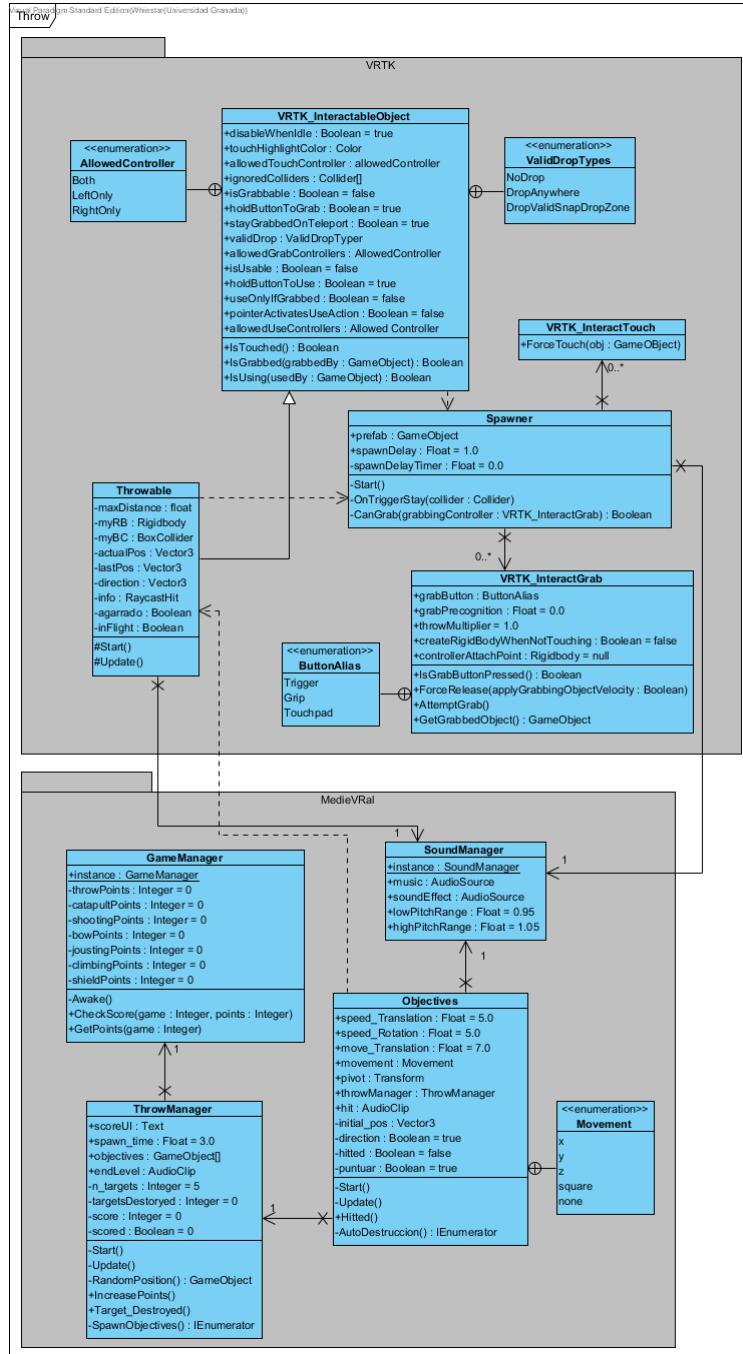


Figura 5.3: Diagrama de clases de la primera prueba (Trowing).

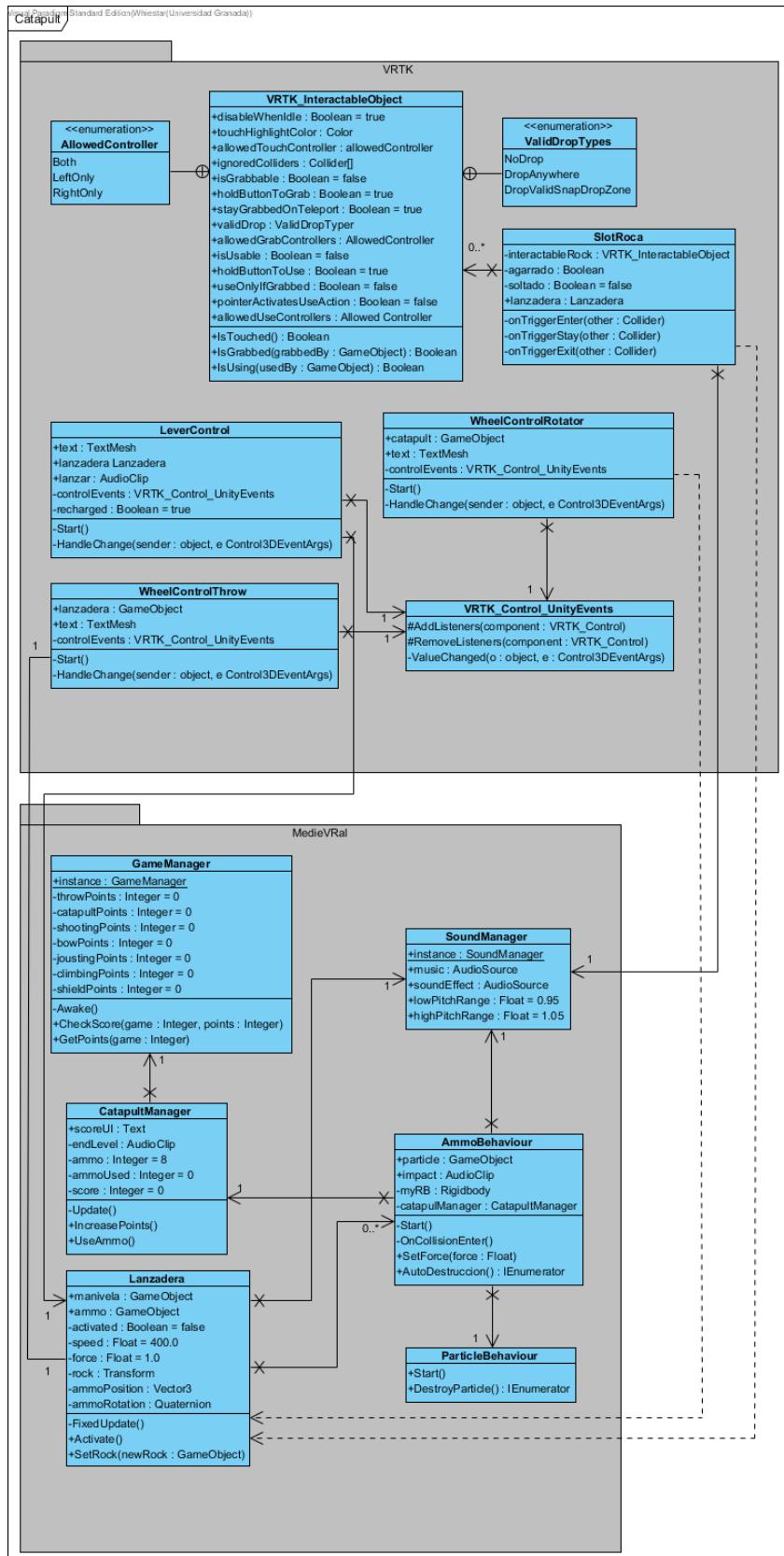


Figura 5.4: Diagrama de clases de la segunda prueba (Catapult).

En la figura 5.5 se muestra el diagrama de clases de la tercera prueba 'Shooting'. En dicho diagrama se muestran las siguientes clases:

- **ShootingManager**: Esta clase se encarga de instanciar los objetivos de la prueba 'Shooting', de manejar la puntuación antes de que finalice la prueba y de enviar la puntuación final a la clase *GameManager*.
- **Targets**: Esta clase gestiona el comportamiento de los objetivos y de detectar cuando son alcanzados por las balas y son destruidos. Utiliza la enumeración *Movement* para definir como se mueven los objetivos.
- **FlintLockPistol**: Esta clase gestiona la utilización de la pistola así como de generar los GameObjects que contienen la clase *BulletBehaviour*.
- **BulletBehaviour**: Es la encargada de proporcionar al GameObject las propiedades físicas para que salga volando y pueda alcanzar los objetivos.
- **SlotAmmo**: Es la clase encargada de detectar cuando se ha colocado pólvora y munición en el cañón de la pistola.

En la figura 5.6 se muestra el diagrama de clases de la cuarta prueba 'Jousting'. En dicho diagrama se muestran las siguientes clases:

- **JoustingManager**: Esta clase se encarga de instanciar los objetivos de la prueba 'Jousting', de manejar la puntuación antes de que finalice la prueba y de enviar la puntuación final a la clase *GameManager*.
- **Rings**: Esta clase gestiona el comportamiento de los objetivos y de detectar cuando son alcanzados por la lanza y son destruidos. Utiliza la enumeración *Movement* para definir como se mueven los objetivos.
- **Spear**: Gestiona la funcionalidad de la lanza, como objeto no soltable y con fuerza de golpe, y detecta cuando colisiona con los objetivos *Rings*.

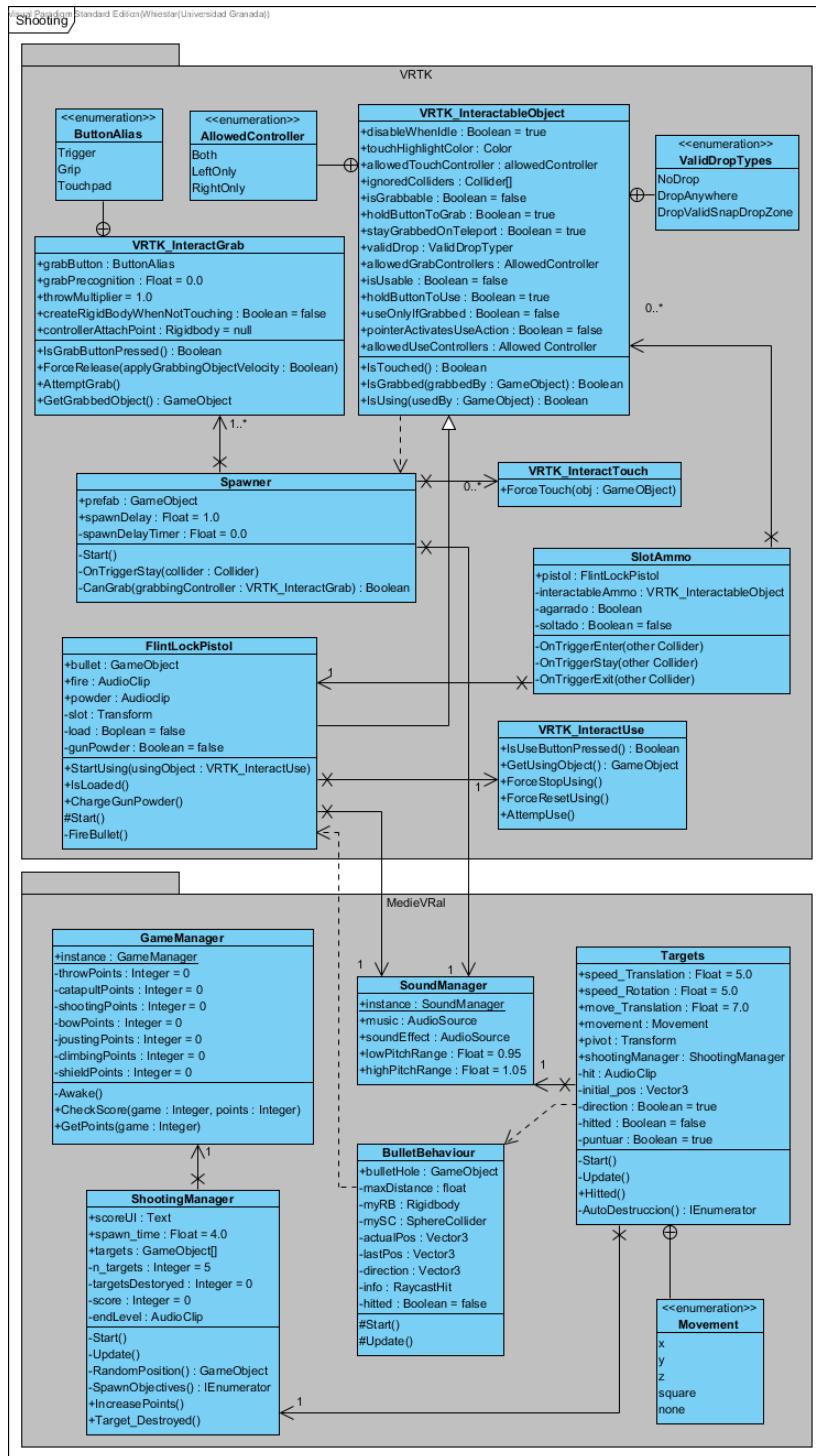


Figura 5.5: Diagrama de clases de la tercera prueba (Shooting).

