



UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MÁSTER
MÁSTER EN INGENIERÍA INFORMÁTICA

MineRVa

Experiencia de juego en Realidad Virtual en un museo

Autor

Pedro Manuel Gómez-Portillo López

Director

Francisco Luis Gutiérrez Vela



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, 16 de julio de 2019

MineRVa

Experiencia de juego en Realidad Virtual en un museo

Autor

Pedro Manuel Gómez-Portillo López

Director

Francisco Luis Gutiérrez Vela

Pedro Manuel Gómez-Portillo López

Granada – Spain

E-mail: pedromanuel.gomezportillo@gmail.com

Web site: <https://github.com/gomezportillo/mineRVa>

© 2019 Pedro Manuel Gómez-Portillo López

Este documento y la información que contiene se proporcionan de forma confidencial, con el único fin de que el destinatario del título los lea, y no puede ser divulgado a ningún tercero ni utilizado para ningún otro propósito sin el permiso expreso por escrito del autor.

MineRVa: Experiencia de juego en Realidad Virtual en un museo

Pedro Manuel Gómez-Portillo López

Palabras clave: Realidad virtual, desarrollo de videojuegos, Unity, VRTK

Resumen

Los videojuegos permiten a sus usuarios sumergirse en mundos ficticios y meterse en la piel de personajes muy diferentes a ellos mismos en el mundo real, permitiéndoles vivir historias que de otro modo no serían capaces. Uno de los grandes cambios en el paradigma de los videojuegos está siendo la Realidad Virtual, que utilizan tecnologías inmersivas para aumentar en gran medida la sensación derealismo.

Específicamente, las tecnologías asociadas a la Realidad Virtual permiten que personas que padecen algún tipo de impedimento físico y que no pueden acceder a centros, como pueden ser museos, puedan visitarlos de una forma relativamente realista y en muchos casos incluso motivadora por la propia naturaleza de la tecnología que se emplea. Este proyecto tiene como objetivo desarrollar un sistema interactivo e inmersivo utilizando tecnologías de Realidad Virtual enmarcado en los procesos de desarrollo de un videojuego real y aplicando metodologías ágiles.

La temática de este sistema se basa en un museo en el que el jugador debe encontrar un cuadro robado ayudado por el vigilante a través de diversas salas correspondientes a las distintas etapas de la historia de arte, resolviendo pruebas temáticas en cada una.

Este proyecto explorará diversas mecánicas de interacción con el usuario y objetos virtuales, haciendo uso de físicas avanzadas e intentará integrarlas en su narrativa de la manera más orgánica posible para el jugador.

MineRVa: Virtual reality game experience in a museum

Pedro Manuel Gómez-Portillo López

Keywords: Virtual reality, game development, Unity, VRTK

Abstract

Video games make their users feel immersed in fictional worlds and live the life of characters very different from them, making them live stories that otherwise would not be able to. One of the great changes in the paradigm of video games is Virtual Reality, which uses immersive technologies to greatly increase the feeling of realism.

Specifically, the technologies associated with Virtual Reality allow people who suffer from some physical impediment and who cannot access centers, such as museums, to visit them in a relatively realistic and in many cases even motivating way, thanks to the nature of the technology that is being used. This project aims to develop an interactive and immersive environment using Virtual Reality technologies framed in the development processes of a real video game while applying agile methodologies.

The theme of this game is based on a museum in which the player must find a stolen painting helped by the guard through various rooms corresponding to the different stages of art history, solving thematic riddles in each one.

This project will explore different interaction mechanics with the user and virtual objects, making use of advanced physics while trying to integrate them in the narrative in the most organic way possible for the player.

D. Francisco Luis Gutiérrez Vela, Profesor del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***MineRVa, Experiencia de juego en Realidad Virtual en un museo***, ha sido realizado bajo su supervisión por **Pedro Manuel Gómez-Portillo López**, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 16 de julio de 2019.

El director:

Francisco Luis Gutiérrez Vela

A mi hermano

Índice general

1. Introducción	1
1.1. Motivación del proyecto	2
1.2. Motivación personal	2
1.3. Entorno	3
1.4. Objetivos	4
1.4.1. Objetivo principal	4
1.4.2. Objetivos secundarios	4
1.5. Estructura del documento	4
2. Estado del arte	7
2.1. Historia del Arte	8
2.1.1. Prehistórico	8
2.1.2. Mesopotámico y precolombino	9
2.1.3. Arte clásico	9
2.1.4. Arte medieval	10
2.1.5. Renacentismo	11
2.1.6. Barroco	12
2.1.7. Siglo XIX	12
2.1.8. Siglo XX	13
2.1.9. Aplicaciones	14
2.2. Realidad Virtual	14
2.2.1. Dispositivos	18
2.2.2. Librerías de desarrollo	22
2.2.3. Aplicaciones	26
2.3. Diseño 3D	29
2.3.1. Suites de modelado 3D	29
2.3.2. Desarrollo	32
2.4. Desarrollo de videojuegos	35
2.5. Conclusiones	36
3. Análisis inicial del problema	39
3.1. Concepto inicial	39
3.2. Narrativa	40

3.3. Mundo virtual	40
3.3.1. Estética	40
3.3.2. Descripción de las salas	41
3.3.3. Retos	42
4. Tecnologías y recursos	43
4.1. Recursos hardware	43
4.2. Recursos software	44
4.2.1. Sistema Operativo	44
4.2.2. Principales librerías	44
4.2.3. Herramientas de desarrollo	45
4.2.4. Herramientas de documentación	46
4.2.5. Herramientas de edición	46
5. Metodologías	47
5.1. Metodologías de desarrollo de videojuegos	48
5.1.1. Fase de pre-producción	49
5.1.2. Fase de producción	49
5.1.3. Fase de post-producción	49
5.2. Metodología de trabajo	50
5.2.1. Proceso de desarrollo	50
5.2.2. Metodología de desarrollo	51
5.3. Conclusiones y aplicación de las metodologías	56
6. Plan de entregas	59
6.1. Plan de entregas	59
6.2. Descripción de las entregas	63
6.2.1. Entrega 0	64
6.2.2. Entrega 1	64
6.2.3. Entrega 2	65
6.2.4. Entrega 3	65
6.2.5. Entrega 4	66
6.2.6. Entrega 5	66
6.2.7. Entrega 6	67
6.2.8. Entrega 7	67
7. Desarrollo	69
7.1. Entrega 0	70
7.1.1. Estructuración del proyecto	70
7.1.2. Primera toma de contacto con Unity	70
7.1.3. Configuración del entorno de desarrollo	71
7.2. Entrega 1	72
7.2.1. Modelado e importación	72
7.2.2. Viajar entre salas	75

7.2.3. Interacción con objetos virtuales	77
7.3. Entrega 2	78
7.3.1. Activación de una palanca virtual	79
7.3.2. Teletransportar los objetos que toquen el suelo	80
7.4. Entrega 3	81
7.4.1. Recogiendo basura virtual	82
7.4.2. Mostrando la descripción de los cuadros	84
7.4.3. Fundido a negro entre salas	86
7.4.4. Prototipado de un arco	87
7.5. Entrega 4	87
7.5.1. Limitando el peso por texturas	88
7.5.2. Disparando a los cuadros	89
7.5.3. Mostrando un candado en las puertas bloqueadas	92
7.6. Entrega 5	93
7.6.1. Niebla virtual	93
7.6.2. Cuadros con pistas y cajones interactivos	94
7.6.3. Descripción de la sala de las vanguardias	96
7.6.4. Puzzles virtuales con piezas de cuadros	97
7.7. Entrega 6	100
7.7.1. El almacén del vigilante y el cuadro robado	100
7.7.2. Sala final y desenlace	101
7.7.3. Modelo del vigilante y animaciones	103
7.7.4. Implementación de un sistema de diálogos	105
7.7.5. Menú principal del juego	108
7.7.6. Música de fondo y efectos de sonido	109
7.8. Entrega 7	111
7.8.1. Evaluación con usuarios reales	112
7.8.2. Versión final del juego y trailer	115
8. Conclusiones y Trabajo futuro	117
8.1. Conclusiones	117
8.2. Trabajo futuro	118
9. Apéndices	121
9.1. Apéndice 1: Guía de salas	121
9.2. Apéndice 2: Game Design Document	133
9.3. Apéndice 3: Evaluación de satisfacción con usuarios	148
9.4. Agradecimientos	157
Bibliografía	160

Índice de figuras

1.1.	Diagrama de entorno de <i>MineRVa</i>	3
2.1.	Mapa conceptual de <i>MineRVa</i>	8
2.2.	Zigurat de Uruk, 3200-3000 a. C.	9
2.3.	Patio de las Doncellas del Alcázar de Sevilla, siglo X	10
2.4.	Miguel Ángel Buonarroti, <i>Capilla Sixtina</i> , 1508-1512	12
2.5.	Delacroix, <i>La libertad guiando al pueblo</i> , 1830	13
2.6.	Máquina <i>Sensorama</i> , www.engadget.com	15
2.7.	<i>Sword of Damocles</i> , www.dssource.in	16
2.8.	VCASS, voicesofvr.com	17
2.9.	<i>Virtual Boy</i> , codigoespagueti.com	18
2.10.	Prototipo del <i>Oculus Rift</i> , www.theverge.com	19
2.11.	HTC Vive	19
2.12.	Gafas <i>Glasstron</i> , http://www.mellottsvrpage.com	20
2.13.	Gafas <i>Lenovo Explorer</i>	21
2.14.	Historial de precios de las <i>Lenovo Explorer</i> , pccomponentes.com	21
2.15.	Google Cardboard abiertas	22
2.16.	<i>IndexVR</i> de Valve	23
2.17.	<i>TheStoneFox</i> en Twitter sobre los motivos del cierre de VRTK	24
2.18.	Soldados del ejército americano haciendo uso de la tecnología WARSS, https://hololens.reality.news	27
2.19.	Médico simulando una operación, http://www.baboonlab.com	28
2.20.	Elven Assassin, <i>YouTube/BenPlaysVR</i>	28
2.21.	Tom Roosendaal, imagen obtenida de su cuenta de Twitter	30
2.22.	Captura de pantalla del entorno de trabajo de Blender	31
2.23.	Captura de pantalla del entorno de trabajo de Maya	32
2.24.	Captura de pantalla del entorno de trabajo de 3Ds Max	33
2.25.	Captura de pantalla de un proyecto en Unity	34
2.26.	Captura de pantalla de un proyecto en Unreal Engine	35
2.27.	Captura de pantalla de un proyecto en Godot	35
4.1.	Logo de SteamVR	44

4.2. Logo de Windows Mixed Reality	44
4.3. Logo de VRTK	45
5.1. Fases del desarrollo de videojuegos, adaptado de [Manrubia, 2014]	48
5.2. Dilbert y los roles de equipo, de https://dilbert.com	53
5.3. Flujo de trabajo de Scrum, adaptado de https://www.scrumalliance.org	53
5.4. Lista de reproducción con los vídeos de <i>MineRVa</i>	58
7.1. Resumen de la interfaz de Unity	71
7.2. Boceto de la antesala y la sala de tutorial	73
7.3. Sala 0 renderizada en Unity	74
7.4. Sala 1 renderizada en Unity	75
7.5. Script para cambiar de salas desde el inspector	76
7.6. Boceto de la segunda sala	78
7.7. Sala 2 renderizada en Unity	78
7.8. Script para controlar la palanca desde el inspector	79
7.9. Boceto de la tercera sala	82
7.10. Sala 3 renderizada en Unity	82
7.11. Script para detectar piezas de basura desde el inspector	84
7.12. Cuadro de texto superpuesto al mundo	85
7.13. Boceto de la cuarta sala	88
7.14. Sala 4 renderizada en Unity	89
7.15. Diagrama de secuencia del juego del arco	90
7.16. Boceto de la quinta sala	93
7.17. Sala 5 renderizada en Unity	94
7.18. Boceto de la sexta sala	96
7.19. Sala 6 renderizada en Unity	97
7.20. Pieza de un cuadro sobre uno de los posibles huecos	98
7.21. Sala 7 renderizada en Unity	100
7.22. Imagen del cuadro robado	101
7.23. Sala 8 renderizada en Unity	102
7.24. Algunos de los modelos que se valoraron para vigilante	103
7.25. Máquina de estados del controlador de animaciones	104
7.26. Diagrama Máquina de diálogos del gestor de diálogos	106
7.27. Menú principal del juego	109
7.28. Resultados de la evaluación con usuarios	114
7.29. Media de los resultados de la evaluación	115
8.1. Script para cambiar el idioma desde el inspector	119

Índice de cuadros

6.1.	Plan de entregas 0 y 1	60
6.2.	Plan de entregas 2, 3 y 4	61
6.3.	Plan de entregas 5 y 6	62
6.4.	Plan de entrega 7	63

Índice de listados

7.1.	Fragmento del script para viajar entre salas	76
7.2.	Fragmento del script para abrir y cerrar una puerta abatible con una palanca	79
7.3.	Fragmento del script para teletransportar un objeto si toca el suelo	80
7.4.	Fragmento del script para detectar piezas de basura	83
7.5.	Fragmento del script para activar la descripción de los cuadros	86
7.6.	Fragmento del script para actualizar un cuadro	91
7.7.	Fragmento del script para gestionar el juego del arco	91
7.8.	Fragmento del script para ignorar colisiones	95
7.9.	Fragmento del script detectar piezas de cuadros	99
7.10.	Fragmento del script del gestor de diálogos	106
7.11.	Fragmento del script para activar animaciones	107
7.12.	Fragmento del script gestionar el menú	109
7.13.	Fragmento del script para activar efectos de sonido	111

Índice de acrónimos

IDE	Integrated Development Environment
TFM	Trabajo Final de Máster
VR	Virtual Reality
VRTK	Virtual Reality Toolkit
GDD	Game Design Document
HMD	Head-mounted display
AR	Augmented Reality
MR	Mixed Reality
SDK	Software Development Kit
MIT	Massachusetts Institute of Technology
CGI	Computed-generated imagery
GUI	Graphical User Interface
UDP	Unified Development Process
RUP	Rational Unified Process
SaaS	Software as a Service

Capítulo 1

Introducción

SOLAMENTE en 2018, el **Museo Reina Sofía** de Madrid recibió 3,9 millones de visitantes, aproximadamente un 2% más que el año anterior. Del mismo modo, el **Museo de Prado** recibió casi 3 millones, un 2,4% más que en 2017. Otro ejemplo de esto es el **Macba**, el Museo de Arte Contemporáneo de Barcelona, que recibió en 2018 casi 332,000 visitantes, un 28% más que el año anterior¹. Con estos datos, puede verse que el interés por los museos está creciendo cada vez más.