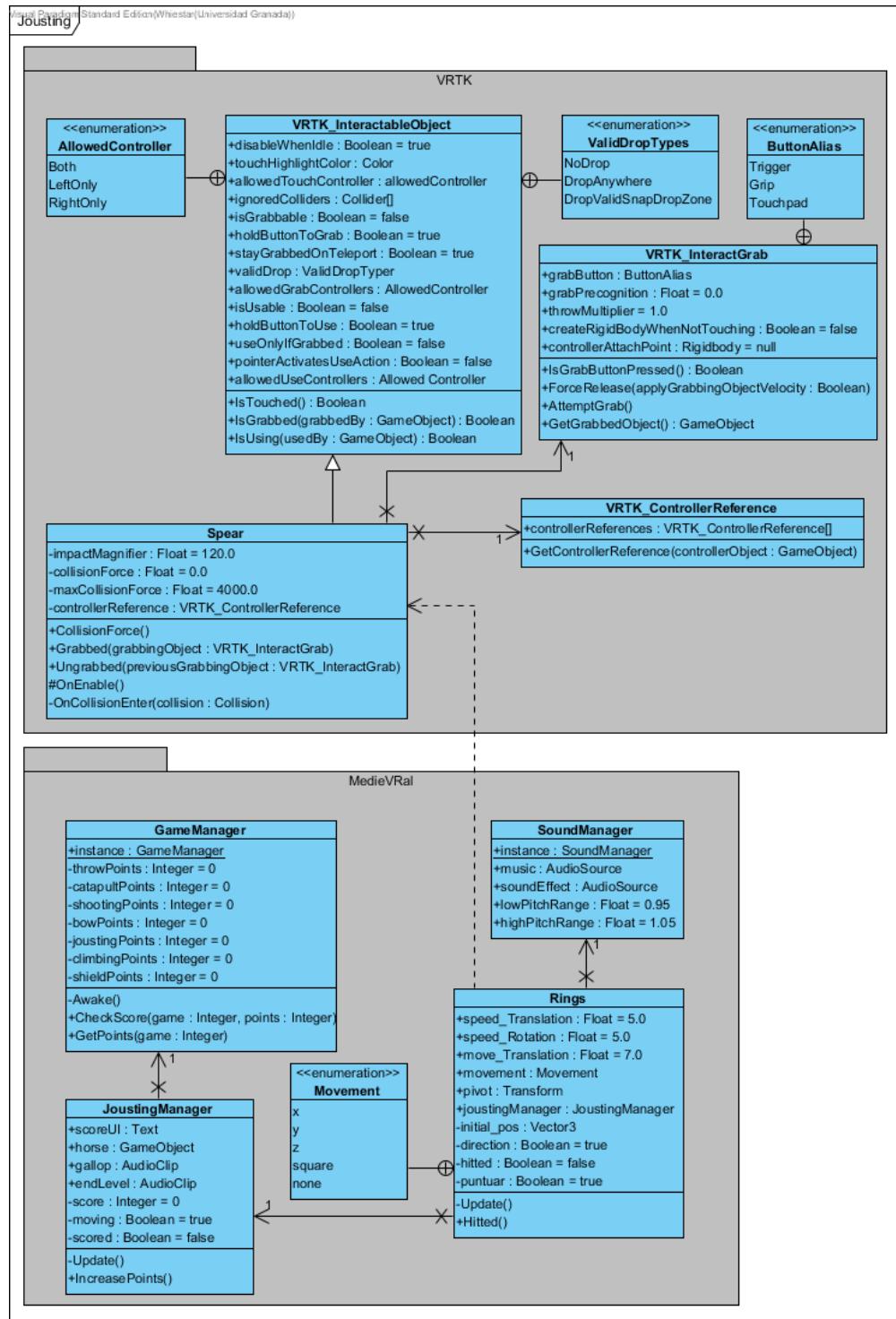


Figura 5.6: Diagrama de clases de la cuarta prueba (jousting).

En la figura 5.7 se muestra el diagrama de clases de la quinta prueba 'Bow'. En dicho diagrama se muestran las siguientes clases:

- **BowManager:** Esta clase se encarga de instanciar los objetivos de la prueba 'Bow', de manejar la puntuación antes de que finalice la prueba y de enviar la puntuación final a la clase *GameManager*.
- **BowTargets:** Esta clase gestiona el comportamiento de los objetivos y detecta cuando son alcanzados por los objetos *Arrow* y son destruidos. Utiliza la enumeración *Movement* para definir como se mueven los objetivos.
- **Arrow:** Esta clase es la encargada de gestionar el comportamiento del proyectil una vez es lanzado por el arco, de detectar las colisiones con los objetivos y de destruir el objeto.
- **ArrowGrip:** Esta clase es la encargada de gestionar el comportamiento del proyectil cuando es agarrado y cargado en el arco.
- **BowAim:** Se encarga de definir el comportamiento del arco cuando es agarrado: se coloca una flecha en la cuerda (detectando la posición de agarre del controlador y de agarre de la flecha en la cuerda con la clase *BowHandle*), se tensa y se dispara proporcionándole al GameObject del proyectil las propiedades físicas para que salga volando y pueda alcanzar los objetivos. También le pasa por parámetro a la clase *BowAnimation* el frame en el cual la animación se corresponde con el estado actual del arco.
- **Follow:** Se encarga de modificar la rotación y posición de la flecha para que encaje en la cuerda del arco.
- **BowHandle:** Esta clase guarda la posición de agarre del arco, de la flecha en la cuerda y del lado en el que se encuentra dicha flecha.
- **BowAnimation:** Esta clase se encarga de generar la animación del arco según lo tenso que esté.
- **ArrowSpawner:** Clase que genera copias de los GameObject Arrows introducidos por parámetro, cuando un controlador interactúa con el objeto que contiene la clase.
- **VRTK_DeviceFinder:** Es un script del paquete VRTK el cual devuelve los GameObject de los controladores.

5.2. Diagramas de clases

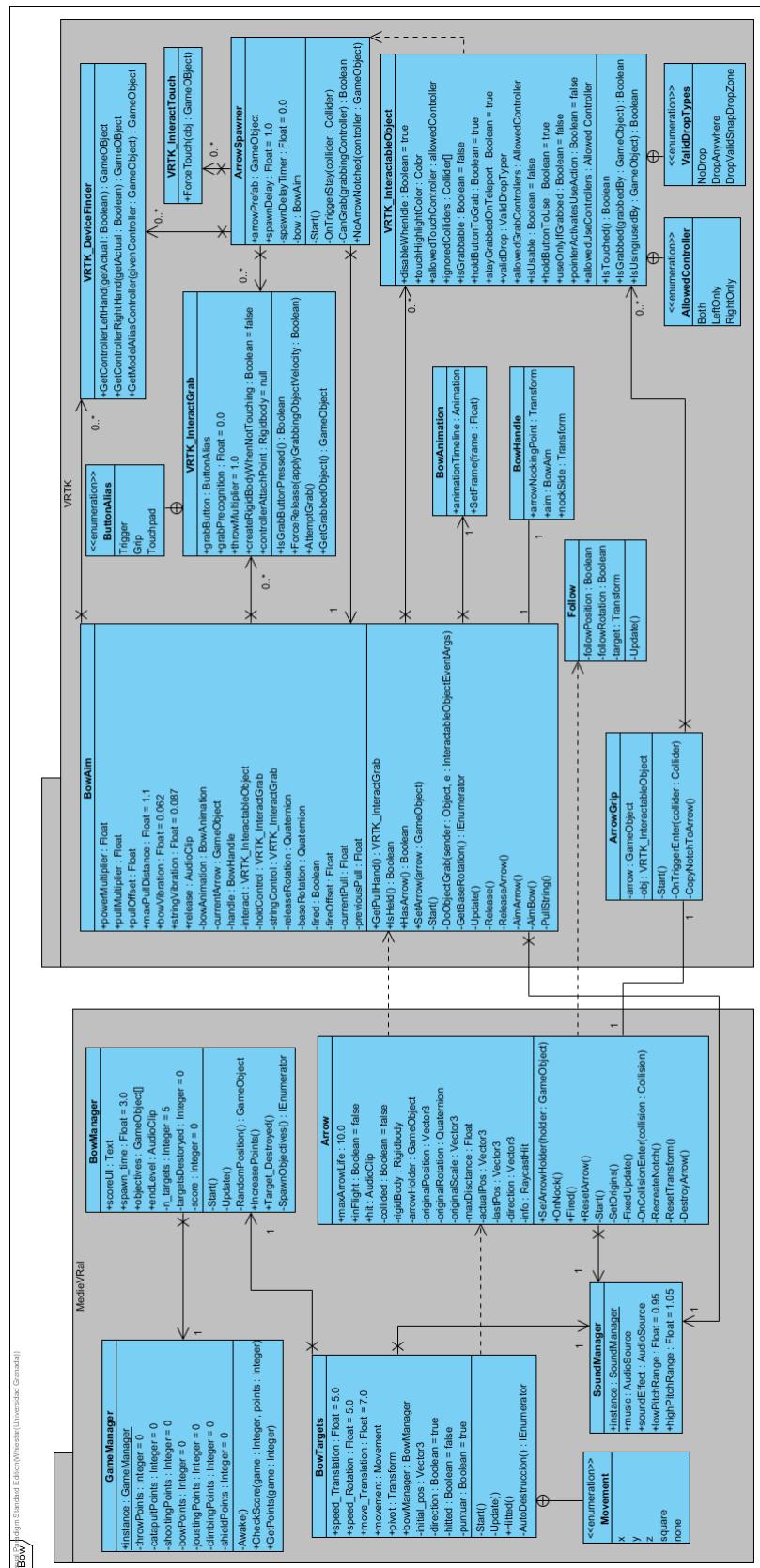


Figura 5.7: Diagrama de clases de la quinta prueba (Bow).

En la figura 5.8 se muestra el diagrama de clases de la sexta prueba 'Climbing'. En dicho diagrama se muestran las siguientes clases:

- **ClimbingManager:** Esta clase se encarga de detectar el número de bombas colocadas en el muro, gestionar la duración de la prueba, de manejar la puntuación antes de que finalice y de enviar la puntuación final a la clase *GameManager*.
- **BombSpawner:** Clase que genera copias de los GameObject Bombs introducidos por parámetro, cuando un controlador interactúa con el objeto que contiene la clase.
- **SlotBomb:** Es la clase encargada de detectar cuando se ha colocado un GameObject Bomb en los huecos del muro y de detonarlo.
- **Bomb:** Es la clase encargada de gestionar los efectos visuales creados por la bomba.
- **VRTK_HipTracking:** Es un script del paquete VRTK el cual mantiene un objeto en la cintura del jugador.
- **VRTK_ClimbableGrabAttach:** Es un script del paquete VRTK el cual permite agarrar un objeto para escalarlo.