Pero la industria de ocio que indiscutible más dinero genera es la **industria de videojuegos**, que solo en 2018 generó 43 mil millones de dólares, un 18% más que en 2017². Por dar un contexto de estas cifras, en 2016 la suma conjunta del cine y la música en el mundo *solo* generó 815 millones de dólares³.

Dado el éxito global de esta industria, han comenzado a nacer nuevas maneras de jugar y nuevas técnicas de interacción con los usuarios, como la **Realidad Virtual**. La Realidad Virtual es un paradigma de interacción que intenta ofrecer a sus jugadores una experiencia lo más **inmersiva** posible, para lo que hace uso de la visión estereoscópica de las personas para hacerles creer que el mundo real es lo que ven, y no solo una pantalla.

Además, los dispositivos que permiten hacer uso de estas tecnologías se están abaratando cada vez más, hasta el punto de que un jugador casual puede permitirse adquirir un headset VR para utilizarlo esporádicamente, en lugar de ser una tecnología que solo se plantena comprar aquellos que

¹https://www.abc.es/cultura/arte/abci-museos-espanoles-crecen-2018-201901021635_noticia.html

²<https://tcrn.ch/2HtFcZS>

³<https://www.elperiodico.com/es/ocio-y-cultura/20180827/videojuegos-cine-musica-cultura-comparativa-7001883>

creen que le van a dar mucho uso.

1.1. Motivación del proyecto

El principal objetivo a la hora de integrar videojuegos y tecnologías de Realidad Virtual en este proyecto es el poder crear respectivamente entornos interactivos e inmersivos que, unidos al potencial motivador que tienen de manera intrínseca los videojuegos, consigan animar a los usuarios que disfruten de *MineRVa* a visitar museos reales y ser, principalmente para aquellos que no puedan hacerlo por razones físicas, un sustituto que les permita disfrutar en la medida de lo posible de ello.

1.2. Motivación personal

Siempre he estado muy interesado en el desarrollo de videojuegos y en el proceso técnico que hay detrás. Durante el grado, que estudié en la Universidad de Castilla-La Mancha, tuve la suerte de coincidir con algunos profesores que también estaban muy interesados en este tema y que habían organizado dos cursos; el primero, en 2015, fue el **Curso de Desarrollo de videojuegos multi-plataforma para dispositivos móviles**⁴, un curso de 100 horas en el que desarrollamos un juego en 2D en Haxe utilizando el framework OpenFL.

Tras él, dos años más tarde completé el **Curso de Experto en desarrollo de videojuegos**⁵, de 750 horas, en el que realizamos varios proyectos en C++ utilizando el motor de gráficos Ogre3D y otras tecnologías como la librería de interfaces CEGUI o el motor de físicas Bullet.

A raíz de estos cursos comencé a interesarme por Blender, un programa de diseño 3D que integra todos los componentes necesarios para modelar, texturizar, animar y renderizar prácticamente cualquier proyecto.

Por un lado, nunca había trabajando con entornos de desarrollo específicos para videojuegos como Unity, y ni mucho menos con tecnologías de Realidad Virtual, lo que me pareció un reto extremadamente interesante. Por otro, aunque ya había trabajado desarrollando sistemas interactivos nunca lo había hecho aplicando metodologías adecuadas y específicas de desarrollo de videojuegos, por lo que este proyecto parecía el contexto perfecto como primera toma de contacto con ellas.

⁴<http://www.curso-openfl.cedv.quijost.com/>

⁵<http://cedv.es>

Del mismo modo, siempre me ha interesado mucho la Historia del Arte y he intentado visitar museos siempre que he tenido la oportunidad, por lo que estoy muy motivado al poder integrar ambos en este proyecto.

1.3. Entorno

El entorno de ejecución de *MineRVa* está compuesto por unas gafas de Realidad Virtual, dos controladores y un ordenador personal. Por tanto, el programa final debe ser lo suficientemente ligero y eficiente en cuanto a recursos para poder ser ejecutado en un ordenador que no sea especialmente potente.

La figura 1.1, renderizada en Blender, presenta un ejemplo de ejecución en el que se puede ver a un usuario llevando puestos el headset VR y los mandos mientras juega en uno de los escenarios iniciales.



Figura 1.1: Diagrama de entorno de *MineRVa*

Además, como en este juego no es necesario hacer movimientos rápidos y bruscos, no es necesario contar con mucho espacio libre alrededor del jugador, lo que lo hace aún más accesible al público en general, que no tendrá que disponer de una sala amplia específica para utilizar tecnologías VR.

1.4. Objetivos

1.4.1. Objetivo principal

El objetivo de *MineRVa* es crear una experiencia inmersiva en la que el jugador se sienta gratificado al ir resolviendo los pequeños acertijos que se le proponen. Con ello se pretende alcanzar dos objetivos diferentes; por un lado, se busca conseguir que nativos digitales que no suelan visitar museos encuentren en *MineRVa* una excusa para hacerlo de manera divertida, y por el otro que usuarios más mayores a los que les gusta visitarlos encuentre en la temática de este proyecto una excusa para jugarlo y disfrutar de la misma forma. En cualquiera de los dos casos se espera que, tras completar el juego, la mayoría de usuarios se interesen en visitar museos reales.

Por otro lado, uno de los nichos de jugadores más pequeños y a la vez uno en el que más se ha pensado es en aquellas personas con movilidad reducida que físicamente no pueden visitar museos. Se espera ayudar especialmente a este colectivo consiguiendo una experiencia equivalente en la medida de lo posible, además de intentar hacerla más amena y motivadora enmarcándola en el contexto de un videojuego.

1.4.2. Objetivos secundarios

Como objetivos secundarios, de naturaleza más personal, se presenta el aprender a desarrollar programas basados en tecnologías de Realidad Virtual, así como a elegir y configurar un entorno adecuado para ello.

Además, al estudiar metodologías de desarrollo de videojuegos reales se espera obtener un mejor conocimiento de cómo funciona el desarrollo de un proyecto de estas características.

1.5. Estructura del documento

Este documento ha sido estructurado siguiendo la normativa de TFM de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada a través de los siguientes capítulos.

Capítulo 2. Estado del arte

En este capítulo se realiza una presentación del estado actual de las áreas tratadas en este TFM, recopilando y exponiendo información relevante

relacionada con la Realidad Virtual o el Desarrollo de Videojuegos, entre otros.

Capítulo 3. Análisis inicial del problema

En este capítulo se introduce este proyecto y se hablará de la concepción inicial y de la narrativa de *MineRVa*.

Capítulo 4. Tecnologías y recursos

Este capítulo recoge los recursos tanto hardware como software que se han utilizado para el desarrollo y documentación de este proyecto.

Capítulo 5. Metodologías

A lo largo de este capítulo se hablará de las metodologías de desarrollo de videojuegos y se presentarán las metodologías de trabajo y desarrollo seguidas a lo largo de este proyecto.

Capítulo 6. Plan de entregas

En este capítulo se presenta y detalla el plan de entregas llevado a cabo al inicio del proyecto para poder cumplir con el plazo de tiempo de entrega.

Capítulo 7. Desarrollo

A lo largo de esta capítulo se presenta el trabajo realizado en el marco del plan de entregas del proyecto, además de los resultados obtenidos y los principales problemas encontrados.

Capítulo 8. Conclusiones y Trabajo futuro

En este capítulo se presenta el resultado final del proyecto, las conclusiones extraídas de su desarrollo y se proponen unas líneas de trabajo futuro para continuar con este proyecto.

Capítulo 2

Estado del arte

En este capítulo se presentarán y describirán las principales áreas en las que este TFM se ha basado; **Historia del Arte**, **Realidad Virtual**, **Diseño 3D** y **Desarrollo de Videojuegos**.

Uno de los retos que ha supuesto este proyecto ha sido el de estudiar y comparar las diferentes alternativas que se presentaban a la hora de resolver un determinado problema; por ello, en los casos en los que esto aplique se presentarán las diferentes opciones disponibles que han sido estudiadas.

La descripción de estado del arte de cada una de las áreas en las que se ha basado este trabajo comenzará con un recorrido general a lo largo de su historia, hablando de sus contribuciones y posibilidades para terminar presentando su estado actual. Por ello, el estudio de cada una de las áreas mencionadas se dividirá y se detallará a través de sus subáreas, lo que permitirá un estudio más profundo y más detallado de la misma.

La sección de la **Historia del Arte** comenzará exponiendo un breve recorrido por sus diferentes etapas de manera crónica. Además, se explicará el origen del nombre de este TFM.

A continuación, en la sección de la **Realidad Virtual** se hablará de su historia, las tecnologías y recursos software disponibles y de las interfaces con las que pueden interactuar los usuarios de esta tecnología.

La siguiente sección hablará del **Diseño 3D**, de qué programas y suites existen actualmente para modelar tridimensionalmente y qué IDEs hay disponibles para desarrollar e implementar sistemas interactivos.

Por último, se hablará del **Desarrollo de Videojuegos** y de sus aplicaciones en este proyecto desde un punto de vista teórico.

La figura 2.1 ofrece un resumen de las diferentes tecnologías que colaboran en *MineRVA* y de las relaciones que hay entre ellas.

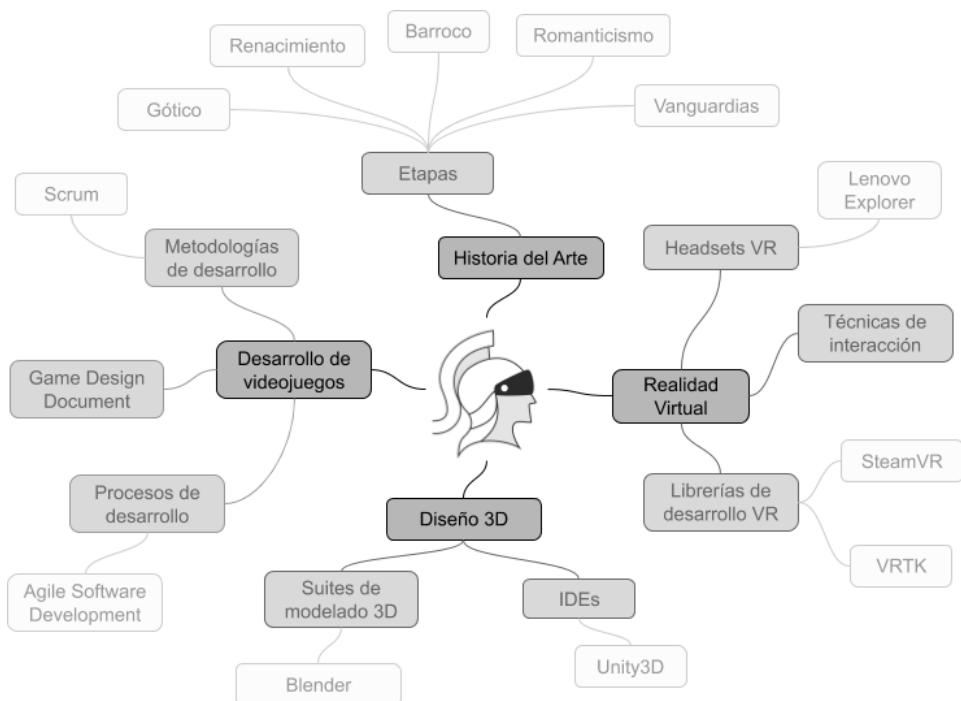


Figura 2.1: Mapa conceptual de *MineRVA*

2.1. Historia del Arte

Para entender el contexto artístico de este TFM primero se presentará un breve resumen a través de las etapas del Arte [Ramírez, 2010].

2.1.1. Prehistórico

Habrá que remontarse a la etapa prehistórica más antigua, el **Paleolítico**, para encontrar las primeras muestras de arte. Los primeros homínidos plasmaron su realidad en las paredes de las cuevas y abrigos que habitaron, legándonos lo que hoy conocemos con el nombre de **pinturas rupestres**. Sirviéndose de pigmentos naturales, grasa animal, carbón y sangre, representaron grupos humanos danzantes y escenas de cacerías con un sentido mágico y sagrado, al existir la creencia de que favorecían la fertilidad de las mujeres de la tribu o fomentaban la caza. El mismo cometido tenían

las toscas estatuillas realizadas en piedra y hueso, que con el tiempo fueron sustituidas por otras más refinadas. Por tanto, lo que hoy catalogamos como producciones artísticas a las que atribuimos un valor estético, no fueron ideadas como tal, sino que cumplían una función concreta como medio para obtener ganancias materiales.

2.1.2. Mesopotámico y precolombino

Será en la antigua región de **Mesopotamia** donde encontraremos el origen de la civilización en los términos en los que hoy la entendemos y un arte más acorde con la concepción actual. Las numerosas culturas que se sucedieron en este reducido punto geográfico levantaron una arquitectura pensada y medida, digna de grandes ingenieros capaces de proyectar edificios de gran envergadura, sus conocidos *zigurats*, figura 2.2 (antecedentes de las pirámides egipcias), por medio de sistemas de arcos y bóvedas. Las artes plásticas no se quedaron atrás. Hoy en día se conservan restos escultóricos y relieves de gran calidad técnica, que han marcado una etapa tan florida como misteriosa de nuestra historia.

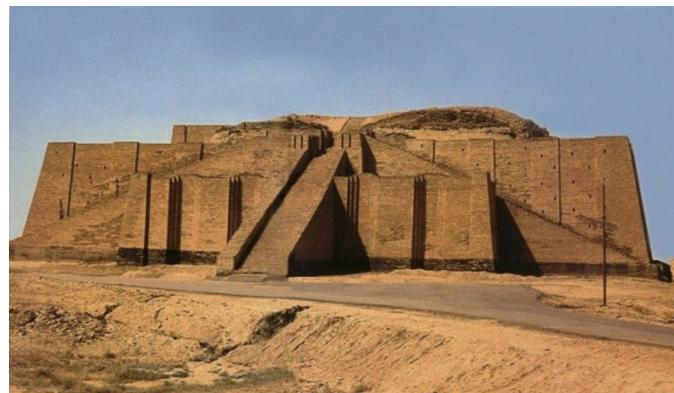


Figura 2.2: *Zigurat de Uruk*, 3200-3000 a. C.