En la figura 5.9 se muestra el diagrama de clases de la séptima prueba 'Shield'. En dicho diagrama se muestran las siguientes clases:

- **ShieldManager:** Esta clase se encarga de instanciar los projectiles en los puntos de aparición en la prueba 'Shield', gestionar la duración de la prueba, de manejar la puntuación antes de que finalice y de enviar la puntuación final a la clase *GameManager*.
- **ArrowProjectile:** Esta clase gestiona el comportamiento de los projectiles y de detectar cuando alcanzan al jugador o al escudo.
- **Shield:** Gestiona la funcionalidad del escudo, como objeto no soltable, y detecta cuando colisiona con los GameObjects ArrowProjectile.

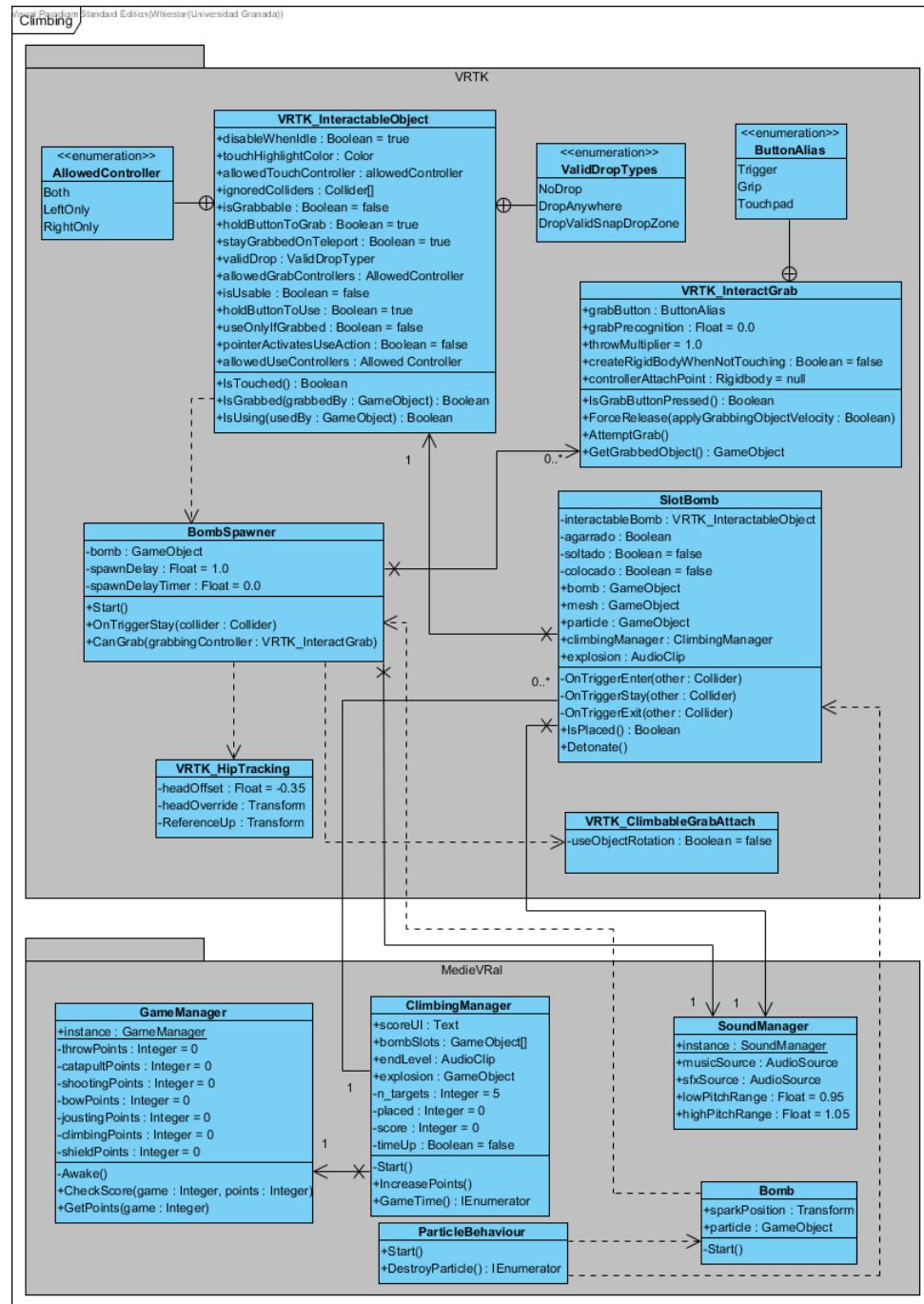


Figura 5.8: Diagrama de clases de la sexta prueba (Climbing).

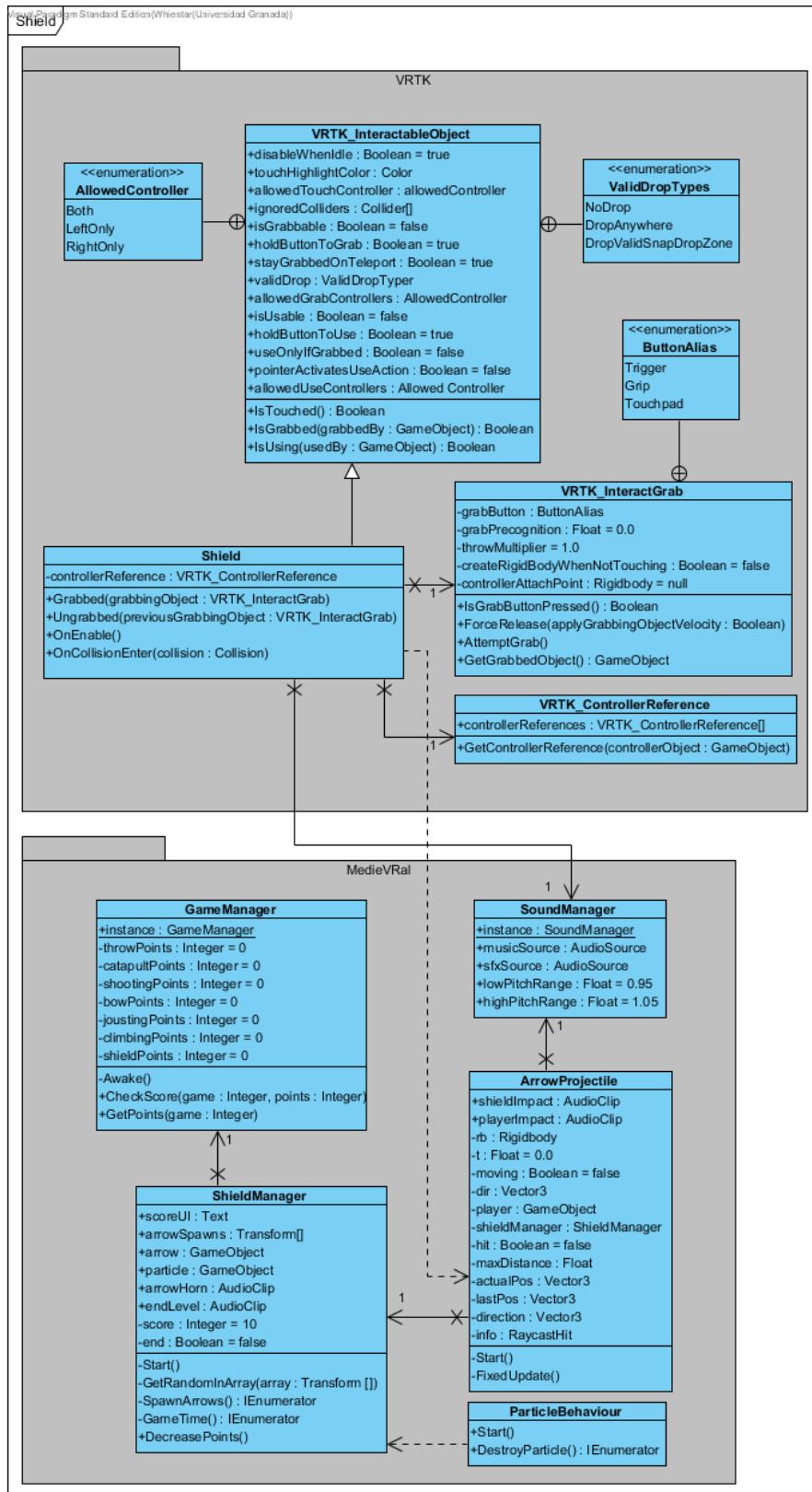


Figura 5.9: Diagrama de clases de la séptima prueba (Shield).

Capítulo 6

Implementación

Como resultado del proyecto se presenta una versión en fase alpha del juego diseñado. Esta versión tiene una jugabilidad limitada dado que el usuario necesitaría alrededor de 1 hora aproximadamente para completarlo y cuenta con una serie de modelos 3D en su mayoría de Unity Asset Store o de repositorios gratuitos, ya listados anteriormente.

6.1. Inicialización de HTC Vive

Para poder utilizar las gafas HTC Vive en unity, aparte de la instalación de las distintas herramientas necesarias descritas en el manual de usuario 8.2, también necesitamos incluir una serie de componentes en la escena.

Primero hay que crear un EmptyObject en la jerarquía de la escena, que lo podemos llamar ‘VRTKSDK’, y le añadimos el script *VRTK_SDK Manager*. Creamos un hijo para este objeto, que lo podemos llamar ‘SteamVRSDK’, al cual le incluimos como hijos los Prefabs, ubicados en ‘SteamVR/Prefabs’, [CameraRig] y [SteamVR]. Al GameObject SteamVRSDK tenemos que añadirle el script *VRTK_SDK Setup* y en la configuración del script seleccionamos SteamVR en el apartado ‘SDK Selection/Quick Slect’. 6.1

Y en GameObject VRTKSDK tenemos que configurar el script *VRTK_SDK Manager* de manera que en el apartado Setups hacemos clic en el símbolo ‘+’ y arrastramos su GameObject hijo SteamVRSDK al hueco que acaba de aparecer. 6.2

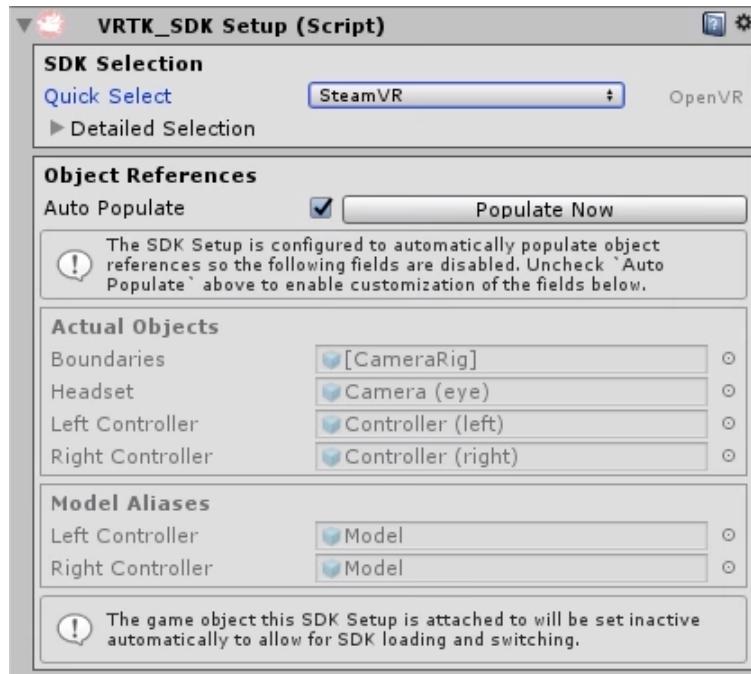


Figura 6.1: Configuración del script *VRTK_SDK Setup*.

Ahora para añadir scripts a los controladores creamos dos EmptyObjects y los llamamos ‘LeftController’ y ‘RightController’; y los añadimos en la configuración del script *VRTK_SDK Manager* 6.2 en el apartado ‘Script Aliases’. Por último añadimos los scripts necesarios para que los controladores tengan comportamientos adicionales. Por ejemplo, añadiendo los scripts *VRTK_Pointer*, *VRTK_Straight Pointer Renderer* y *VRTK_Controller Events* a un controlador nos permite lanzar un puntero pulsando el botón Touchpad, el cual nos teletransporta a la ubicación señalada.

De manera que la jerarquía que nos quedaría al final de la inicialización sería la que se corresponde con la figura 6.3.

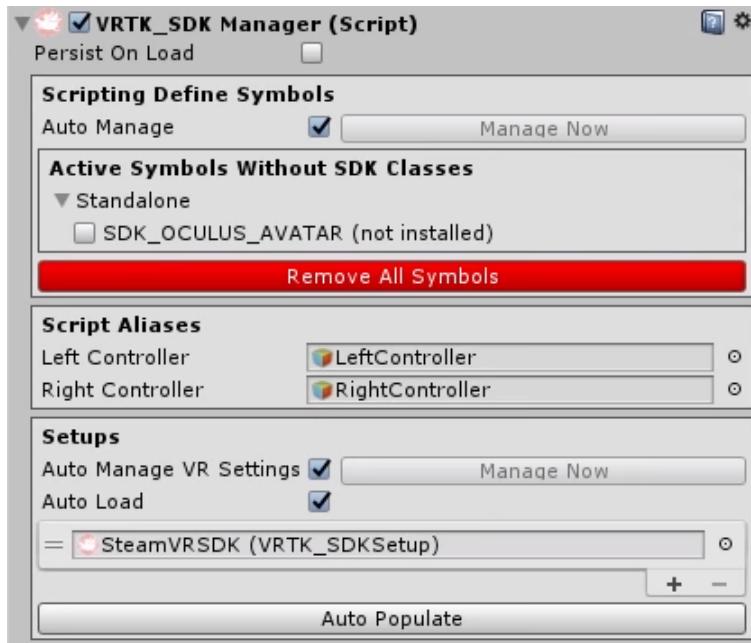


Figura 6.2: Configuración del script *VRTK_SDK Manager*.

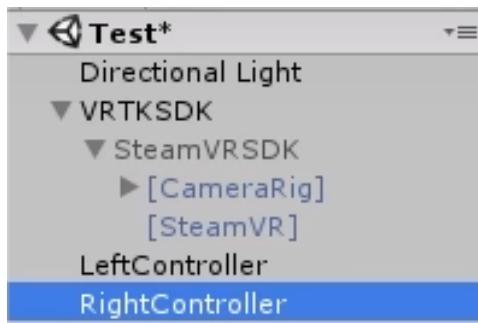


Figura 6.3: Jerarquía de GameObjets final.

6.2. Mecanismos de Interacción

Se han establecido unos controles básicos a la hora de interactuar con los mandos. Estos controles son:

- **Agarrar:** Permite al usuario que los objetos hereden la posición del controlador, pareciendo que el jugador ha agarrado el objeto. Para agarrar un objeto se utiliza el botón Grip.

- **Usar:** Permite que un objeto realice una acción asignada. Para usar un objeto se utiliza el botón Trigger.
- **Teletransportación:** Teletransporta al jugador a la posición designada por el puntero si la zona destino está habilitada. Para ello se utiliza el botón Touchpad.
- **Menú:** Abre el menú que habilita al usuario volver al nivel principal y consultar las puntuaciones. Para abrirlo se utiliza el botón System Menú y para elegir una opción se señala con el Touchpad y se selecciona con el Trigger.

6.3. Contenido y funcionalidades

A lo largo de la implementación se han definido una serie de Prefabs de Unity, de manera que estos objetos sean utilizables en futuras actualizaciones y modificaciones. Los Prefabs creados contienen una serie de componentes como transformaciones, colisionadores o scripts.

6.3.1. Selector de niveles

En este nivel 6.4 el jugador es capaz de seleccionar las distintas pruebas a las que puede jugar.



Figura 6.4: Escena de selección de nivel.

Los elementos de esta escena son:

- **TeleportPoints:** Son los destinos a los que el jugador puede teletransportarse mediante el puntero de los controladores. 6.5

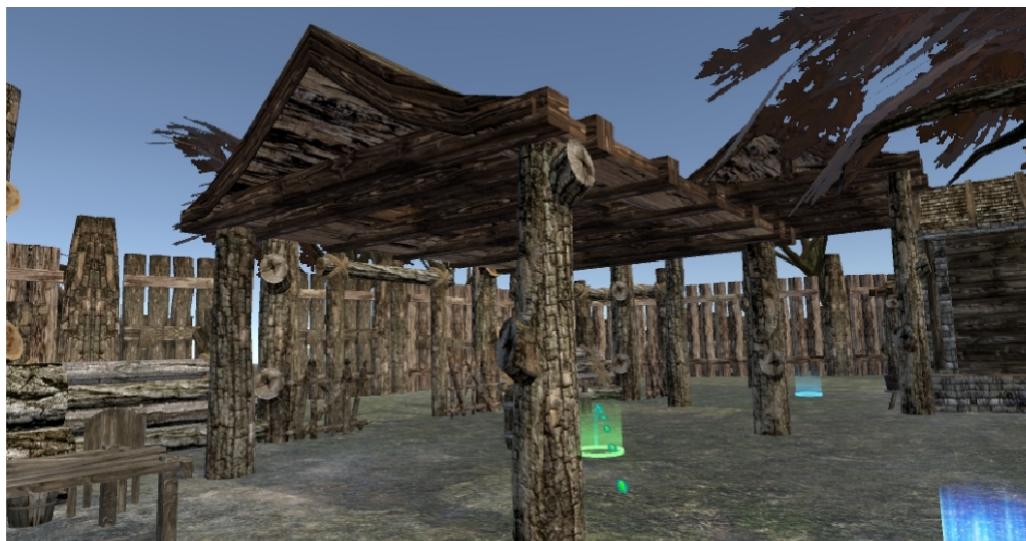


Figura 6.5: Teletransportación a un destino.

- **TeleportingSpheres:** Estos objetos son los encargados de detectar cuando el jugador acerca la esfera a las gafas y cargar la escena de la prueba correspondiente. 6.6



Figura 6.6: Esfera de selección de nivel.

6.3.2. Throwing

En esta prueba 6.7 el jugador debe alcanzar los objetivos con objetos lanzables.



Figura 6.7: Prueba Throwing.

Los elementos de esta escena son:

- **Dagger:** Son los objetos que el jugador puede agarrar con el botón Grip y lanzar a los objetivos. 6.7
- **DaggerSpawner:** Este objeto es el encargado de generar mas GameObject Dagger. Se encuentra en la cintura del jugador y hereda la posición del mismo. Cuando el jugador lo agarra se genera un GameObject Dagger en la posición del controlador.
- **Targets:** Estos objetos son los objetivos que el jugador tiene que alcanzar con los GameObject Dagger. 6.7



Figura 6.8: Objetivo alcanzado por daga.

6.3.3. Catapult

En esta prueba 6.9 el jugador debe alcanzar los muros y torres del castillo y las casas con la munición de la catapulta.



Figura 6.9: Prueba Catapult.

Los elementos de esta escena son:

- **Catapult:** Es el objeto con el que interactuaremos durante la prueba para alcanzar los objetivos. 6.9
- **Shuttle:** Es un objeto hijo de la catapulta el cual aplica una fuerza a la munición dependiendo de lo cargada que esté para lanzarla. 6.9
- **WheelRotator:** Es una manivela que controla la rotación de la catapulta. 6.10



Figura 6.10: Catapulta cargada y apuntando al objetivo.

- **WheelThrow:** Es una manivela que controla la posición del objeto Shuttle. 6.10
- **Lever:** Es la palanca que acciona el objeto Shuttle. 6.10
- **Rock:** Son los objetos munición que se colocan en la catapulta. 6.10
- **Castle:** Son los objetivos que el jugador tiene que alcanzar con los objetos munición. Hay dos tipos: Muro y Casa. 6.11



Figura 6.11: Explosión del objetivo alcanzado.

6.3.4. Shooting

En este nivel 6.12 el jugador tiene que acertar en distintos objetivos usando una pistola de pólvora.



Figura 6.12: Prueba Shooting.

Los elementos de esta escena son:

- **Target:** Estos objetos son los objetivos que el jugador tiene que al-

canzar con los objetos Ammo. 6.13



Figura 6.13: Objeto alcanzado por bala.

- **Ammo:** Son los objetos que tienen que alcanzar los objetos Target a partir del objeto FlintLockPistol. Para cargar la pistola hay que colocarla en la boca del arma. 6.12
- **FlintLockPistol:** El usuario puede agarrar este objeto y si acerca el objeto GunPowder y Ammo a la boca de la pistola la cargará y podrá disparar el objeto Ammo. 6.12
- **GunPowder:** Este objeto es necesario a la hora de que el objeto Ammo salga disparado. Para ello se colisiona este objeto con la boca del arma antes de cargar el objeto Ammo. Se encuentra en la cintura del jugador y contiene un snap point, que es un punto de agarrado donde se puede dejar y coger el objeto. 6.14
- **AmmoSpawner:** Este objeto es el encargado de generar mas objetos Ammo. Cuando el jugador lo agarra se genera un objeto Ammo en la posición del controlador.



Figura 6.14: Objeto gunpowder cargando el arma.

6.3.5. Jousting

En este nivel 6.15 el jugador tiene que acertar en distintos objetivos mientras que se mueve a lo largo del mapa montado a caballo.

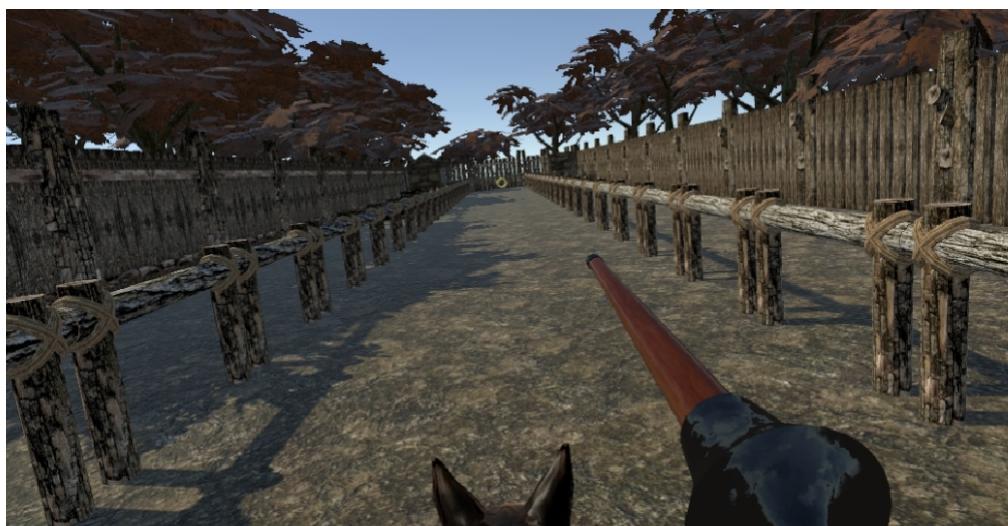


Figura 6.15: Prueba Jouster.

Los elementos de esta escena son:

- **Horse:** Es el objeto caballo el cual se mueve hasta el final del nivel y

el jugador hereda su posición.