Paralelamente, cabe destacar la importancia de las manifestaciones artísticas que están teniendo lugar en el continente americano. Los templos **incas** y **mayas** son solo un ejemplo del ingenio y la habilidad de aquellas civilizaciones.

2.1.3. Arte clásico

Los **griegos** abrieron paso a un nuevo tipo de arte basado en la medida humana: la arquitectura está hecha por y para el hombre, y tanto la escultura

como la pintura están protagonizadas por la figura humana, en la que reinan los ideales de armonía y proporción.

Los **romanos**, muy influenciados por los anteriores, erigieron una arquitectura grandilocuente a la vez que funcional, pensada para el disfrute de los grandes hombres de la época y la exaltación de su poder, igual que los arcos de triunfo y los monumentos conmemorativos. La escultura, heredera de la griega, se vuelve más realista y poderosa.

El nombre de este TFM viene de la **deidad romana Minerva**, que es la **diosa del arte**, la sabiduría y la guerra, además de la protectora de Roma y la patrona de los artesanos. El poeta Ovidio se refería a Minerva como *la diosa de las mil obras* [Lomba, 2013]. Su equivalente en la mitología griega es Atenea.

2.1.4. Arte medieval

El **arte islámico** se extendió por Próximo Oriente y avanzó hasta la Península Ibérica, donde los musulmanes permanecieron durante ocho siglos (711-1492). En función de sus zonas de dominio emprendieron un arte muy dispar, centrando sus esfuerzos en la arquitectura especialmente. Fue característico de sus construcciones el uso del arco apuntado, el lobulado y sus variantes, y el arco de herradura, este último sobre todo en la Península dada la influencia visigoda en nuestro territorio. Las yeserías de entramados vegetales, las tracerías, la eboraria y la incorporación del agua y de la naturaleza en la obra artística son otros rasgos distintivos del arte islámico. Ejemplo de ello es el Alcázar de Sevilla, que puede verse en la figura 2.3.



Figura 2.3: Patio de las Doncellas del Alcázar de Sevilla, siglo X

La llegada de la **Edad Media** en Europa después de la caída del Imperio Romano de Occidente y la irrupción del cristianismo como nueva religión

dominante tuvieron una repercusión directa en el arte generado a partir del siglo V en Europa. La cristiandad encontrará en el **arte paleocristiano** su primera manifestación, en un principio clandestina, a través de los sarcófagos y pinturas de catacumbas y, tras la legalización de la religión, de manera pública a través de las grandes basílicas.

Pero fue en el **románico**, primer estilo internacional de la cultura occidental, donde el cristianismo se manifestó de manera total a partir del siglo X. La arquitectura se caracterizó, en general, por la sencillez estructural, basándose en gran medida en la producida por los romanos. La profusión decorativa se concentró en espacios muy concretos de los edificios, como las grandes portadas de las iglesias, adornadas con conjuntos escultóricos centrados en la vida de Jesús, la Virgen María y los Santos patrones, algunos de gran complejidad iconográfica. La escultura, al igual que la pintura, se caracterizó por la sencillez de las formas y el hieratismo de gestos y actitudes: el naturalismo no era una prioridad para los promotores de las obras, más interesados en la expresión espiritual de las figuras.

Las formas simples del románico se fueron complicando progresivamente hasta culminar en un nuevo estilo, el **gótico**, caracterizado por la verticalidad, la luminosidad y la mayor riqueza ornamental. Los avances constructivos posibilitaron la edificación de templos de mayor altura y vanos cada vez más amplios que se cubrieron con vidrieras. En cuanto a las artes plásticas, prevalece la costumbre de colmar las portadas de acceso con importantes ciclos iconográficos esculpidos. La escultura se volvió más naturalista en estas fechas, rompiendo con la rigidez románica aunque manteniendo el halo de espiritualidad. La pintura recubrió los muros de las iglesias y se traspasó también a la tabla y a la miniatura, que iluminó los libros litúrgicos.

2.1.5. Renacimiento

Con el Renacimiento se volvió la mirada a la época clásica, recuperando el arte a la medida del hombre. Los artistas comenzaron a ser reconocidos como trabajadores intelectuales y dejaron de ser considerados como simples artesanos. Fue el momento de grandes arquitectos como Brunelleschi y Alberti, que erigieron grandes templos cristianos inspirándose en las magníficas obras romanas; escultores de la talla de Donatello, que ejecutó obras realistas con buenas dosis de idealización a la manera clásica; y pintores como Botticelli y Piero della Francesca, introductor de la perspectiva. Otros artistas pasaron a la historia como grandes hombres del Renacimiento polifacéticos e inclasificables: Leonardo da Vinci, Rafael o Miguel Ángel Buonarroti, cuya obra más famosa es la Capilla Sixtina (figura 2.4).



Figura 2.4: Miguel Ángel Buonarroti, *Capilla Sixtina*, 1508-1512

2.1.6. Barroco

La última etapa renacentista del Manierismo fue revelando lo que ocurriría con las artes a partir del siglo XVII. El Barroco fue la etapa de eclosión de una arquitectura grandilocuente y teatral que cumplió con una doble función: por un lado, la de manifestar el poder de los comitentes de la obra, que utilizaban la arquitectura como auténtica propaganda política; y por otro lado, en la arquitectura religiosa, la de adoctrinar a unos fieles fácilmente impresionables. Las iglesias se recubrieron de pinturas murales, que ocuparon también las bóvedas a modo de rompimientos celestes, y de ciclos escultóricos que representaron la vida de Cristo y de los Santos, sirviendo sus figuras como modelos de vida. La pintura encontró en esta época su momento más destacado hasta la fecha en cuanto a técnica y revaloración social. Pintores de la talla de Caravaggio en Italia o de Velázquez en España introdujeron una nueva manera de plasmar el mundo a través de unas composiciones cada vez más complejas y la técnica del claroscuro.

2.1.7. Siglo XIX

En sus últimos momentos, el Barroco adquiere una profusión decorativa y un nivel de recargamiento inusitados en la Historia de los estilos. Los arquitectos del siglo XIX vinieron a romper con esta grandilocuencia y condujeron a la arquitectura hacia el academicismo y la mesura. No obstante, no se puede hablar de homogeneidad estilística en toda la centuria. La arquitectura, si se caracterizó por algo, fue por el eclecticismo y la recuperación de tendencias pasadas a través de los historicismos. Así, volvemos a encontrar cresterías en el Neogótico, grandes órdenes arquitectónicos en el Neoclásico o rasgos de inspiración hispanomusulmana en el Neomudéjar, entre otros.

Lo mismo ocurrió con la pintura. La mayor libertad en el sistema artístico permitió que los artistas elaboraran unos estilos cada vez más personales, dando lugar a unas producciones muy dispares. Románticos, realistas, impresionistas, simbolistas, y un largo etcétera hicieron de este siglo uno de los más prolíficos y originales hasta el momento.



Figura 2.5: Delacroix, *La libertad guiando al pueblo*, 1830

2.1.8. Siglo XX

Los movimientos de vanguardia vinieron a transformar profundamente y sin retorno a la totalidad de las artes. La arquitectura dejó atrás las formas tradicionales de los historicismos y empezó a hacer uso de los nuevos materiales y tecnologías, además de aumentar su preocupación por el entorno circundante. Aunque tampoco podemos hablar de homogeneidad total, el racionalismo fue la tendencia predominante y la que más impronta dejó en el panorama arquitectónico del siglo XX. Autores como Le Corbusier y Wright levantaron unos edificios ante todo funcionales sirviéndose de líneas puras, formas geométricas y módulos regulares.

En plástica, las vanguardias se manifestaron en lo que ha pasado a la historia como *ismos*. Incontables tendencias de gran originalidad y complejidad teórica coincidieron temporal y espacialmente. Una de las primeras fue el Fauvismo, y su técnica, la más temperamental hasta la fecha, dando todo el protagonismo al color usado de manera emotiva; el Expresionismo, subjetivo y visceral; el Cubismo, rompedor en su visión simultánea de la realidad; el Futurismo y su violencia teórica y expresiva, acorde con el momento histórico del que es fruto; el Surrealismo y su dimensión onírica; y la abstracción en todas sus vertientes, entre otros.

Todas estas corrientes abrieron el camino a un arte definitivamente plural e inclasificable, además de nuevos géneros como la instalación, la *performance* y el arte sonoro, y un sistema de mercado que se ha convertido en el verdadero dueño del devenir artístico.

2.1.9. Aplicaciones

La Historia del Arte ha sido una rama de estudio que siempre ha generado mucho interés, y prueba de ello son las innumerables aplicaciones destinadas a mostrarla de manera didáctica y divulgativa.

En lo que refiere a páginas web existen muchos portales, como *SmartHistory*¹ o *The Museum With No Frontiers*², que proporcionan de manera gratuita y desinteresada toda clase de contenidos y recursos de temática artística.

También, y motivado por la amplia disponibilidad de dispositivos móviles entre la población actualmente, existen diversidad de aplicaciones móvil con este mismo fin, como son *Google Arts & Culture*³ que permite a sus usuarios visitar virtualmente exposiciones de arte, o *DailyArt*⁴ para Android, que cada día ofrece la imagen y descripción de una obra de arte, ambas también gratuitas.

Y por último, en lo que refiere a videojuegos, se han desarrollado multitud de ellos con temática artística y de museos. Un ejemplo de ello, y en lo que refiere a aventuras gráficas, en 1992 fue lanzado *The Dagger of Amon Ra*, en el que el jugador debe resolver un asesinato en un museo egipcio. Hay muchos más y de muchos otros géneros, como *Discover Babylon*, *Time Explorer* o *Versailles – A game of intrigue at the Court of Louis XIV*.

2.2. Realidad Virtual

Podríamos decir que, en resumen, la Realidad Virtual es una herramienta capaz de crear entornos creíbles, inmersivo e incluso multisensoriales para sus usuarios, de manera que sientan que están en un sitio en el que realmente no están; es decir, se encuentran en un **mundo virtual**.

Antes de empezar a hablar de la Realidad Virtual, sin embargo, es im-

¹<https://smarthistory.org/>

²<http://www.museumwnf.org/>

³<https://play.google.com/store/apps/details?id=com.google.android.apps.cultural>

⁴<https://play.google.com/store/apps/details?id=com.moiseum.dailyart2>

portante diferenciar los conceptos Realidad Virtual, Realidad Aumentada y Realidad Mixta. Mientras que el primero intenta sumergir por completo al usuario en un mundo virtual, la **Realidad Aumentada** propone complementar el entorno real superponiendo sobre él objetos digitales. Por último, la **Realidad Mixta** une ambos conceptos para permitir interactuar con objetos reales dentro de un mundo virtual, estar completamente inmerso en un mundo virtual o reproducir elementos virtuales en un entorno real.

Los orígenes del concepto de Realidad Virtual no están claros del todo ya que, aplicando la definición anterior, en la época del Renacimiento y con el desarrollo de las técnicas artísticas de perspectiva los propios pintores eran capaces de crear mundos realistas que no existían [Schnipper, 2017].

Sin embargo, las primeras referencias más modernas a este concepto vienen de la mano de un autor de cine. En los años 50 el cineasta **Morton Heilig** propuso un *experiencia de teatro* con la que poder engañar a los sentidos del público para transportarles a otro momento y lugar. Pocos años después, en 1962, construyó un prototipo llamado **Sensorama**, un artilugio capaz de despertar varios de los sentidos de sus usuarios (figura 2.6). Este artilugio era capaz de generar en sus usuarios la experiencia de montar en moto en las calles de Brooklin sintiendo el viento en la cara, la vibración del asiento de la motocicleta, una vista tridimensional y los olores de la ciudad. Sin embargo, debido a lo caro que era, no pudo continuarse con su producción [Brockwell, 2016].



Figura 2.6: Máquina *Sensorama*, www.engadget.com

Tres años después, en 1965, el ingeniero **Ivan Sutherland** escribió el

artículo *A head-mounted three dimensional display* [Sutherland, 1965], o por sus siglas HMD, en el que describe a las imágenes bidimensionales como poco inmersivas y propone un casco con una pequeña pantalla en cada ojo que hacen uso de la **visión estereoscópica** del ser humano.

Tres años después de la publicación de este artículo, en 1968, Iván Sutherland y su estudiante Bob Sproull desarrollaron el que se considera primer dispositivo para aplicaciones inmersivas. Dado su abultado tamaño, como puede verse en la figura 2.7, finalmente se le acabó llamando ***Sword of Damocles***. Además, este dispositivo contaba con rastreador posicional del casco, por lo que la posición de las imágenes que se visualizaban a través de él correspondía con su posicionamiento virtual. Pese a todo, y como aún no se disponía de la suficiente tecnología, los gráficos eran muy básicos [López, 2018].



Figura 2.7: *Sword of Damocles*, www.dsoucre.in

A lo largo de los años setenta, se siguieron desarrollando varios prototipos como *Grope*, de la Universidad de Carolina del Norte, o *Videoplace*, que detectaba mediante cámaras las siluetas de las sombras de los usuarios y las proyectaba sobre una pantalla, lo que les permitía interactuar con ellas [Mazuryk and Gervautz, 1999].

Los años 80 fueron muy prolíficos para los simuladores de Realidad Virtual y, paralelamente, comenzaron las primeras discusiones entre diversos filósofos tecnológicos, como Jaron Lanier, Myron Krueger o Ted Nelson, por la búsqueda del término que definiera una tecnología aún en proceso de nacer [López, 2018].

En 1982, Thomas Furnes desarrolló el simulador más avanzado del momento, el *Visually Coupled Airborne Systems Simulator*, o VCASS, que estaba contenido en su totalidad dentro del casco [Cadena, 2008]. Aún así seguía

siendo bastante voluminoso, como puede verse en la figura 2.8



Figura 2.8: VCASS, voicesofvr.com

Finalmente, el término *Realidad Virtual* fue acuñado por Jaron Lanier, fundador de una de las empresas VPL Research, una de las primeras en comercializar estos sistemas a gran escala [Fernández et al., 2002]. Lanier proponía unos dispositivos que, acoplados en nuestro cuerpo, fueran capaz de hacernos percibir a través de los sentidos sensaciones realistas de un mundo diferente al nuestro, de tal manera que cuanto más avanzados sean estos dispositivos más realistas serán las sensaciones y más difícil será diferenciar entre este mundo y el real [López, 2018].

Y poco a poco, principalmente movido por la creciente industria del videojuego y sus múltiples usos potenciales, estas tecnologías han seguido desarrollándose. Una de las primeras aplicaciones comerciales en los videojuegos fue la *Virtual Boy* de Nintendo. Fue lanzada al mercado en 1995 y contaba con dos pequeñas pantallas monocromáticas que reproducían juegos. Este dispositivo puede verse en la figura 2.9.

Aunque esta consola fue un fracaso (apenas se vendieron 770.000 unidades en todo el mundo⁵) demostró que se podían desarrollar dispositivos similares.

Y con esto llegamos a la actualidad, donde poco a poco, y principalmente por el abaratamiento de esta tecnología, el mercado de dispositivos de Realidad Virtual está creciendo y cada vez hay más dispositivos al alcance de los usuarios.

⁵<http://www.gamepro.com/gamepro/domestic/games/features/111823.shtml>, 2007



Figura 2.9: *Virtual Boy*, codigoespagueti.com

2.2.1. Dispositivos

Actualmente, los principales dispositivos que permiten jugar en VR destinados a usuarios normales y que, por tanto, son en los que se ha centrado este TFM, son los siguientes.

Oculus Rift

En el año 2012, Palmer Luckey fundó la compañía Oculus VR con la idea de crear un dispositivo destinado a jugar en VR y con un precio viable para los jugadores casuales. A finales de ese mismo año, y con ayuda de John Carmack (programador en juegos como Doom, Quake o Wolfenstein), mostró el primer prototipo del dispositivo y fue un éxito, por lo que lanzó una campaña en Kickstarter que superó con creces el dinero necesario, consiguiendo casi 2 millones y medio de dólares⁶ y que puede verse en la figura 2.10.

Año tras año fue mejorando sus prototipos hasta que en marzo de 2014 Facebook compró esta compañía por un total de 2,300 millones de dólares. Finalmente, en mayo de 2015, se anunció la versión finalizada de las gafas lista para su comercialización. Actualmente pueden comprarse por unos 240€.

⁶<https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game>



Figura 2.10: Prototipo del *Oculus Rift*, www.theverge.com

HTC Vive

Vive son unas gafas de realidad virtual co-fabricadas por HTC y Valve. En 2014 se mostraron los primeros prototipos de un sistema de realidad virtual producido por la empresa Valve, pero no fue hasta marzo de 2015 cuando HTC reveló oficialmente las Vive. Antes de la versión para consumidores se fabricaron las Vive PRE, que fueron gratuitas para algunos desarrolladores de videojuegos con el fin de promover sus aplicaciones.



Figura 2.11: HTC Vive

A principios de 2016 se anunció su lanzamiento con un precio de salida de 900€, aunque actualmente pueden adquirirse por la mitad, unos 450€.

PlayStation VR

Este headset no es la primera propuesta de PlayStation de VR. Su primer intento comercial, *Glasstron*, se lanzó en 1997 para el juego *MechWarrior*

2. Estas gafas permitían a jugadores adoptar una perspectiva visual desde dentro de la cabina de la nave, como si la controlaran.



Figura 2.12: Gafas *Glasstron*, <http://www.mellottsvrpage.com>

A mediados de 2014, el desarrollador de PlayStation Anton Mikhailov anunció en una conferencia que su equipo había estado trabajando en el Project Morpheus durante 3 años, rediseñando el *PlayStation Move* para adaptarlo a la tecnología VR. Al año siguiente este proyecto se bautizó como *PlayStation VR*. Finalmente lanzada al mercado en 2016, actualmente puede comprarse por unos 250€, aunque solo funciona con un sistema PlayStation 4.

Lenovo Explorer

La compañía Lenovo también ha dado el salto a los headsets VR desarrollando uno propio, llamado Explorer. Este dispositivo, lanzado al mercado en octubre de 2017, hace uso del *Microsoft Mixed Reality*, un framework de desarrollo implementado por Microsoft para su headset AR, las *Microsoft HoloLens*. Como su nombre sugiere, este framework está pensado para trabajar tanto con tecnologías VR como AR.

A diferencia del resto de headsets, el Explorer tiene dos pequeñas cámaras integrados en la parte delantera, lo que le permite escanear y transmitir la ubicación del usuario, de los mandos y del entorno haciendo uso de la tecnología de rastreo sin balizas de Microsoft, similares a las que se usan para HoloLens. Además incluye dos controladores, como puede verse en la figura 2.13.



Figura 2.13: Gafas *Lenovo Explorer*

Además, tienen el precio más competitivo de todas, llegando a haber podido ser compradas en la página pccomponentes.com por 150€, como puede verse en la figura 2.14, aunque actualmente rondan los 200€.



Figura 2.14: Historial de precios de las Lenovo Explorer, pccomponentes.com

Por todo esto se cree que son las más probables para ser adquiridas en los jugadores interesados en esta tecnología, por lo que han sido elegidas para usarse en este proyecto. En el capítulo 4 se hablará en más profundidad de su tecnología y sus especificaciones técnicas.

Google Cardboard

La opción más económica para comenzar en el mundo de la VR son los headsets de cartón. El más famoso es el de Google, que permite usar un teléfono móvil como pantalla para abaratizar mucho los costes, de manera que el headset es simplemente una caja de cartón vacía con dos pequeñas lentes

para los ojos, como puede verse en la figura 2.15, y que puede comprarse en Amazon por menos de 10€.



Figura 2.15: Google Cardboard abiertas

El principal problema es que los móviles, al no ser dispositivos especialmente potentes, no permiten el renderizado de escenarios complejos, por lo que no consiguen una sensación de inmersión muy buena.

Index VR

Anunciado por Valve en mayo de este año y aún no disponible para comprar, Index VR⁷ es el último headset incluido en la lista. Es un dispositivo para jugar en VR compuesto por un headset, dos controladores y dos estaciones de posicionamiento (figura 2.16) que, según la propia Valve, permitirán un *tracking* mucho más preciso de la posición y los movimientos de los jugadores, que podrán moverse libremente en un área de 10x10 metros.

Por otro lado, es de lejos el dispositivo destinado al público general más caro del mercado, estando disponible para reserva el pack con los tres periféricos anteriormente presentados por 1080€.

2.2.2. Librerías de desarrollo

Como la Realidad Virtual es una tecnología relativamente nueva que aún no se ha abaratado todo lo que puede, no ha conseguido llegar a los hogares de la mayoría de sus potenciales usuarios. Por tanto, como la demanda es muy baja, la oferta de aplicaciones también lo es y, en consecuencia, hay muy pocas empresas que hayan invertido en desarrollar frameworks y librerías de desarrollo para aplicaciones que hagan uso de la Realidad Virtual.

⁷<https://www.valvesoftware.com/es/index>



Figura 2.16: *IndexVR* de Valve

Aún así, han aparecido algunas librerías desarrolladas por empresas con la mayor cuota de mercado y que pueden permitirse innovar, como Steam, y otras que se han centrado en esta tecnología y lo necesitaban, como Oculus. A continuación se presentan las más importantes.

VRTK

Virtual Reality Toolkit⁸ es un framework de desarrollo multi-librería desarrollado en solitario bajo licencia MIT por *TheStoneFox*⁹, o Harvey Ball, un hombre residente en Birmingham, Reino Unido. Este framework solo está disponible para el IDE Unity3D y permite desarrollar paralelamente para distintas librerías a través de una interfaz común; es decir, podemos desarrollar para SteamVR y Oculus a la vez usando la interfaz de VRTK.

Pero el desarrollo de esta librería no ha sido fácil para *TheStoneFox*; todo empezó en abril de 2016 cuando, sin apenas experiencia en desarrollo de videojuegos, comenzó un proyecto con Unity3D y SteamVR y se dio cuenta de lo confuso y poco documentado que estaba todo, por lo que decidió desarrollar su propio SDK o, en sus propias palabras¹⁰,

I wanted to build something for it as I was new to game dev. I'd been using Unity for about a month just as a hobby, I tried to use the SteamVR Unity plugin and found it confusing, realized a lot of people found it confusing and started VRTK as a way to help people get into developing for VR."

⁸<https://vrtoolkit.readme.io>

⁹<https://github.com/thestonefox>

¹⁰<https://uploadvr.com/vrtk-stone-fox-unity-tool/>

Y poco a poco fue avanzando en su proyecto y ganando la atención de los desarrolladores hasta que decidió abrirse una cuenta de Patreon¹¹, donde llegó a ingresar 2000\$ al mes, y lo que comenzó como un pasatiempo a tiempo parcial se convirtió en su único proyecto. Además, comenzó a realizar conferencias por todo el mundo gracias a las que fue ganando popularidad, como la *Unite Europe 2017*, cuya ponencia puede verse en YouTube¹².

Pero a finales de 2017, entre el nulo apoyo que recibía de la comunidad, que hacía que él fuera literalmente el único desarrollador del proyecto (como puede verse en la figura 2.17), y que no quería que la cuenta de Patreon diera la impresión de que el proyecto era solvente, porque no lo era, decidió cerrar esta cuenta y dar por finalizado el proyecto. Aún así, aunque dijo que no añadiría funcionalidad nueva, se comprometió a seguir manteniéndolo durante algún tiempo, principalmente arreglando bugs¹³.

TonyVT SkarredGhost @SkarredGhost · Dec 31, 2017

Replies to @humptycalderon @VR_Toolkit

Thanks to you for sharing! Theoretically the project is **#opensource**, so everyone could continue to develop it, but practically, without the lead developer supporting it, it's very hard

VRTK @VR_Toolkit

It's been open source from the beginning, people could have helped develop it up until this point, but as it's all been left to one person to do a majority of the work then I don't see many people helping out now.

2 8:21 PM - Dec 31, 2017

See VRTK's other Tweets

Figura 2.17: *TheStoneFox* en Twitter sobre los motivos del cierre de VRTK

Por suerte, cuatro meses después y gracias a una donación por parte de Oculus, el proyecto volvió a la vida. El propio *TheStoneFox* aclaraba en Twitter que no era ningún tipo de compra, ya que VRTK seguiría siendo gratuito y mantendría la misma filosofía que antes de cerrarse¹⁴.

A día de hoy la página de Patreon vuelve a estar abierta, aunque apenas ingresa un tercio de la cantidad de antes de dar por finalizado el proyecto.

¹¹<https://www.patreon.com/vrtk>

¹²<https://youtu.be/AbIiBrg8yT4>

¹³<https://skarredghost.com/2017/12/28/vrtk-shutting-will-make-vr-development-unity-easier-now/>

¹⁴<https://skarredghost.com/2018/04/30/vrtk-announces-version-4-thanks-to-an-oculus-grant/>

Actualmente existen varios proyectos comercializados desarrollados gracias a este framework, como *Deism*, *One of the last* u *Oddescape*. Uno de los más famosos es *quiVr*¹⁵, un *tower defense* en el que el jugador encarna a un arquero que debe acabar con todas las oleadas de enemigos que aparecen antes de que lleguen a su base.

TheStoneFox también realiza pequeños vídeos enseñando todos los juegos que utilizan su framework con el fin de darles publicidad y de mostrar la potencia que tiene. Todos estos vídeos pueden verse en una lista de reproducción¹⁶ en la que los añade.

SteamVR

Steam es una plataforma de distribución digital de videojuegos cuya dueña es Valve. Steam lanzó en 2014 una nueva iniciativa para adaptar la Realidad Virtual a los juegos digitales, llamada SteamVR¹⁷, que permite el desarrollo y uso de juegos de realidad virtual a partir de su plataforma, es decir, solo los juegos distribuidos en Steam podrán acceder a estas librerías.

A principios de 2014, Valve inició un proyecto para VR para su plataforma. En junio de 2014 se mostraron los primeros avances pero no fue hasta la Mobile World Congress de 2015 cuando se anunció públicamente. El primer dispositivo que usó esta tecnología fue el HTC Vive, el headset VR de HTC.

Actualmente la plataforma cuenta con más de 1200 experiencias VR con todo tipo de juegos y simuladores. Además, desde 2017 tienen en fases de desarrollo la incorporación de la Realidad Aumentada en colaboración con Microsoft¹⁸.

Windows Mixed Reality

Windows Mixed Reality, anteriormente Windows Holographic, es una plataforma de MR desarrollada por Microsoft e incluida de manera nativa en Windows 10 desde principios de 2018.

El principal dispositivo para este framework son las Microsoft HoloLens, un headset que en realidad es un ordenador inalámbrico autónomo corriendo Windows 10. Utiliza sensores avanzados, dos pantallas estereoscópicas de alta definición y sonido espacial que le permiten ejecutar aplicaciones de realidad aumentada.

¹⁵<http://quivr.net/>

¹⁶<http://yt.vu/p/PLTiD-q2AfVNlrAN5w0WK6h0yjy6zL1tWv>

¹⁷<https://steamcommunity.com/steamvr>

¹⁸<https://generacionxbox.com/microsoft-steamvr-realidad-aumentada/>

Las HoloLens estuvieron desarrollándose durante casi cinco años antes de su lanzamiento público en 2015, aunque inicialmente fue concebido no como un headset independiente sino como una tecnología para la cámara Kinect¹⁹.

Inicialmente, las HoloLenses únicamente estaban disponibles en Norteamérica, pero gracias a la positiva respuesta que han recibido de parte de sus usuarios llegarán próximamente a Europa.

Además, con el fin de atraer a más usuarios, Microsoft ha lanzado **Windows Mixed Reality for SteamVR**, un wrapper que permite jugar a juegos desarrollados con la librería SteamVR con un headset que implemente su tecnología. Es necesario descargar este software desde Steam como si fuera un juego.