- **Lance:** Es el objeto que fuerza su agarre en el controlador derecho del jugador, con el que hay que alcanzar los objetivos.
- **Ring:** Estos objetos son los objetivos que el jugador tiene que alcanzar con el objeto Spear.



Figura 6.16: Objetivo de Jouesting.

6.3.6. Bow

En este nivel 6.17 el jugador tiene que acertar en distintos objetivos usando un arco y flechas. Para esta prueba se ha utilizado el modelo de un arco animado del paquete VRTK diseñado con Figuras 3D básicas debido a la dificultad de encontrar un modelo 3D animado de un arco.



Figura 6.17: Prueba Bow.

Los elementos de esta escena son:

- **Target:** Estos objetos son los objetivos que el jugador tiene que alcanzar con los objetos Arrow. 6.18
- **Arrow:** Son los objetos que tienen que alcanzar los objetos Target a partir del objeto Bow. 6.17
- **Bow:** El usuario puede agarrar este objeto y si acerca una flecha a la cuerda del arco podrá lanzarla con una fuerza dependiendo de lo tenso que esté. 6.17
- **ArrowSpawner:** Este objeto es el encargado de generar mas objetos Arrow. Se encuentra en la cintura del jugador y hereda la posición del mismo. Cuando el jugador lo agarra se genera un objeto Arrow en la posición del controlador.

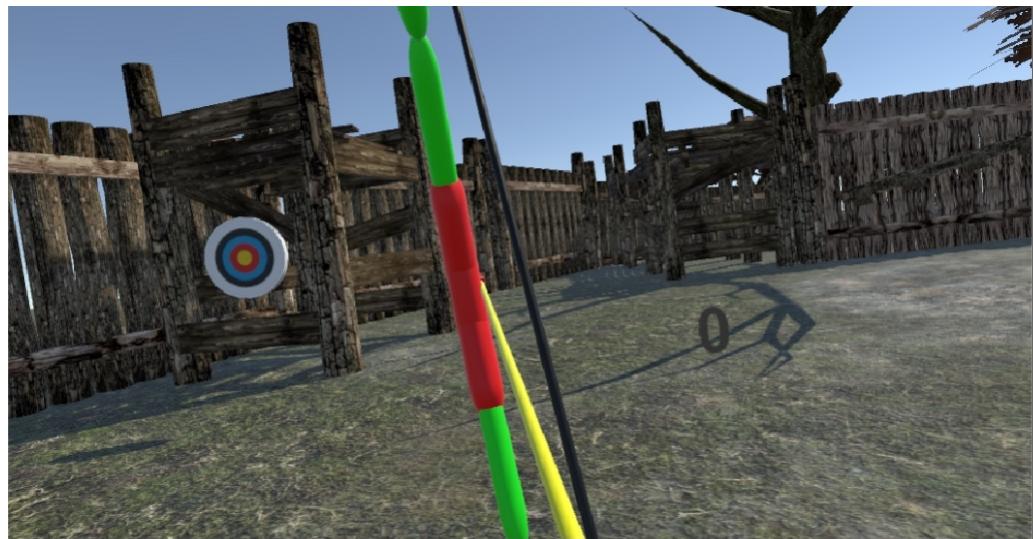


Figura 6.18: Arco tensado.



Figura 6.19: Objetivo alcanzado por flecha.

6.3.7. Climbing

En este nivel 6.20 el jugador tiene que colocar bombas en unos huecos del muro del castillo mientras lo escala.



Figura 6.20: Prueba Climbing.

Los elementos de esta escena son:

- **Bomb:** Es el objeto bomba el cual el jugador tiene que colocar en los huecos del muro. 6.21
- **BombHip:** Es el objeto en la cintura del jugador que genera mas objetos Bomb. Cuando el jugador lo agarra se genera un objeto Bomb en la posición del controlador. 6.21



Figura 6.21: Objetos BombHip y Bomb.

- **BombSlots:** Estos objetos indican donde se tienen que colocar las bombas en el muro. 6.22
- **ClimbableBricks:** Estos objetos son los que permiten al jugador agarrarse y escalar el muro. 6.22

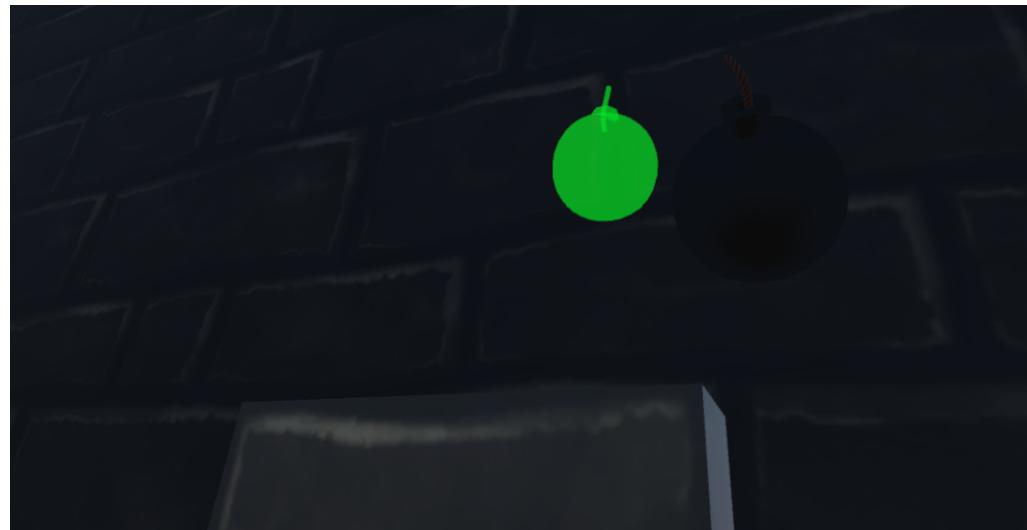


Figura 6.22: Objetos BombSlot y ClimbableBricks.

6.3.8. Shield

En este nivel 6.23 el jugador tiene que defenderse de proyectiles con un escudo. Si el jugador es alcanzado por un proyectil perderá puntos.

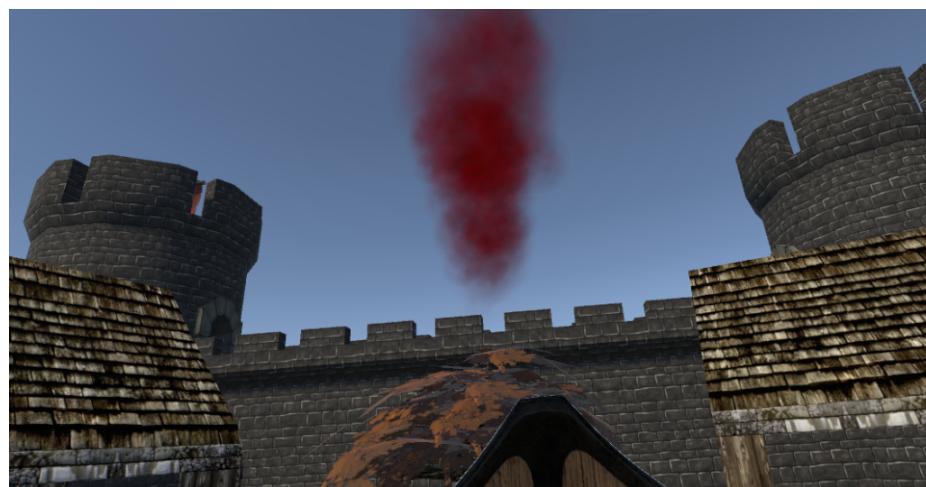


Figura 6.23: Prueba Shield.

Los elementos de esta escena son:

- **RedSmoke**: Es el objeto que indica por donde va a venir un proyectil. 6.23
- **Shield**: Es el objeto que fuerza su agarre en el controlador derecho del jugador, con el que hay que defenderse de los proyectiles. 6.24
- **ArrowProjectiles**: Estos objetos son los proyectiles que intentan alcanzar al jugador. 6.24
- **ArrowSpawns**: Estos objetos son las posiciones donde se instancian los objetos ArrowProjectiles.

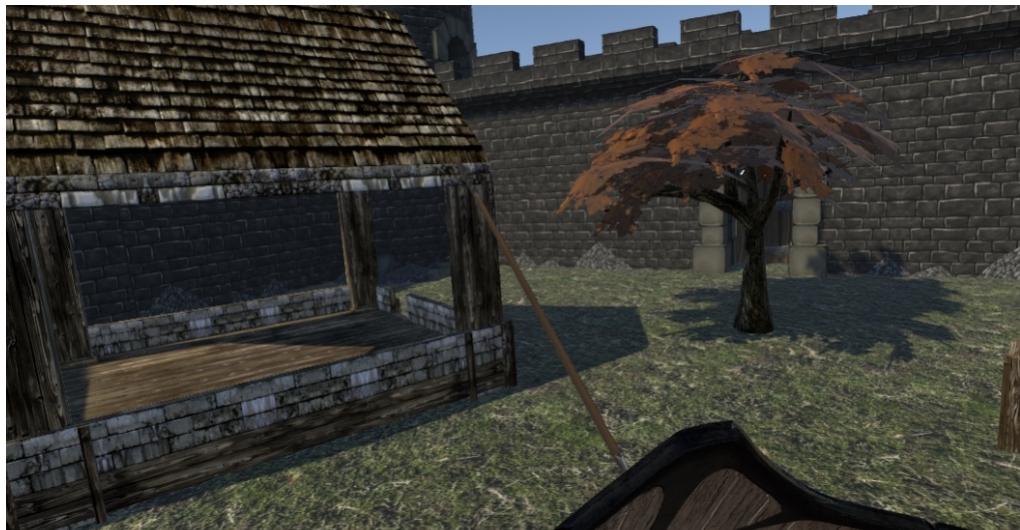


Figura 6.24: Objeto ArrowProjectile clavada en el escudo.

6.4. Problemas resueltos

6.4.1. Colisiones en movimiento

A la hora de detectar colisiones en movimiento los componentes Collider no son capaces de devolver el momento exacto en el que colisionan, obteniendo falsas colisiones. Para solucionar este problema se ha implementado el uso de Raycasting [28]. Al igual que para su uso en el modelado de sólidos [29] también se puede utilizar para comprobar futuras colisiones realizando una proyección del objeto en la dirección en la que avanza.

Para ello se definen una serie de variables: lastPos (la posición anterior del objeto), actualPos (la posición actual del objeto), direction (la dirección del objeto) y maxDistance (la distancia máxima a la que se va a comprobar si hay colisión). Para cada actualización de la posición del objeto se calcula:

```

1     lastPos = actualPos;
2     actualPos = transform.position;
3     direction = (actualPos - lastPos);
4     maxDistance = Vector3.Distance(actualPos, lastPos);
5

```

Listing 6.1: Actualización de variables

De manera que con las variables actualizadas se lanza el método *Physics.Raycast* para calcular si desde la posición anterior en una dirección con una distancia máxima el objeto colisionará:

```

1     Debug.DrawRay(lastPos, direction, Color.green, 10f);
2     //Se usa la herramienta Raycast para generar rayos que calculen
3     //la colision con los objetivos dada una distancia maxima.
4     if (Physics.Raycast(lastPos, direction, out info, maxDistance))
5     {
6         ...
7     }

```

Listing 6.2: Raycasting empleado en el GameObject Dagger

Si detecta que va a colisionar, el objeto *info* contendrá la información de la colisión y se comprueba si va a colisionar con un objetivo, en este caso una diana:

```

1
2
3     Debug.DrawRay(lastPos, direction, Color.red, 10f);
4     //Se comprueba si se ha colisionado con una diana.
5     if (info.collider.gameObject.tag == "Diana")
6     {
7         Debug.DrawRay(lastPos, direction, Color.blue, 10f);
8     }

```

Listing 6.3: Colisión con el objeto Diana

A partir de la función ‘Debug.DrawRay’ se puede observar en tiempo de ejecución el funcionamiento de Raycast. Se han dibujado de verde cuando no se detectan colisiones, de rojo cuando se detectan colisiones con otros objetos que no sean objetivos y de azul cuando se va a alcanzar un objetivo.