Oculus SDK

El framework de desarrollo para Oculus, también llamado Oculus SDK²⁰, es un conjunto de librerías desarrolladas por Oculus para su headset VR, Oculus Rift.

Este SDK está disponible para los IDEs Unity3D y Unreal Engine, además de poder descargarse para trabajar directamente en Windows.

2.2.3. Aplicaciones

Los usos de la tecnología VR son casi tan variados como los colectivos la utilizan. En esta sección se presentan las principales aplicaciones de la VR.

Simulación

Una de las aplicaciones más potentes de la VR es la simulación. Por su componente inmersivo, la VR es perfecta para este tipo de aplicaciones, ya que permite a sus usuarios sentir que están en un entorno diferente sin los peligros que pueda conllevar.

Dentro de la simulación, uno de los campos de aplicación más importantes es el militar, donde poder entrenar a personas para situaciones de riesgo sin poner en peligro sus vidas es de vital importancia.

¹⁹<https://www.wired.com/2015/01/microsoft-hands-on/>

²⁰<https://developer.oculus.com/downloads/>

Un ejemplo de esto es el Departamento de Defensa de los Estados Unidos de América, que ha desarrollado el *Weapons Augmented Reality Scoring System*, o WARSS, un entorno en el que los soldados pueden practicar situaciones y tácticas a las que podrían enfrentarse un día usando tecnologías VR y MR, concretamente con dispositivo HoloLens de Microsoft, como puede verse en la figura 2.18.



Figura 2.18: Soldados del ejército americano haciendo uso de la tecnología WARSS, <https://hololens.reality.news>

Otro de los campos en los que triunfa esta tecnología por los mismos motivos es en la medicina, donde es imprescindible que los médicos sepan cómo manejar situaciones delicadas para la vida de los pacientes antes de tener que enfrentarse a ellas por primera vez ya que, gracias a la ayuda de programadores y modeladores 3D, se puede reproducir con exactitud la anatomía de los pacientes y practicar cirujías antes incluso de llevarlas a cabo, reduciendo los costes de utilización de cadáveres para pruebas y permitiendo repetir la operación cuantas veces fuera necesario.

En figura 2.19 puede verse a un cirujano con el headset VR Oculus Rift haciendo uso de esta tecnología.

Videojuegos

Como ya se ha expuesto antes, la industria de los videojuegos es actualmente una de las más lucrativas y, como los propios videojuegos requieren de la inmersión de sus usuarios, ha sido fácil para la VR abrirse paso a través de ella.

Actualmente, los juegos en VR que más éxito tienen en Steam²¹ son aquellos en los que el jugador debe disparar, ya sea con armas, como en

²¹<https://store.steampowered.com/vr/>



Figura 2.19: Médico simulando una operación,
<http://www.baboonlab.com>

el caso de *Pavlov VR* o *Blade and Sorcery*, o con arcos, como en *Elven Assassin*, como puede verse en la figura 2.20.



Figura 2.20: Elven Assassin, *YouTube/BenPlaysVR*

Reconstrucción

Dada la naturaleza inmersiva de la VR, es perfecta para, mediante sistemas de modelado tridimensionales, recrear lugares tal y como eran en algún momento de la historia para poder estudiarlos o verlos fácilmente.

Un ejemplo de esto sería las aplicaciones VR²² que han surgido a raíz de la cercana trajeada que sufrió la iglesia de Notre-Dame de París en la que

²²<https://uploadvr.com/notre-dame-vr/>

un incendio provocó que la mayor parte de su interior se quemara. Estas aplicaciones permiten ver cómo era antes del incendio de un modo mucho más realista e inmersivo que verlo a través de un vídeo.

2.3. Diseño 3D

El diseño 3D es esencial en el desarrollo de *MineRVa*, ya que tanto a la hora de diseñar y modelar las salas como implementar sus funcionalidades será necesario basarse en este área.

En esta sección se presentarán y describirán las dos áreas del diseño 3D más importantes para desarrollo de este proyecto.

2.3.1. Suites de modelado 3D

Lo principal a la hora de trabajar con modelos 3D es poder diseñarlos y modelarlos. Para ello, actualmente existen multitud de programas y suites completas a disposición de los usuarios. Los más importantes y con la mayor cuota de mercado son los siguientes.

Blender

Blender²³ es un programa informático multiplataforma, dedicado especialmente al modelado, iluminación, animación y renderizado de gráficos tridimensionales, además de incluir herramientas para esculpir tridimensionalmente y hasta un motor de juegos propio, el Blender Game Engine. Aunque inicialmente era distribuido de forma gratuita pero sin el código fuente, actualmente es *open source* bajo la licencia GNU General Public License, o GPL, y está disponible para prácticamente todos los sistemas operativos para ordenador.

Todo empezó a finales de los años 80 cuando Ton Roosendaal, el futuro fundador de Blender, era el líder de NeoGeo, el mayor estudio de animación de Holanda. Por aquel entonces la herramienta 3D que utilizaban era muy lenta y cara de mantener, por lo que decidieron reescribirla de 0, plantando la semilla de lo que después sería el software de creación que ahora conocemos como Blender²⁴.

Pero mientras que NeoGeo continuaba trabajando en Blender, Roosendaal pensó que el mundo entero podría beneficiarse de su herramienta, por

²³<https://www.blender.org/>

²⁴<https://www.blender.org/foundation/history/>



Figura 2.21: Tom Roosendaal, imagen obtenida de su cuenta de Twitter

lo que a finales de los 90 fundó una división de NeoGeo, Not a Number (o NaN a partir de ahora) para distribuir gratuitamente Blender y fomentar su desarrollo y su mercado, ya que la mayoría de los programas comerciales de modelado de la época costaban miles de dólares.

A principios del año 2000, NaN recaudó más de 4,5 millones de euros procedente de sus inversores, y pronto pasó a tener más de 50 empleados trabajando alrededor del mundo intentando mejorar y promocionar Blender, publicando la versión 2.0 unos meses después.

Desafortunadamente, debido a las decepcionantes ventas y al continuo clima de dificultades económicas, los inversores decidieron dar por terminado el desarrollo de Blender. Pero por suerte Roosendaal no permitió que Blender desapareciera, así que en marzo de 2002 fundó la organización no lucrativa Blender Foundation, cuyo primer objetivo fue continuar con el desarrollo como un proyecto de código abierto basado en la comunidad de usuarios.

La compañía tuvo que obtener 100,000€ para poder comprar los derechos del código fuente y los de propiedad intelectual de Blender a su antigua compañía, NaN, que consiguieron ser reunidos por la comunidad en menos de 7 semanas. Finalmente, el 13 de octubre de 2002, Blender fue liberado al mundo bajo los términos de GNU General Public License.

Actualmente, el desarrollo de Blender sigue gracias a un equipo de voluntarios procedentes de diversas partes del mundo, liderados por Ton Roosendaal, que continúan actualizándolo gracias a las donaciones en su página web, obteniendo actualmente 36,521€ al mes²⁵. Además, periódicamente realizan cortos en los que muestran la potencia de su herramienta como *Big Buck Bunny*, *Caminandes*, *Sintel* o *Agent 327*.

²⁵<https://fund.blender.org/>

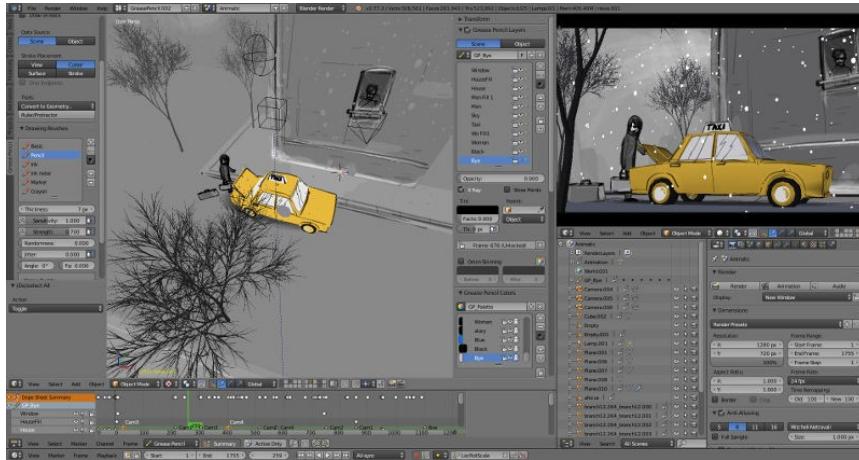


Figura 2.22: Captura de pantalla del entorno de trabajo de Blender

Autodesk Maya

Maya es un programa informático dedicado al desarrollo de gráficos 3D por ordenador, efectos especiales y animación. Además, es el único software de 3D acreditado con un Oscar gracias al enorme impacto que ha tenido en la industria cinematográfica como herramienta de efectos visuales.

Maya surgió tras la fusión de Alias y Wavefront, dos empresas canadienses dedicadas al CGI los gráficos generados por ordenador, ya que ambas se utilizaban de manera conjunta prácticamente siempre. Este programa se desarrolló en estrecha colaboración con Disney durante la producción de Dinosaurio a finales de la década de los 1990. Una de los requisitos de Disney era que la interfaz gráfica de usuario pudiera ser fácilmente personalizable, lo que ayudó a que años después Maya se convirtiera en el estándar de la industria gracias a que cada compañía podía cambiar la GUI a su gusto.

Autodesk Maya se comercializaba en dos versiones; *Maya Complete* es la versión básica y *Maya Unlimited* la avanzada, que contiene a su versión más simple además de añadir toda clase de módulos con funcionalidad para trabajar con pelo, tejidos, fluidos....

Autodesk 3Ds Max

Anteriormente 3D Studio Max, Autodesk 3Ds Max es un programa de creación de gráficos y animación 3D desarrollado por Autodesk, que lo compró a su anterior dueño por 182 millones de dólares. Actualmente es uno de los programas de animación 3D más utilizado, especialmente para el

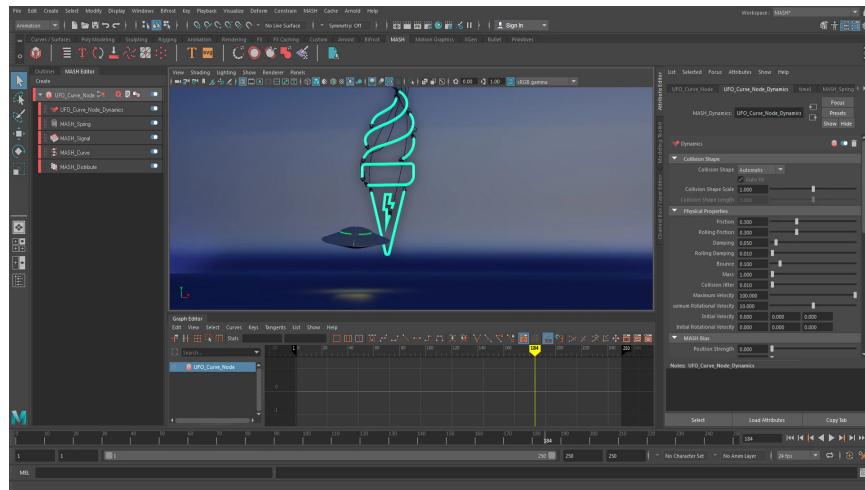


Figura 2.23: Captura de pantalla del entorno de trabajo de Maya

desarrollo de videojuegos, anuncios de televisión y arquitectura.

Todo empieza a mediados de los años 1980, con la creación del Grupo Yost. Este grupo estaba compuesto por varios expertos del sector que comenzaron a trabajar paralelamente en soluciones software que les permitieran trabajar con gráficos 3D. A finales de los 1980 decidieron unirlos en el paquete *The Cyber Studio*, al que poco a poco fueron añadiendo funcionalidades. Este programa no pasó desapercibido para la compañía Autodesk, dueña de AutoCAD, y al año siguiente contrataron a los principales desarrolladores de *The Cyber Studio* para que trabajasen para ellos en su propio programa.

A principios del 1990 salió al mercado el programa 3D Studio, aunque no fue hasta 1996 cuando salió al mercado el 3D Studio Max, totalmente terminada y diseñada especialmente para Windows. Actualmente, en su versión 9 su nombre oficial Autodesk 3Ds Max.

2.3.2. Desarrollo

Actualmente existen varios entornos de desarrollo integrado, o IDE por sus siglas en inglés, destinados a facilitar el desarrollo de videojuegos. En esta sección se presentan los más importantes.

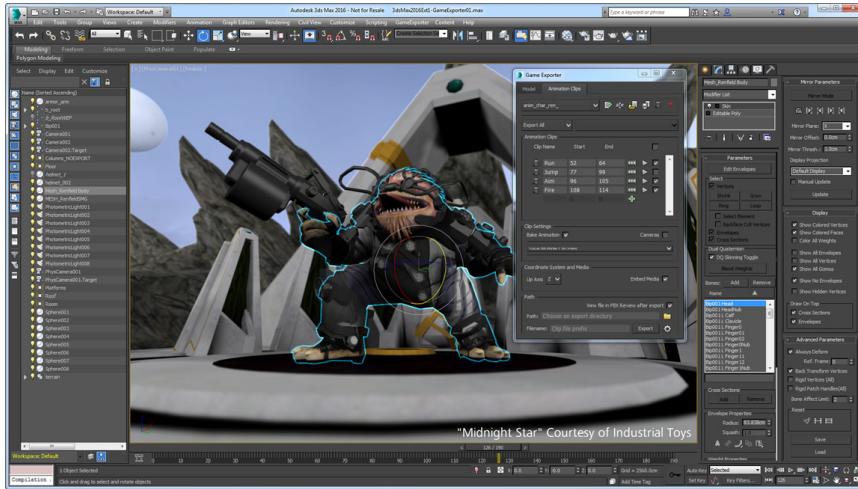


Figura 2.24: Captura de pantalla del entorno de trabajo de 3Ds Max

Unity3D

Unity es un motor de videojuego multiplataforma creado por Unity Technologies disponible para los principales sistemas operativos, y permite compilar para varias plataformas cómodamente.

La compañía Unity Technologies fue fundada en 2004 en Dinamarca, la cual desarrolló Unity para su primer juego, *GooBall*, que fue un fracaso. Aún así, los integrantes de la compañía reconocieron el valor del motor de juegos que había hecho y decidieron continuar con su desarrollo hasta lanzar oficialmente Unity un año después, en 2005.