Como se aprecia en la primera figura 6.25 al realizar un lanzamiento fallido se detectaba en rojo la colisión de la daga con el controlador y una vez lanzado, no detectaba colisiones. En cambio en la segunda figura 6.26

antes de llegar a colisionar se detectan distintas proyecciones del objeto que podían colisionar con el objetivo por lo que hay pequeños trazos azules a lo largo de la linea, consiguiendo el punto exacto de colisión.

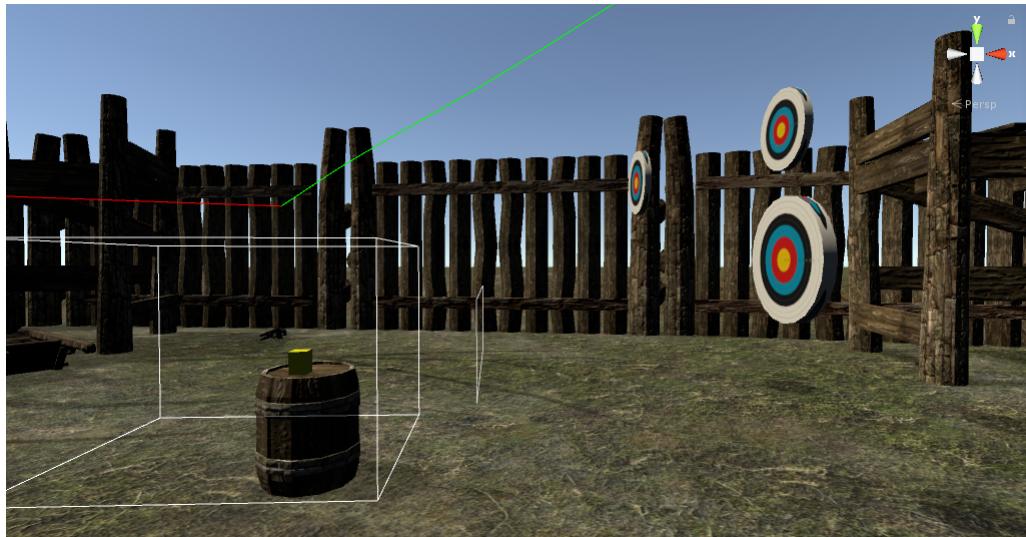


Figura 6.25: Lanzamiento fallido.

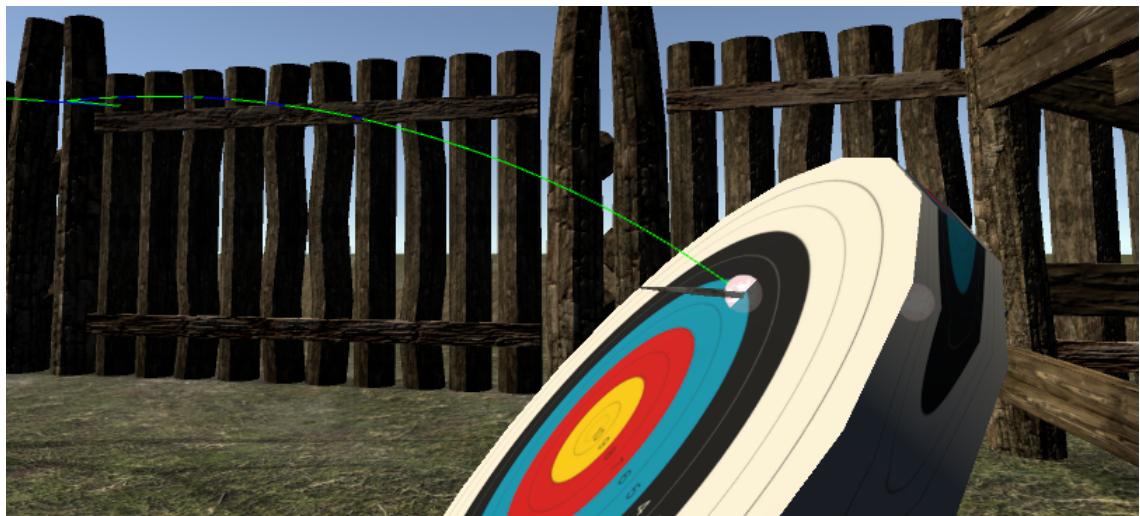


Figura 6.26: Lanzamiento en el blanco.

6.4.2. Cambiar y repetir el nivel

A la hora de poder volver al selector de nivel o repetir la misma prueba se ha incluido un menú que además permite consultar la puntuación del jugador

6.27. Este menú hereda la posición y la rotación del headset de manera que siempre se encuentra en el centro de la vista del jugador cuando está activo. Para activar y desactivar el menú se utiliza un script para gestionar los eventos de cuando se pulsa el botón ‘Menu Application’ del controlador izquierdo. Para seleccionar la opción de volver al selector de nivel o repetir la prueba actual se utiliza el puntero del controlador izquierdo con el botón ‘Touchpad’ y se pulsa el botón ‘Trigger’ mientras se señala la opción.



Figura 6.27: Menú del usuario.

Capítulo 7

Pruebas de usabilidad

7.1. Pruebas realizadas

A continuación se muestran los resultados de las pruebas de uso realizadas a 5 usuarios con distinta experiencia como jugadores, donde únicamente el usuario 4 ha probado con anterioridad la RV. El número de usuarios que han participado en las pruebas es reducido debido, por un lado, a que las gafas HTC vive y sus componentes requieren cierto cuidado por lo que las pruebas debían de estar supervisadas y atendidas en todo momento, y por otro lado, y también relacionado con el motivo anterior, debido a la falta de tiempo para buscar más usuarios que puedan participar en las pruebas.

La evaluación, cuyo resultado se muestra en la tabla 7.1, se ha enfocado exclusivamente a la capacidad del individuo para adaptarse al juego y lo cómodo que se sienta jugándolo, teniendo en cuenta factores como mareos o náuseas. A su vez, las limitaciones del trabajo en tiempo no han permitido la realización del estudio sobre más individuos.

Los parámetros medidos son los siguientes:

- **Edad:** Se indica la edad del jugador.
- **Relación con los juegos:** Muestra la experiencia que tiene el jugador en el campo de los videojuegos. Los posibles valores son:
 1. Veterano
 2. Experimentado
 3. Conocedor
 4. Jugador ocasional

5. Apenas experimentado

- **Dispositivo más usado:** Indica que dispositivos utiliza el usuario mas a menudo para jugar a videojuegos.
- **Adaptación al dispositivo:** Indica la facilidad que ha tenido el usuario para adaptarse a los mecanismos de la RV. Los posibles valores son:
 1. Se ha desenvuelto sin problemas.
 2. Ha tenido pocos problemas para adaptarse.
 3. Le ha costado adaptarse al dispositivo.
 4. Ha tenido bastantes problemas para desenvolverse en la RV.
 5. No ha conseguido dominar los mecanismos.
- **Inmersión:** Indica si el jugador se ha sentido cómodo y ha conseguido desenvolverse en el mundo de RV como si fuera el mundo real.
- **Finalización del juego:** Indica si el jugador ha sido capaz de obtener buenas puntuaciones en las pruebas y conseguir completarlas todas.

Usuario	1	2	3	4	5
Edad	63	62	22	30	16
Relación con los juegos	1	2	2	5	3
Dispositivo más usado	Dispositivo Móvil	Dispositivo móvil y PC con ratón y teclado	Consola con mando	PC con ratón y teclado	Dispositivo móvil y consola con mando
Adaptación al dispositivo	4	3	3	1	1
Inmersión	No	Si	Si	Si	Si
Finalización del juego	No	No	Si	Si	Si

Cuadro 7.1: Resultados de las pruebas sobre los usuarios.

7.2. Conclusiones sobre las pruebas realizadas

Al haber supervisado todas las pruebas he podido comprender a primera vista como a jugadores con mas experiencia y menor edad le ha sido mas fácil adaptarse al medio en el que se encontraban, en cambio los jugadores

de edad más avanzada y a la vez menos experiencia en el campo de los videojuegos les costaba mas hacerse a los controles y al entorno del primer nivel. Los comentarios mas destacados al terminar de jugar han sido:

“Al no poder ver tu propio cuerpo no sabes realmente si estas flotando o si te mantienes sobre el suelo” - Usuario 1

“Es incomodo adaptarse al principio, pero una vez que le pillas el truco es fácil jugar” - Usuario 3

“La prueba de la catapulta es menos natural que las demás” - Usuario 4

El primer usuario, al tener una experiencia en los videojuegos casi nula, le costo bastante tiempo adaptarse al entorno del mundo y a los controles de las HTCVive. Aun así consiguió superar con buena puntuación las primeras cuatro pruebas. Su conclusión al primer contacto con RV fue:

“No me parece que sea como el mundo real, los objetos no pesan al cogerlos y es difícil saber cuan lejos los puedes lanzar”

El segundo usuario, a pesar de no estar familiarizado demasiado con los videojuegos, consiguió desenvolverse mejor que el primer usuario, teniendo casi la misma edad. Superó 5 de las 7 pruebas dado que en la prueba de justa al tener un movimiento no voluntario en el caballo le daban mareos. Su conclusión al terminar de jugar fue:

“El mundo es bonito, parece que estas en campamento de bandidos de verdad”

El tercer usuario mostró un gran interés en disfrutar del entorno y de los paisajes del juego, intentaba descubrir si podía ver diferentes partes del mapa moviéndose por diferentes sitios. Una vez había explorado el entorno decidió enfocarse en las pruebas, llegando a pasarse todas sin demasiada dificultad. Su conclusión al terminar de jugar fue:

“El botón de agarrar al estar colocado en la empuñadura parece que estas agarrando de verdad los objetos”

El cuarto usuario, al tener gran experiencia en los videojuegos y en su diseño, no demostró apenas dificultad en adaptarse al entorno o a los controles y se centro en obtener las máximas puntuaciones posibles. Llego a terminar todas las pruebas con casi puntuaciones perfectas en apenas media hora. Su conclusión al terminar de jugar fue:

“En general la mayoría de escenarios son intuitivos pero algunos como la catapulta o la justa no se sienten del todo naturales”

El quinto usuario no tuvo apenas problemas en adaptarse al juego, al principio no conseguía coger bien los objetos y los soltaba mientras los intentaba agarrar, pero en general se desenvolvió sin problemas. Completo

todas las pruebas, aunque tuvo que repetir varias veces la del lanzamiento de cuchillos. Al finalizar el juego su conclusión fue:

“A veces quieres parar y soltar los mandos en mesas del juego y no te das cuenta de que se caen al suelo”

Una vez dispongo de todos los datos, casi todos los usuarios se han sentido inmersos en la RV y 3 de 5 usuarios han conseguido superar todas las pruebas. Además, la adaptación de los usuarios a las gafas y controladores ha sido en general alto tanto en jugadores con menos experiencia en videojuegos como en los que más. De todos los usuarios 4 volverían a elegir las gafas de realidad virtual para volver a jugar. La progresión de los jugadores en el juego en la mayoría de casos ha sido progresiva, pasaban de no saber moverse bien y no saber usar bien los controladores a desenvolverse sin muchos problemas en el mundo virtual. Por lo tanto, se podría deducir que no hay pruebas concluyentes de que la experiencia en juegos influya en la inmersión, aunque es probable que la edad del usuario si sea mas influente, dado que los usuarios de mayor edad les ha sido mas complicado moverse e interactuar con el mundo en los primeros minutos.

Como conclusión a las pruebas, a los usuarios les ha parecido mas inmersivo las pruebas donde lanzaban objetos como el lanzamiento de cuchillo, el disparo con pistola o el tiro con arco y también la prueba de la escalada y el escudo, debido a esto es posible concluir que estas pruebas tienen un detalle en común y es el control de la movilidad del personaje y el control de los objetos de su entorno. La prueba de la catapulta no es tan inmersiva debido a que al dirigirla con manivelas y palancas no se siente como si realmente se estuviera realizando esa acción; y la prueba de la justa al no tener el control del caballo sino simplemente avanzar con él, no deja al usuario experimentar el control de la realidad virtual.

Capítulo 8

Conclusiones y trabajos futuros

8.1. Conclusiones

Durante la realización de este Trabajo de Fin de Grado he podido ampliar mi conocimiento sobre diferentes herramientas como Unity, que me ha sido enseñada en asignaturas de este año, y HTC Vive.

En el capítulo 1, sección de introducción, se propusieron una serie de objetivos, cuyos resultados y conclusiones serán comparados.

Al principio del proyecto se realizó un estudio suficientemente completo de la herramienta Unity a partir de su documentación y se realizó una documentación de todos los ejemplos de la librería VRTK 8.2. Además, se realizaron ejemplos de escenas utilizando la librería VRTK en Unity para comprobar su funcionalidad.

El estudio realizado sobre los factores inmersivos me ha permitido desarrollar un juego enfocado a que el jugador se sienta cómodo jugando y que estos incluidos inciten al jugador a abstraerse del mundo real.

En este proyecto se ha desarrollado un videojuego completamente funcional que incluye los factores inmersivos presentados en el capítulo 4. Debido a las diferentes maneras que existen en el juego para interactuar con el mundo, estas permiten que el jugador encuentre las pruebas mas afines a él. Este juego puede suponer una herramienta útil para las personas que pueden tener su primer contacto con la RV, ya que al estar enfocado a la inmersión, es mas fácil para el jugador acostumbrarse a la RV.

De hecho en las pruebas realizadas, aunque ha sido con una cantidad pequeña de usuarios, la mayoría de ellos se han sentido cómodos jugando y han conseguido “sumergirse” en la Realidad Virtual a pesar de la Edad o su

experiencia en videojuegos.

En general, en este proyecto se ha podido apreciar el potencial que tienen los dispositivos de RV a pesar de no disponer de una habilidad de programación alta ni de un equipo de desarrollo.

8.2. Trabajos futuros

Dado que este proyecto ha sido realizado desde un principio con vistas a mejorarlo y poder crear un juego mas complejo, se realizarán posteriormente modificaciones y aportaciones al videojuego que actualmente se encuentra en fase alpha. Algunas de las aportaciones que se podrían realizar al proyecto son las siguientes:

- **Cambio de estética de los mandos:** A los mandos se les podría añadir una apariencia de manos con animaciones según la acción que realice el usuario, es decir según el botón que pulse del controlador. Orientado a la estética de este juego podrían incluirse unos guanteletos de metal como si fueran las manos del caballero.
- **Añadir más pruebas o ampliar las existentes:** Se podrían incluir pruebas de carácter más complejo como una prueba de espadas en la que se encuentre una solución a la colisión entre objetos devolviendo feedback al jugador o mejorar pruebas ya existentes como la justa en la que se podría incluir una IA con la que hay que luchar.
- **Mejorar el entorno del juego:** Cuanto más desarrollado se encuentre el entorno de un juego mas facilidad tiene el usuario para perderse en él y mucho más si la finalidad de este juego es la inmersión. Mejorar el mapa del juego implicaría aumentar la inmersión del jugador.
- **Mejorar la mecánica de Teletransporte:** La mejor solución hasta el momento para conseguir desplazar la Play Area del jugador es a partir de la teletransportación, pero en el momento en el que se encuentre una mecánica que no limite la movilidad del jugador en una habitación, éste no será consciente en ningún momento de que puede chocarse con las paredes y por tanto la inmersión será aún mayor.

Bibliografía

- [1] HTC Corporation. About htc vive. <https://www.vive.com/eu/>, Consultado en Agosto de 2018.
- [2] Valve Corporation. About steamvr performance test. https://store.steampowered.com/app/323910/SteamVR_Performance_Test/, Consultado en Agosto de 2018.
- [3] HTC Corporation. About htc vive set up. <https://www.vive.com/us/setup/vive/>, Consultado en Agosto de 2018.
- [4] Oculus Corporation. About oculus rift. <https://www.oculus.com/>, Consultado en Agosto de 2018.
- [5] Sony Corporation. About playstation vr. <https://www.playstation.com/es-es/explore/playstation-vr/>, Consultado en Agosto de 2018.
- [6] Valve Corporation. About steamvr. <https://steamcommunity.com/steamvr>, Consultado en Agosto de 2018.
- [7] Sysdia Solutions Ltd. About virtual reality toolkit. <https://vrtoolkit.readme.io/>, Consultado en Agosto de 2018.
- [8] axe_93. Medieval catapult 3d model. <https://free3d.com/3d-model/medieval-catapult-74683.html>, Consultado en Agosto de 2018.
- [9] Arms Museum. Flintlock pistol 3d model. <https://sketchfab.com/models/792ca024195d4292829d8d0b304ced30#>, Consultado en Agosto de 2018.
- [10] ZacLinMac. Cannon ball 3d model. <https://www.turbosquid.com/3d-models/free-cannon-ball-3d-model/696049>, Consultado en Agosto de 2018.
- [11] Horse with saddle 3d model. <http://www.cadnav.com/3d-models/model-46223.html>, Consultado en Agosto de 2018.

- [12] Rings model. https://www.models-resource.com/custom_edited/sonicthehedgehogcustoms/model/10616/, Consultado en Agosto de 2018.
- [13] Vertici. 3d simple lance. <https://www.turbosquid.com/FullPreview/Index.cfm/ID/1286837>, Consultado en Agosto de 2018.
- [14] evilvoland. Bomb 3d model. <https://www.turbosquid.com/FullPreview/Index.cfm/ID/1034107>, Consultado en Agosto de 2018.
- [15] Pino4et. Old damaged shield. <https://www.turbosquid.com/3d-models/free-shield-corona-pbr-3d-model/1074562>, Consultado en Agosto de 2018.
- [16] 3dregenerator. Bow quiver arrow 3d model. <https://free3d.com/3d-model/bow-quiver-arrow-46863.html>, Consultado en Agosto de 2018.
- [17] svyart. Short rope 3d model. <https://www.cgtrader.com/free-3d-models/various/various-models/short-rope>, Consultado en Agosto de 2018.
- [18] CC Attribution Creative Commons Attribution. Sand pile model. <https://sketchfab.com/models/da0abb659bc140fa9836f90f2773383d>, Consultado en Agosto de 2018.
- [19] Antonio Bautista Bailón Morillas. Metodología agil en la programación de videojuegos. <http://decsai.ugr.es/~bailon/clases/PL/metodologia.html#/metodologia>, Consultado en Agosto de 2018.
- [20] Kenneth S Rubin. *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.
- [21] GanttProject. About gantt. <https://www.ganttpointer.biz/>, Consultado en Agosto de 2018.
- [22] Inc. GitHub. About github. <https://github.com/>, Consultado en Agosto de 2018.
- [23] Atlassian. About trello. <https://trello.com/>, Consultado en Agosto de 2018.
- [24] Epic Games. About unreal engine. <https://www.unrealengine.com/en-US/what-is-unreal-engine-4>, Consultado en Agosto de 2018.

- [25] Unity Technologies. About unity. <https://unity3d.com/es>, Consultado en Agosto de 2018.
- [26] Valve Corporation. About the lab. https://store.steampowered.com/app/450390/The_Lab/?l=spanish, Consultado en Agosto de 2018.
- [27] Kevin M Malloy and Leonard S Milling. The efectiveness of virtual reality distraction for pain reduction: a systematic review. *Clinical psychology review*, 30(8):1011–1018, 2010.
- [28] Unity Technologies. Documentation about raycast. <https://docs.unity3d.com/ScriptReference/Physics.Raycast.html>, Consultado en Agosto de 2018.
- [29] Scott DRoth. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18(2):109–144, 1982.
- [30] Valve Corporation. Download steamvr. <https://store.steampowered.com/about/>, Consultado en Agosto de 2018.
- [31] Unity Technologies. Download unity v.2017.3. <https://unity3d.com/es/get-unity/download/archive>, Consultado en Agosto de 2018.
- [32] Unity Technologies. Unity getting started. <https://docs.unity3d.com/es/current/Manual/GettingStarted.html>, Consultado en Agosto de 2018.
- [33] Juan Alberto Martínez López. Documentación de los ejemplos de la libreria vrtk. <https://drive.google.com/open?id=14BSAvpXbifWmOlFQzSbOZVHkQR8ZhCbG>, Consultado en Agosto de 2018.
- [34] Miguel Vega. Casos de uso. <https://lsi.ugr.es/~mvega/docis/casos%20de%20uso.pdf>, Consultado en Agosto de 2018.

Apéndices

Anexo I: Manual de instalación

En esta sección se muestran los requisitos y los pasos para instalar el proyecto.

Hardware Necesario

Para el uso de este proyecto es necesario los siguientes componentes hardware:

- Unas gafas de realidad virtual HTC Vive.
- Tarjeta gráfica: NVIDIA Asus Dual GTX 1060 3GB (para tener una experiencia fluida).
- CPU: Intel Core i5 4590 .
- RAM: 8GB .
- Un puerto HDMI.
- Un puerto USB 2.0 .
- Sistema Operativo: Windows 7 o superior.

Software Necesario

Para el uso de este proyecto es necesario el siguiente software:

- Unity V.2017.3 .

- El paquete SteamVR que contiene la SDK de las gafas HTC Vive.
- El paquete VRTK (Virtual Reality Tool Kit).

Instalación

Primero instalamos Unity V.2017.3 [31] en nuestra máquina, tenemos que asegurarnos de que es la versión 2017.3.x (la subversión no hace falta que sea ninguna específicamente) ya que sino puede haber incompatibilidades en el proyecto, e instalamos también Steam [30].

Dentro de Steam, después de haber iniciado sesión, colocamos el ratón sobre la pestaña 'biblioteca' y nos aparecerá una lista de opciones en la que seleccionamos 'herramientas'. Ahora nos aparecerán distintos software en los que buscamos 'SteamVR' hacemos clic izquierdo y seleccionamos instalar.

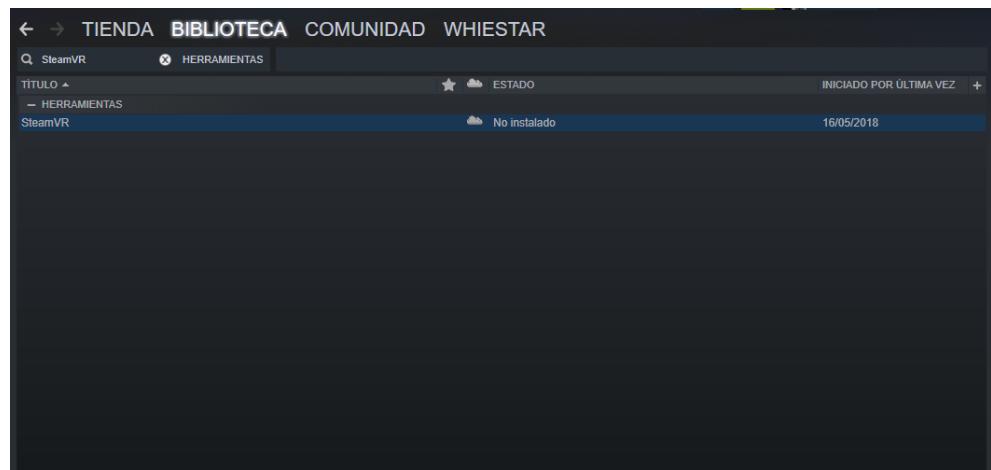


Figura 1: SteamVR en el menú herramientas.