A día de hoy existen dos versiones; Unity Personal, una versión gratis de Unity para principiantes que no incluye servicio al cliente, capacitación ni servicios adicionales, y Unity Pro, que sí los incluye, costando 125\$ al mes. Además, los desarrolladores de un juego ingrese más de 100,000\$ al año deberán suscribirse a la versión Pro obligatoriamente. En total, más de cinco millones de personas han utilizado Unity en 2017²⁶.

Unreal Engine

Unreal Engine es un motor de juego creado por la compañía Epic Games que al estar escrito en C++ presenta un alto grado de portabilidad, por lo que es una de las herramientas más utilizadas por los desarrolladores de juegos.

²⁶<https://unity3d.com/public-relations>

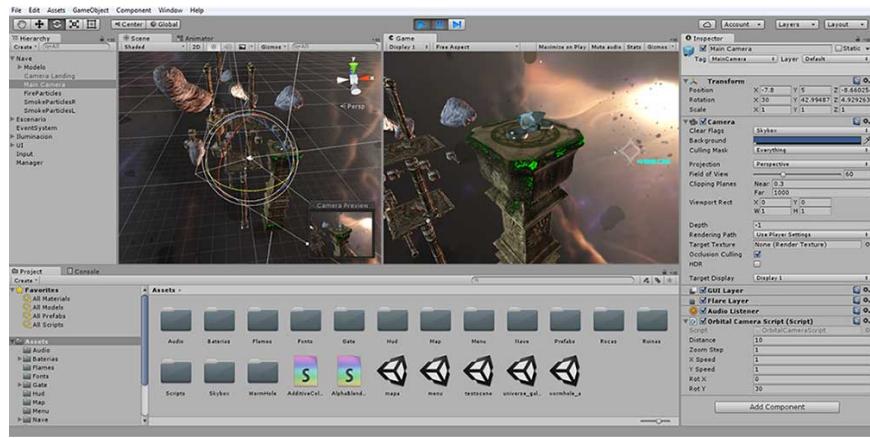


Figura 2.25: Captura de pantalla de un proyecto en Unity

Este motor fue mostrado inicialmente en el juego de disparos en primera persona *Unreal* en 1998, que incluía detección de colisiones, IA y herramientas para trabajar con juegos en red. El motor se reescribió y cuatro años después se lanzó su segunda versión, que incluía físicas para vehículos y sistemas de partícula, además de soporte para PlayStation 2 y XBox.

Poco a poco, y gracias al éxito de este IDE, a lo largo de los años ha ido aumentando sus características y añadiendo soporte para las tarjetas gráficas y videoconsolas que han ido apareciendo.

Desde marzo de 2015 Unreal Engine 4 está disponible públicamente de forma gratuita aunque, si se decide comercializar el juego, la desarrolladora deberá aportar el 5% de los ingresos que obtenga a Epic²⁷.

Godot

Godot es un motor de videojuegos 2D y 3D *open source* publicado bajo la licencia MIT desarrollado por su comunidad y disponible para los principales sistemas operativos.

Godot fue desarrollado y utilizado privativamente por la empresa argentina OKAM Studios desde principios del 2001, aunque en febrero de 2014 su código fuente fue liberado y publicado en GitHub²⁸. Es el principal motor de videojuegos *open source*, y su comunidad ayuda activamente a desarrollarlo.

²⁷<https://www.unrealengine.com>

²⁸<https://github.com/godotengine/godot>

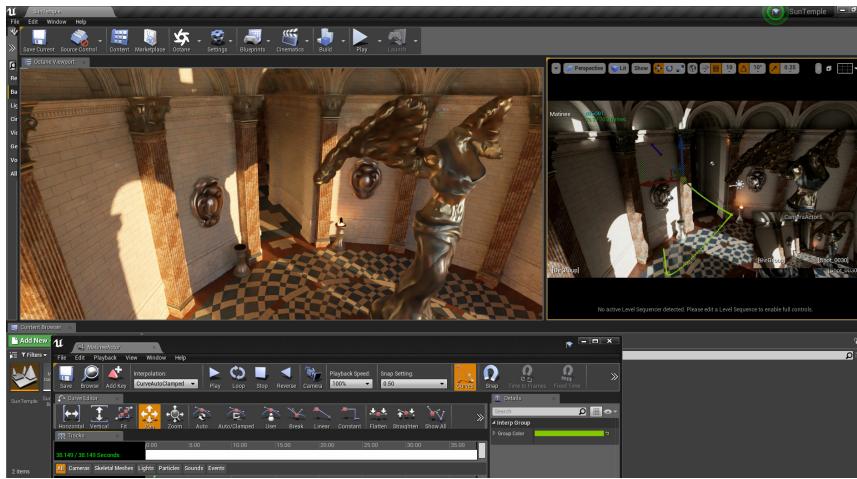


Figura 2.26: Captura de pantalla de un proyecto en Unreal Engine

Además, actualmente tiene una página en Patreon²⁹ con más de 1,000 mecenas y obtiene más de 10,000\$ al mes.

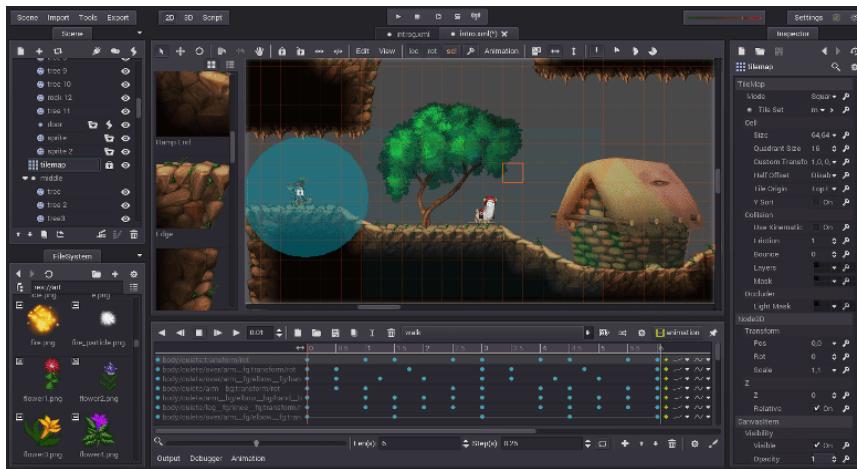


Figura 2.27: Captura de pantalla de un proyecto en Godot

2.4. Desarrollo de videojuegos

El desarrollo de videojuegos es el proceso de creación de un videojuego desde su concepción inicial hasta su versión final. Es una actividad multidis-

²⁹<https://www.patreon.com/godotengine>

ciplinar que involucra profesionales de distintos campos, como programadores, diseñadores gráficos, animadores o compositores de música, entre otras áreas.

A principios de la década de 1960 comenzaron a desarrollarse los primeros videojuegos, pero no estaban disponibles para el público general ya que éste no poseía dispositivos para ejecutarlos dado su elevado precio. No fue hasta la década de 1970 y la llegada de la primera generación de consolas de videojuegos y ordenadores domésticos, como el Apple I, cuando empezaron a desarrollarse videojuegos a nivel comercial. Además, como los ordenadores tenían muy poca potencia y los juegos eran muy simples, un único programador podía hacerse cargo del juego completo en no más de 6 meses [nex, 1997].

Sin embargo, la creciente potencia de procesamiento de los ordenadores, los estándares de la industria y las expectativas del jugador han hecho que esto deje de ser la norma y solo pase en casos aislados. El precio promedio de la producción de un videojuego lentamente aumentó de 1-4 millones de dólares en 2000 a más de 5 millones en 2005, y luego a más de 20 millones en 2010³⁰.

Uno de los documentos más importantes, si no el más, a la hora de desarrollar un videojuego es el **Game Design Document**, o por sus siglas en inglés GDD a partir de ahora. Este es un documento creado y editado por el equipo de desarrollo del juego y contiene toda la descripción del videojuego y se utiliza para organizar esfuerzos dentro de un equipo de desarrollo. Este documento no es el guión del juego, sino una síntesis que propone lo que va a ser para que todo el mundo tenga la misma idea en mente [Bethke, 2003]. Es equivalente a la **Biblia** de una serie de televisión.

Como podemos ver, el desarrollo de un videojuego ha ido adquiriendo un carácter mucho más formal, por lo que se necesita coordinar a muchos profesionales teniendo en cuenta salarios y tiempos para que el producto final pueda generarse correctamente. En el capítulo 5 se discutirá en mayor profundidad las actuales metodologías de desarrollo que existen y cómo se ha adaptado este proyecto a ellas.

2.5. Conclusiones

A lo largo de este capítulo se han presentado las principales áreas y tecnologías en las que se ha basado este proyecto y que deberán trabajar conjuntamente y hacer sinergia para su correcta completitud.

³⁰https://en.wikipedia.org/wiki/Video_game_development

El papel de la Historia del Arte es principal en este proyecto; tanto su temática como su narrativa están estrechamente a este área, y la estética de cada una de las salas se verá altamente influenciada por la etapa artística que representa.

Por otro lado, el componente inmersivo de la Realidad Virtual es de esencial importancia, y se utilizará junto a la naturaleza motivadora de los videojuegos para hacer que todos los usuarios se sientan totalmente inmersos en el mundo virtual que les ofrece *MineRVa*. Este mundo será modelado completamente a mano, por lo que la utilización de técnicas y tecnologías de diseño 3D es de vital importancia.

Capítulo 3

Análisis inicial del problema

En este capítulo se presenta el nacimiento y la idea inicial de la que parte este proyecto, así como su narrativa y la descripción del mundo virtual en el que tiene lugar.

3.1. Concepto inicial

Lo primero que se hizo antes de empezar a trabajar en el proyecto fue proponer una narrativa en la que basar el diseño y desarrollo; por ello, el primer paso que se dio en el proyecto fue redactar el documento **Guía de salas**, que puede verse en el apéndice 9.1. En este documento se redactó la primera versión de la narrativa del juego además de una propuesta con las salas que habría en el museo y qué obras de arte y puzzles tendría cada una. Este documento fue enviado al tutor de este TFM y, tras él dar el visto bueno al mismo, se comenzó a trabajar en el proyecto.

Además, como se está trabajando siguiendo una metodología enmarcada en el desarrollo de videojuegos, como se explica en profundidad en el capítulo 5, el siguiente documento en el que se empezó a trabajar y que sirvió de andamio para el proyecto fue el **Game Design Document**, o GDD. En un desarrollo real, este documento contendría la información necesaria para que todos sus integrantes tuvieran clara la idea global del juego, y en este caso es lo mismo; el documento contiene, entre otra información, el resumen, los objetivos o la narrativa del juego, y puede consultarse en el anexo 9.2.

En resumen, *MineRVa* es un juego con una historia de un jugador desarrollado en un museo compuesto por varias habitaciones ambientadas en las principales épocas históricas con diferentes pruebas que el jugador tiene que superar para avanzar.

3.2. Narrativa

La narrativa del proyecto ha intentado ser enmarcada en un museo creíble y realista, de tal manera que se ha colocado al jugador en la piel de un detective infalible que ha sido contratado por el museo para resolver el misterio de la desaparición de su mayor obra de arte, y todo apunta a que ha sido robado. Este museo es el más grande del mundo y tiene obras de arte de todos los rincones, desde cuadros a estatuas.

El encargado de recibir al jugador en el museo es su vigilante, un vetusto y campechano señor que ha pasado toda su vida trabajando en él. Este personaje será el encargado de introducir al jugador la historia y las mecánicas del juego, de explicarle su misión y de guiarle a través de las distintas salas, ayudándole para que no se quede atascado.

Pero cuál es la sorpresa del jugador cuando, tras completar todas las pruebas y llegar al final del museo, descubre el misterioso ladrón de guante blanco es en realidad el propio vigilante, quien ha robado el cuadro para su disfrute personal.

Como puede verse, es necesario el vigilante disponga de un método para poder comunicar información al jugador, y lo que más encaja con la naturaleza de este proyecto son **diálogos unidireccionales**, ya que el jugador no podría hablar con él. El vigilante utilizará estos diálogos para explicar al usuario su misión en cada sala, le informará de algunos de sus errores y le felicitará cuando realice correctamente las tareas más importantes.

3.3. Mundo virtual

A continuación se describirán la estética y la composición del mundo virtual de este proyecto.

3.3.1. Estética

Como el presupuesto de este proyecto es nulo y el tiempo disponible para completarlo es especialmente bajo, la estética deberá ser acorde a estos factores. Por ello, se ha elegido una **estética low poly**, con lo que los tiempos de modelado se verán muy reducidos. Aún así, se intentará que no sea especialmente plana con la ayuda de texturas medianamente realistas que ayuden a los jugadores a conseguir una mayor inmersión y a dar mayor personalidad a las salas, pero todas ellas deberán ser gratuitas.

Por otro lado, aunque el modelado de las salas deberá ser hecho manual-

mente, se podrán importar de internet modelos tridimensionales especialmente complejos, como las estatuas que se crean necesarias para las salas; estos modelos suelen ser digitalizaciones exactas, que se adaptarán para reducir el número de polígonos, reducir el peso total del juego y hacer que no sea necesario un equipo hardware potente para jugarlo.

3.3.2. Descripción de las salas

Cuando se habla de salas en este proyecto se refiere a las unidades que presentan una historia, una temática y un puzzle distinto a las demás. El museo estará compuesto por 7 salas, aparte de la de tutorial, y cada una de ellas hará referencia a un período artístico distinto y estará ambientada decorada de manera coherente con él.

Aunque inicialmente se planteó hacer bifurcaciones en los caminos que llevan a las salas de manera que el jugador pudiera elegir o incluso que tuviera que ir para atrás para acceder a alguna tras desbloquearla, finalmente se ha decidido hacer un museo lineal, principalmente porque se espera que sea más fácil para los jugadores, y porque tiene más sentido en el contexto de la ambientación temporal de las salas que vayan una después de otra.

El objetivo de cada nivel o sala será resolver el puzzle propuesto para acceder a la siguiente, además de que el usuario interactúe con las obras de arte y aprenda sobre ellas, ya que en alguno de los casos le será necesario.

Como se ha indicado antes, en total habrá **8 salas**. La primera a la que accede el jugador será la del **tutorial**, que tendrá un puzzle especialmente fácil para que el jugador se pueda centrar en entender el concepto, las mecánicas y los controles del juego.

Tras ella el jugador accederá a la sala corresponderá al período **gótico-flamenco**, el primero desde el que se ha decidido trabajar y estará ambientado en el interior de una iglesia gótica, con paredes de piedra altas y bóvedas de crucería. Además, tendrá tres cuadros; a los lados, *El Matrimonio Arnolfini* (1434) de Jan van Eyck y *El descendimiento* de van der Weyden (1436), a modo representativo del género, y en la pared del fondo *El jardín de las delicias* de El Bosco (alrededor del 1500), que se abrirá por la mitad para mostrar la puerta al siguiente nivel.

La siguiente sala estará ambientada en el **renacimiento**, y tendrá ventanas amplias y frisos de mármol. Los cuadros de esta sala serán *La última cena* de Leonardo da Vinci (alrededor de 1495), *La escuela de atenas* de Rafael (1511) y *La creación de Adán*, de Miguel Ángel Buonarroti (1511). Para ambientar la sala también se utilizarán esculturas 3D, como *Piedad* o *Moisés*, ambas de Miguel Ángel, ya modeladas y con licencias de uso libre.

A continuación se encontrará la sala del **barroco**, con altas paredes de piedra. El concepto de esta sala ha cambiado varias veces a lo largo del desarrollo del proyecto, y por cuadros finalmente se han utilizado cuatro cuadros de San Sebastián, que fue un santo al que asetearon hasta la muerte y del que se hicieron muchas versiones a lo largo de este período. Los pintores elegidos han sido Guido Reni (1625), José de Ribera (1636), Mattia Preti (1657) y Peter Paul Rubens (1614).

La siguiente sala corresponderá al **romanticismo**, por lo que se habrá poca iluminación y se utilizarán texturas de madera y un sistema de partículas de niebla para crear inmersividad. Los cuadros elegidos han sido *Fusilamiento de Torrijos y sus compañeros en las playas de Málaga* de Antonio Gisbert Pérez (1887), *Doña Juana la Loca* de Francisco Pradilla y Ortiz (1877) y *El caminante sobre un mar de nubes* de Caspar David Friedrich (1818).

La última sala estará ambientada en las **vanguardias del siglo XX**. Aunque en este período coexisten multitud de estilos distintos, se han elegido el **cubismo**, el **expresionismo**, el **surrealismo** y el **postimpresionismo** como los más representativos, por lo que los cuadros utilizados para esta sala son, respectivamente, el *Guernica* de Pablo Picasso (1937), *El grito* de Edvard Munch (1910), *La tentación de San Antonio* de Salvador Dalí (1946) y *La noche estrellada* de Vincent van Gogh (1889).

Cuando el jugador complete todas las salas podrá acceder a la **sala final**, en la que descubrirá que el vigilante que creía su amigo es en realidad el ladrón de guante blanco al que ha estado persiguiendo desde el principio.

3.3.3. Retos

Como ha podido verse, las salas que componen el museo son muy variadas entre sí, y los acertijos en ellas también. Para evitar que las salas sean monótonas y repetitivas, cada una de ellas tendrá un reto totalmente diferente y hecho a medida que el jugador tendrá que superar para acceder a la siguiente sala.

Todos estas pruebas estarán enmarcadas dentro de diversas dinámicas de Realidad Virtual, donde el jugador deberá ser proactivo e investigar qué debe hacer y cómo hacerlo. Por ejemplo, deberá observar los cuadros para encontrar mensajes ocultos, interactuar con ellos y con otros elementos del entorno o incluso hacer que los propios cuadros cambien para poder avanzar.

Capítulo 4

Tecnologías y recursos

En esta sección se presentarán y detallarán los recursos hardware, lenguajes de programación, librerías y programas utilizados a lo largo del desarrollo de *MineRVa*.

4.1. Recursos hardware

Este proyecto necesita de periféricos especiales para trabajar con tecnologías VR, por lo que los recursos hardware de este proyecto son más exigentes.

- **Lenovo Explorer.** Heatset VR que se utilizará a lo largo del desarrollo. Incluye dos controladores, que son los que el usuario utilizará como interfaz al mundo virtual.
- **Asus R510V.** Ordenador en el que se comenzó la práctica, y que por motivos técnicos hubo de cambiar a mitad del proyecto. Cuenta con un procesador Intel i7-6700HQ, una memoria RAM DDR4 de 8GB, un disco duro HyperX Kingston SSD de 256GB y una tarjeta gráfica NVIDIA GeForce GTX 950M.
- **Lenovo Legion Y530.** Es el segundo ordenador con el que se trabajó y con el que se terminó el proyecto. Cuenta con un procesador Intel i5-8300H, una memoria RAM DDR4 de 8GB, un disco duro HyperX Kingston SSD de 256GB y una tarjeta gráfica NVIDIA GeForce GTX 1050.

4.2. Recursos software

Al haber trabajado con distintas tecnologías, los recursos software de este proyecto son variados, y se presentan a continuación.

4.2.1. Sistema Operativo

Se ha elegido trabajar con **Windows 10**, ya que es el que proporciona mejor soporte para los entornos y librerías de desarrollo, además de ser el que se espera que tengan la mayoría de usuarios de *MineRVA*. Concretamente, se ha utilizado la versión **Education**, ya que se cuenta con licencia de la misma por ser estudiante.

4.2.2. Principales librerías

- **SteamVR**¹ Es la principal librería para la que desarrollará, ya que es tanto la que mejor soporte tiene como la que más usuarios se espera que tengan o que más fácil sea para ellos descargar, ya que se encarga de ello Steam automáticamente.



Figura 4.1: Logo de SteamVR

- **Windows Mixed Reality for SteamVR**². Es el wrapper necesario para utilizar el SDK de Steam con las gafas elegidas, ya que utilizan Windows Mixed Reality funcionar, que es una plataforma desarrollada por Microsoft para trabajar con MR. Del mismo modo que SteamVR, se descarga automáticamente a través de Steam.



Figura 4.2: Logo de Windows Mixed Reality

¹<http://steamvr.com>

²<https://store.steampowered.com/app/719950/>

- **VRTK**³⁴. Es un framework de desarrollo multi-librería desarrollado en solitario por *TheStoneFox* que permite desarrollar y compilar paralelamente para distintas librerías a través de una interfaz común, lo que le ha convertido en uno de los más utilizados del mercado.



Figura 4.3: Logo de VRTK

4.2.3. Herramientas de desarrollo

- **Unity**⁵. (v2019.1.2) Es el IDE elegido para desarrollar este proyecto. El lenguaje de desarrollo para trabajar con él es C#, un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft en el año 2000 que forma parte de su plataforma .NET.
- **Blender**⁶. (v2.79b) es un programa informático multi plataforma, dedicado especialmente al modelado, iluminación, animación y renderizado de gráficos tridimensionales que se utilizará para modelar las salas del museo.
- **Visual Studio Community**⁷. (v2019) Es un IDE desarrollado por Microsoft disponible para los principales sistemas operativos y que soporta una gran variedad de lenguajes. Es el editor por defecto que recomienda Unity, además de sincronizarse con él en tiempo real, por lo que será el utilizado para programar en C#.
- **Atom**⁸. Desarrollado por GitHub, es un IDE de código abierto con soporte para extensiones desarrolladas por la comunidad e integración nativa con git, lo que lo hace especialmente potente para trabajar en todo tipo de proyectos. Se utilizará este programa para editar el resto de los archivos.
- **Git**⁹. Como herramienta de control de versiones se ha utilizado git, diseñado inicialmente por Linus Torvalds, el padre de Linux. Además,

³<https://vrtoolkit.readme.io/>

⁴<https://vrtoolkit.readme.io>

⁵<https://unity.com/es>

⁶<https://blender.org>

⁷<https://visualstudio.com/vs/community/>

⁸<https://atom.io>

⁹<https://git-scm.com/>

los repositorios tanto del proyecto como de esta documentación estarán alojados en **GitHub**.

4.2.4. Herramientas de documentación

- **LATEX.** Sistema de composición de texto que se ha utilizado para la edición de la documentación de este TFM en PDF. Se ha utilizado desde el SaaS Overleaf.
- **LibreOffice Writer.** (v5.5) Editor de documentos *open source* que forma parte del paquete LibreOffice y que se ha utilizado para generar el resto de documentos, como la guía de salas o el GDD.
- **LibreOffice Draw.** (v5.5) Editor utilizado para generar diagramas y gráficos. También forma parte del paquete LibreOffice.
- **LibreOffice Calc.** (v5.5) Editor de hojas de cálculo utilizado para generar gráficas a partir de datos.
- **Inkscape.** (v0.92) Editor de imágenes utilizado para trabajar con los gráficos vectoriales del proyecto.

4.2.5. Herramientas de edición

- **GIMP.** (v2.10.8) Editor de imágenes *open source*, multiplataforma y muy potente, aunque solo puede trabajar con gráficos rasterizados. Sus siglas significan *GNU Image Manipulation Program*.
- **Audacity.** (v2.3.2) Programa multiplataforma para la grabación y edición de audio. Es el editor de audio y sonido más difundido en las distribuciones Linux y se ha utilizado para editar los efectos de sonido del juego.
- **Kdenlive.** (v19.04.2) Acrónimo de *Non-Linear Video Editor* es un editor de vídeo multiplataforma que soporta los principales formatos y estándares. Se ha utilizado para editar los vídeos generados.

Capítulo 5

Metodologías

HUBO un tiempo en el que desarrollar un juego no necesitaba mucho más de un equipo pequeño de personas (3 ó 4 como máximo) y no mucho más de 6 meses de esfuerzo [Dille and Zuur, 2007], pero la exacerbada competencia que existe actualmente en la industria implica equipos de desarrollo de más de 50 personas expertas trabajando durante varios años con un presupuesto de millones de dólares para lograr publicar un solo videojuego comercial [Morales et al., 2010].

Por tanto, dada la evidente complejidad tanto técnica como personal de un videojuego, es necesario manejar su desarrollo haciendo uso de las metodologías y técnicas de planificación adecuadas que involucren una serie de pasos organizados para guiar cada proceso hasta el resultado final; es decir, hacer un buen uso de la **Ingeniería del Software**. Según describe el ingeniero americano Barry Boehm en [Boehm, 1979], un artículo de referencia en este campo, la Ingeniería del Software es *the practical application of scientific knowledge to the design and the construction of computer programs and the associated documentation required to develop, operate and maintain them.*

Tras leer esta definición es fácil darse cuenta de lo fundamental que es llevar a cabo un diseño e implementación adecuados. En este capítulo se hablará de la metodología de desarrollo de videojuegos más común y a continuación se presenta y justificará la metodología de trabajo seguida a lo largo del desarrollo de *MineRVa*.

5.1. Metodologías de desarrollo de videojuegos

Los videojuegos no dejan de ser un recurso informático, aunque están muy próximos en cuanto a consumo y producción al entorno audiovisual. El sistema de producción audiovisual tiene clara relación con gran parte de las fases de desarrollo del videojuego pero, a pesar de ser considerado un software más, no existe una metodología común y propia para su diseño y desarrollo [Manrubia, 2014]. El campo del videojuego ya no es sólo una industria que aporta importantes beneficios sino que se convierte en objeto de estudio en lo que a metodologías se refiere [Aguado et al., 2008].

El desarrollo de un juego, a lo largo de su ciclo de vida, se puede asemejar al de una película de cine, pudiéndose segmentar en tres fases ampliamente diferenciadas: **pre-producción, producción y post-producción**, cada una con sus etapas características [Bethke, 2003].

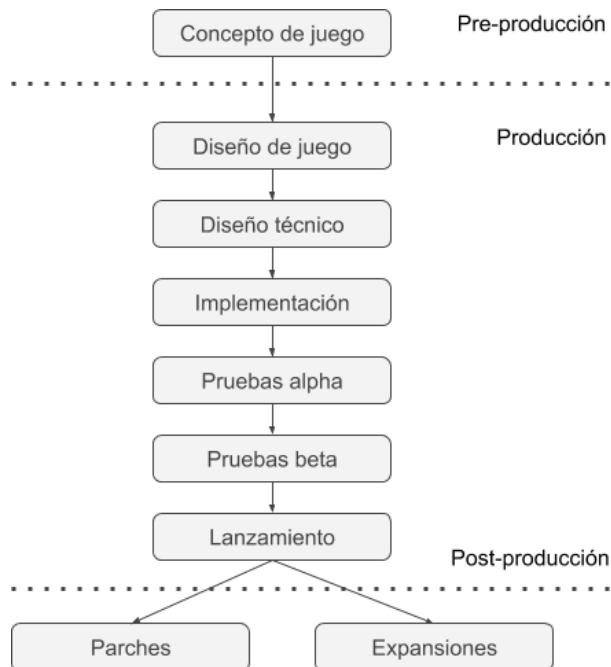


Figura 5.1: Fases del desarrollo de videojuegos, adaptado de [Manrubia, 2014]

Esta metodología puede ser usada con distintos procesos de desarrollo, como *Rational Unified Process*, *Scrum* o *Huddle*. Este último es un proceso específico de desarrollo de videojuegos y se llama así siguiendo la analogía de Scrum, ya que la reunión que se realiza en el fútbol americano antes de

cada jugada se llama *huddle*. Su filosofía es que mediante breves reuniones se planifique cada iteración, permitiendo tanto realizar un seguimiento más específico al avance del proyecto como poder hacer correcciones tempranas ante posibles desviaciones [Morales et al., 2010].

Aunque el problema de proyectos privativos como videojuegos a gran escala es que no suele filtrarse mucha información referente a su parte técnica, sí se sabe que uno de los procesos más populares ha sido el de Cascada, aunque poco a poco las compañías han ido migrando a metodologías de desarrollo ágiles, más flexibles [Morales et al., 2010].

A continuación se explican estas fases con más profundidad.

5.1.1. Fase de pre-producción

En la primera fase del proyecto se define el concepto de juego que estamos buscando así como sus aspectos más relevantes, como su **género**, su **historia** y sus principales **bocetos** para obtener una primera idea de su estilo [Rouse III, 2005]. Además, en esta fase se comienza a dar forma al GDD, del que ya se habló en el capítulo 3.