Configuración

Al iniciar Unity [32] por primera vez nos aparecerá un menú para elegir los proyectos existentes, el cual estará vacío ya que no detectará ninguno.

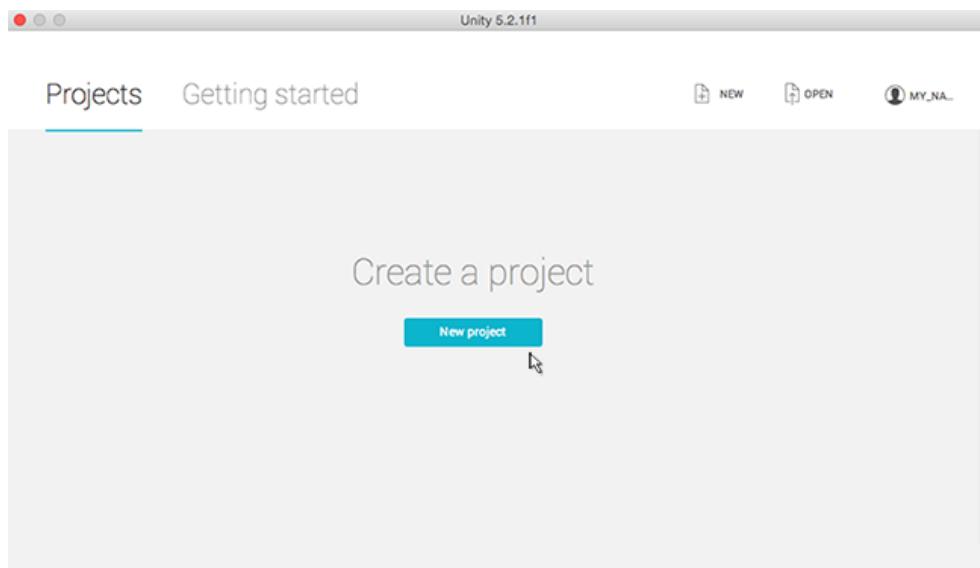


Figura 2: Pantalla de Inicio en Unity.

En el menú de arriba a la derecha seleccionamos la opción 'open' 2 y buscamos el directorio donde se encuentra el proyecto. En el caso de que no se hayan importado los assets de SteamVR y VRTK, dentro del editor de unity con el proyecto abierto vamos a la pestaña de Asset Store donde buscamos los assets de SteamVR y VRTK y hacemos click en 'import'.

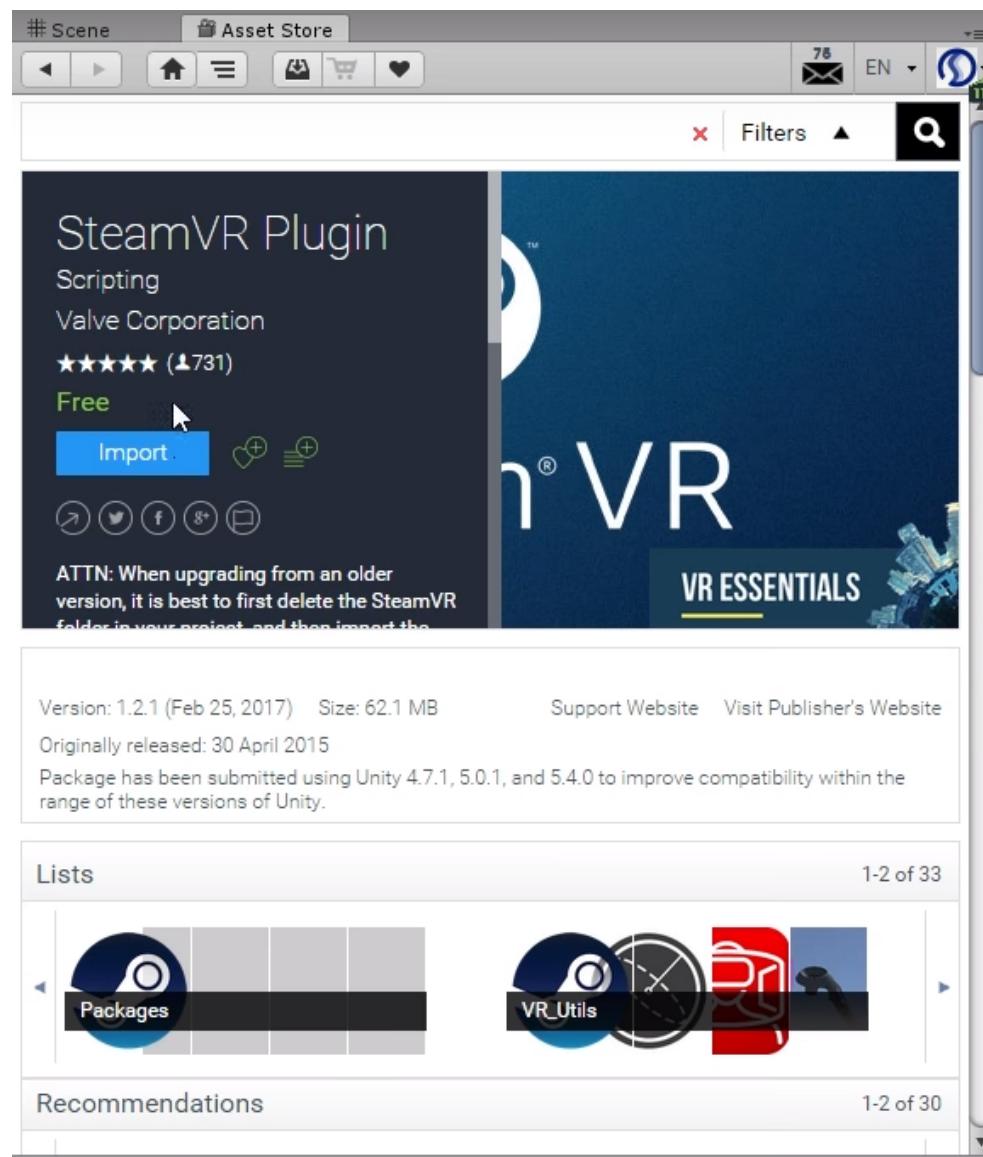


Figura 3: Asset de SteamVR.

Anexo II: Manual de Usuario

Para comenzar el juego ejecutamos la escena SelectLevel, en dicha escena podemos utilizar el botón 'Touchpad' de los mandos para teletransportarnos a las distintos puntos de destino del escenario. En dicho escenario nos encontramos con bolas de cristal que reflejan otros escenarios (pruebas); para acceder a ellos solo hace falta coger dicha bola y acercárnosla a las gafas.

Al principio la mayoría de puntos de destino estarán bloqueados hasta que superemos las pruebas anteriores.



Figura 4: Puntos de destino bloqueados.

Para activar y desactivar el menú se utiliza el botón ‘Menu Application’ del controlador izquierdo. Para seleccionar la opción de volver al selector de nivel o repetir la prueba actual se utiliza el puntero del controlador izquierdo con el botón ‘Touchpad’ y se pulsa el botón ‘Trigger’ mientras se señala la opción.



Figura 5: Menú del usuario.

A continuación se definirán las distintas pruebas dentro del juego:

- **Lanzamiento de objetos:** Consiste en lanzar un objeto, agarrable por el player, a distintas dianas desde una zona del mapa. En la cintura del player se encontrará una un conjunto de dagas las cuales le permite coger munición con el botón Grip del controlador. Los objetos dianas se encuentran a una distancia donde el jugador no puede llegar para darles con el objeto en la mano y tiene que lanzarlo. Cuando dichas dianas son golpeadas por el objeto o pasa un tiempo límite desaparecen. Cuantas más dianas golpee el usuario más puntos conseguirá.



Figura 6: Prueba de lanzamiento de objetos.

- **Catapulta:** La tarea consiste en lanzar munición con una catapulta a una zona determinada para derribar diferentes objetivos. En la zona de la catapulta el usuario tiene 4 objetos interactivos: una manivela para recargarla, otra manivela para rotarla, una palanca para accionarla y munición que se puede colocar en la lanzadera de la misma. Los objetivos a derribar son murallas, torres y casas pertenecientes a un castillo, los objetos casas dan 3 puntos si son golpeados por la munición pero hay murallas y torres que las protegen que proporcionan 1 y 2 puntos respectivamente y hay que derribarlas primero.



Figura 7: Prueba de catapulta.

- Armas de fuego: Se imita el disparo de un mosquete con el que hay que acertar en diferentes objetivos. Al lado del player se encontrará un conjunto de munición la cual le permite coger munición con el botón Grip del controlador y en su cintura el jarrón de pólvora. Se coloca pólvora y munición para dispararla. Los objetos dianas, distantes del jugador, no podrán ser golpeados si no es disparando la munición. Cuando dichas dianas son golpeadas por la munición o pasa un tiempo límite desaparecen. Cuantas más dianas acierte el usuario más puntos conseguirá.



Figura 8: Prueba de armas de fuego.



Figura 9: Disparo el arma.

- **Disparo con arco:** Se simula el uso de un arco para alcanzar dianas o diferentes objetivos. En la cintura del player se encontrará un carcaj de flechas que le permite cogerlas con el botón Grip del controlador. Se coge el arco con un controlador y con el otro se coloca la flecha. Los objetos dianas, distantes del jugador, solo serán golpeados lanzando las flechas con el arco. Cuando dichas dianas son golpeadas por las flechas o pasa un tiempo límite desaparecen. Cuantas más dianas acierte el usuario más puntos conseguirá.



Figura 10: Prueba de disparo con arco.



Figura 11: Disparando a un objetivo.

- Escalada: Se simula con los mandos el agarre de distintos elementos pudiendo alternar ambas manos provocando que el jugador avance. Se simula la escalada por un castillo. Si el headset del usuario colisiona con una pared u objeto, para que no pueda ver a través de los muros, la pantalla se pondrá en negro hasta que retire el headset y no haya colisión. El usuario tiene que escalar mientras coloca unas bombas, situadas en la cintura del mismo, en algunos puntos del muro. Cuando se coloca correctamente una bomba aumenta su puntuación.



Figura 12: Prueba de escalada.



Figura 13: Objetos escalables.

- Bloqueo con escudo: En esta escena nos disparan flechas y tendremos que cubrirnos con el escudo para que no nos alcancen. Se empieza con una puntuación máxima y por cada impacto recibido se disminuirá la puntuación. Un humo rojo y un sonido de corta duración de un cuerno de guerra avisará al jugador por donde viene la flecha. El escudo estará directamente asignado al controlador derecho.



Figura 14: Prueba de bloqueo con escudo.



Figura 15: Señal de ataque.

- Justa: El objetivo es simular una justa de caballeros imitando el movimiento de la lanza para alcanzar un objetivo mientras vamos montados a caballo. Cuando se alcanza aumenta la puntuación del jugador. La lanza estará directamente asignado al controlador derecho.



Figura 16: Prueba de justa.

El sonido largo de un cuerno de guerra indica que una prueba ha finalizado y se ha guardado la puntuación conseguida. Cuando finalicemos todas las pruebas con la puntuación necesaria para superarlas, en la selección de nivel, aparecerán globos significando que se ha ganado.



Figura 17: Ejemplo de globos.

Anexo III: Documentación VRTK

Como primer objetivo de este proyecto se ha realizado un estudio y documentación del paquete VRTK. Esta documentación contiene una explicación de todos los ejemplos de esta librería junto a descripción de los scripts usados. La documentación se encuentra en el siguiente repositorio: <https://drive.google.com/file/d/14BSAvpXbifWmOlFQzSbOZVHkQR8ZhCbG/view> [33].