5.1.2. Fase de producción

Esta es la fase más exigente del proyecto. En ella participan multitud de profesionales muy especializados y confluyen toda clase de actividades, como el diseño técnico, el diseño artístico o el diseño mecánico y la implementación y pruebas de los mismos.

Además, a lo largo de esta fase se generan muchos documentos . Los más importantes son el GDD, cuya finalización debería coincidir con la de esta fase, y la **Biblia del juego**, donde se recogen todas las historias de los personajes, del mundo donde sucede el juego, de su pasado y de los personajes secundarios que aparecen, creando el hilo argumental completo, con todos los detalles [Manrubia, 2014].

Esta fase termina con la generación del **gold master**, la copia definitiva que se envía a fábrica para su producción junto con el arte como la portada o el manual de usuario.

5.1.3. Fase de post-producción

Pero como ya sabemos el ciclo de vida de un producto software no termina con su entrega o, en este caso, su lanzamiento al mercado. Además de las

tareas de marketing pertinentes, será preciso llevar a cabo un seguimiento adecuado para dar respuesta al comportamiento que el mercado ha tenido en relación al juego, ya que puede llegar a ser muy diferente al esperado.

Las tareas más inmediatas pueden ser parches para arreglar elementos o para ajustar su funcionamiento, y si nuestro juego contempla el lanzamiento de expansiones o DLCs (*downloadable content*) también deberemos hacerlo en esta fase. Además, también deberá realizarse un **análisis post-portem** del proyecto para saber qué ha ido mal y poder mejorarlo para futuros proyectos.

Como puede verse, el desarrollo de un videojuego es una tarea larga y extremadamente compleja en la que intervienen profesionales de todo tipo y que es orquestada por un equipo de dirección que se encarga de que todo vaya según lo planeado y otro equipo de financiación encargado de que el presupuesto de producción se cumpla [Manrubia, 2014].

5.2. Metodología de trabajo

Antes de presentar el proceso de desarrollo y la metodología de desarrollo utilizada en este proyecto se describirá lo que es cada uno.

Un **proceso de desarrollo** es una representación abstracta de una metodología de desarrollo. Procesos de desarrollo como por ejemplo UDP o *Agile Development* son abstracciones que definen las actividades que deben realizarse, pero no especifican cómo realizarlas o cuál debería ser el resultado.

Por otro lado, una **metodología de desarrollo** es la implementación específica de un proceso de desarrollo. Ejemplos de eso son RUP, *eXtreme Programming* y *Scrum*. No son completamente explícitos pero definen qué, cuándo y cómo hacer las cosas.

Una vez diferenciados ambos conceptos, se pasa a explicar el proceso y la metodología de desarrollo que ha seguido este TFM.

5.2.1. Proceso de desarrollo

En vista de los requisitos de flexibilidad que propone, y que en un proyecto de esta naturaleza son especialmente importantes, se ha elegido seguir el **Agile Software Development**.

Los principios fundamentales que rigen el Desarrollo Ágil están expuestos

en **The Manifesto for Agile Software Development**¹, que fue escrito en febrero de 2001 por diecisiete ingenieros de software que llegaron a un acuerdo acerca de los cuatro pilares básicos que definían la forma correcta de desarrollar software.

1. **Individuals and interactions** over processes and tools.
2. **Working software** over comprehensive documentation.
3. **Customer collaboration** over contract negotiation.
4. **Responding to change** over following a plan.

La primera regla hace referencia a la importancia del factor humano y a su coordinación, y asigna un papel secundario a seguir la metodología del proyecto de manera invariable.

La segunda regla señala la necesidad de priorizar el software resultante y su funcionalidad y utilidad sobre una documentación exhaustiva del proyecto.

La tercera regla da a entender a la importancia de colaborar con el cliente tanto como sea posible, en lugar de firmar un documento estático al comienzo del proyecto que contenga todos sus requisitos y no aceptar nada que no apareciera inicialmente en él.

Por último, la cuarta regla se enfoca en tratar de tomar decisiones flexibles ante problemas repentinos y cambios no planificados, adaptando el proyecto a ellos, en lugar de tratar de seguir el plan inicial a toda costa.

Del mismo modo, se definieron **12 principios**² explicando qué debe regir cada metodología de desarrollo de software que utiliza este proceso de desarrollo.

5.2.2. Metodología de desarrollo

Dadas las necesidades de desarrollo explicitadas anteriormente, la metodología de desarrollo más cercana a ellas es **Scrum**. Scrum no es una técnica para la construcción de productos; en su lugar, es un marco en el que se pueden emplear varios procesos y técnicas. Por este motivo Scrum no es una metodología exclusiva de desarrollo de software, ya que puede aplicarse a otras áreas,

¹<http://agilemanifesto.org/>

²<http://agilemanifesto.org/principles.html>

se adapta muy bien y permite que el equipo sea flexible para abordar los problemas y lograr resultados de alto valor [Schwaber and Sutherland, 2013].

El principal valor de Scrum es su alta adaptabilidad a los cambios, lo que permite que el equipo y la organización del proyecto respondan rápidamente a los nuevos requisitos. Estos enfoques serían imposibles aplicando otras metodologías, como por ejemplo la de desarrollo en cascada.

Scrum se basa en la teoría empírica de control de procesos, que afirma que el conocimiento proviene de la experiencia y la toma de decisiones basadas en lo que se conoce. Esta metodología se apoya en tres pilares básicos.

- **Transparencia.** Cualquier acción significativa debe ser visible para cada persona en el proyecto. Solo de este modo se puede compartir una comprensión común de lo que está sucediendo en cualquier momento.
- **Inspección.** Los artefactos de Scrum deben ser inspeccionados para detectar resultados indeseados. Estas inspecciones serán más beneficiosas si son realizadas por inspectores cualificados y siempre que no sean tan frecuentes como para interferir en el trabajo.
- **Adaptación.** Scrum utiliza un enfoque iterativo e incremental para adaptarse a los cambios y para controlar los riesgos. Cuando se detecte un problema, estas adaptaciones se deben realizar lo antes posible para desviarse lo menos posible de la planificación inicial.

Etimológicamente, *scrum* es un movimiento en Rugby en el que un equipo se reúne para actuar de manera coordinada para que el balón pase de un extremo del campo al otro opuesto. Por lo tanto, hasta el nombre de esta metodología apunta a la necesidad de trabajar como un equipo sincronizado y bien comunicado para lograr un objetivo común.

De esta manera, Scrum intenta evitar que haya momentos en que un miembro del equipo no sepa qué hacer o no aporte valor al desarrollo, aunque Dilbert, en la figura 5.2, no esté de acuerdo con esto.

Artefactos y eventos

La figura 5.3 presenta una visión general de cómo funciona el flujo de trabajo de Scrum.

Para poder explicar los roles de Scrum más claramente, primero se describirán brevemente los eventos que forman parte de esta metodología [Schwaber and Sutherland, 2013].



Figura 5.2: Dilbert y los roles de equipo, de <https://dilbert.com>

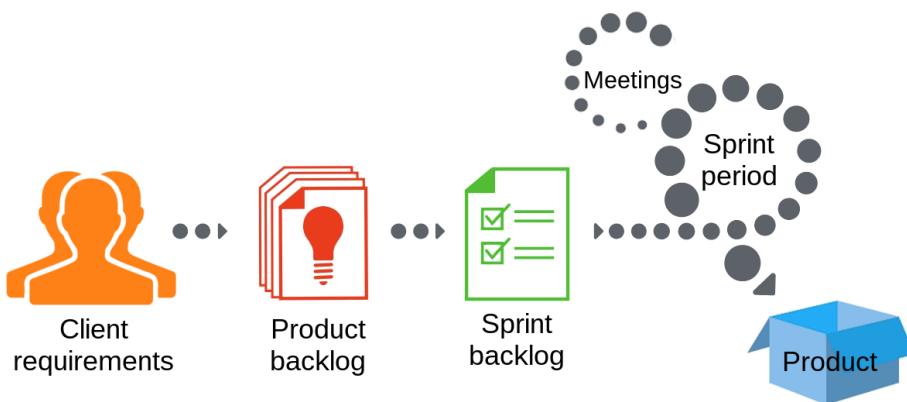


Figura 5.3: Flujo de trabajo de Scrum, adaptado de <https://www.scrumalliance.org>

- **Product backlog.** La cartera del producto es una abstracción que sirve para controlar las **historias de usuario** o *user stories*. Éstas reflejan los requisitos que debe satisfacer el proyecto, que son establecidos por el cliente o el propietario del producto.

Pero el *product backlog* no solo permite agregar historias de usuarios, sino también todas las características, funciones y soluciones a las necesidades del proyecto. Cada uno de ellas estará formado por una descripción, una prioridad y el valor que agrega al producto. Además, se puede añadir el miembro Scrum asignado para completarla y cualquier observación relevante.

El *product backlog* nunca se cierra, por lo que desde el inicio hasta el cierre del proyecto se pueden añadir nuevas historias de usuario o se pueden eliminar las que ya existen. Por lo tanto, a medida que el proyecto avanza la lista tiende a crecer y alcanzan su máximo justo antes de la entrega final. Los requisitos nunca dejan de evolucionar, por lo

que el *product backlog* es un **objeto vivo**. Los cambios en los requisitos comerciales, las condiciones del mercado o la tecnología pueden hacer que cambie.

- **Sprint.** El concepto clave de Scrum es un sprint, una **fase de trabajo** que dura un máximo de un mes, lo que genera un nuevo artefacto que puede ser potencialmente integrado en el producto final.

Se debe evitar que el tiempo del sprint sea demasiado largo, ya que de lo contrario la definición de lo que se está construyendo se vuelve demasiado amplia y la complejidad y el riesgo aumentan.

Cada sprint puede considerarse un proyecto autocontenido, ya que se utilizan para lograr objetivos. Cada sprint tiene una definición de lo que se construirá, un diseño y un plan flexible que guiará su construcción y el artefacto resultante. Un nuevo sprint comienza inmediatamente después de que el anterior termine.

Un sprint está formado por varios eventos y artefactos.

- **Sprint period.** Es el período de tiempo que el sprint está destinado a durar.
- **Sprint planning meeting.** Es una reunión dirigida por el **scrum master**, que debe asegurarse de que cada miembro haya entendido correctamente el resultado de la reunión.

Esta reunión, que se lleva a cabo al comienzo de cada sprint, permite planear el trabajo a realizar de forma colaborativa por parte del equipo Scrum a partir del *product backlog*.

- **Sprint backlog.** Es un conjunto de elementos que pertenecen al *product backlog* y que han sido **seleccionados para el sprint actual**. Es un pronóstico realizado por el equipo sobre qué funcionalidad estará en el próximo **incremento**. El *sprint backlog* puede contener más de un solo elemento; de hecho, el equipo de Scrum añade los elementos identificados como necesarios para cumplir con el objetivo del sprint.

Existen varias técnicas destinadas a lidiar con la necesidad de predecir cuánto puede durar una historia de usuario, como los **Mapa de afinidad** o el **Planning Poker**. Esta última es una técnica de estimación basada en el consenso dedicada a ayudar al equipo de desarrollo a tomar una decisión coordinada. Cada miembro hará una predicción en base a algunos valores prefijados y la representará el tiempo que cree que puede llevar terminar cada historia de usuario. Si todo el mundo coincide se dará por buena, pero si no se discutirá y se repetirá la votación hasta que todo el mundo coincida³.

³<https://www.mountaingoatsoftware.com/agile/planning-poker>

- **Daily Scrum.** Es un evento de 15 minutos para que el **equipo de desarrollo sincronice sus actividades** y cree un plan para el día actual. Esto se hace mirando el trabajo desde el último Scrum diario y prediciendo el trabajo que podría realizarse antes del siguiente. A lo largo de esta reunión, cada miembro del equipo responderá a tres preguntas:
 1. ¿Qué hice ayer?
 2. ¿Qué voy a hacer hoy?
 3. ¿Creo que hay algo que está dificultando el desarrollo?De esa modo es muy fácil saber exactamente qué está haciendo cada miembro del equipo y detectar problemas emergentes.
- **Sprint review.** Es una reunión informal que se lleva a cabo al final de cada sprint para **inspeccionar el trabajo realizado** y actualizar el *product backlog*. La duración de ésta reunión suele ser directamente proporcional a la duración del sprint.
- **Sprint retrospective.** Esta reunión se lleva a cabo después del *sprint review* y antes de la planificación del próximo sprint, y ayuda al equipo de Scrum a inspeccionar el trabajo realizado y **detecte puntos débiles** antes del el próximo sprint.
- **Product.** Es el motivo por el cual se desarrolla el proyecto; el artefacto final que se entregará al cliente.

Flujo de trabajo y roles

El marco trabajo de Scrum consta de varios roles; el **product owner** o propietario del producto, el **Scrum master** o maestro Scrum, **equipo de desarrollo** y los **stakeholders** o interesados, quienes juntos forman el **Scrum team**. Cada rol se relaciona con los otros por medio de varias actividades, eventos, interacciones y artefactos, y cada uno tiene un propósito específico y es esencial para el éxito del proyecto. Estas funciones y sus interacciones se muestran a continuación [James and Walter, 2010].

- **Product owner.** Es el responsable de **maximizar el valor del producto** coordinando correctamente las relaciones del Scrum team. Además, es el único responsable de administrar el *product backlog*, lo que incluye,
 - Definir las **historias de usuario**.
 - Ordenar las **historias de usuario** para alcanzar los objetivos de la mejor manera posible.

- Asegurarse de que el *product backlog* sea visible, transparente y **claro para todo el mundo.**
- **Scrum master.** Es el responsable de garantizar que la **metodología Scrum sea entendida y aplicada por todo el mundo.** El Scrum master debe actuar como líder al servicio del equipo Scrum y ayudar a aumentar el valor de los artefactos que genere.
El Scrum master puede ayudar a cada rol a maximizar su valor de varias maneras:
 - **Al product owner.** Su interacción principal con este rol está dirigida a garantizar la gestión adecuada del *product backlog*, haciendo que las historias de usuario sean lo más claras posible.
 - **Al equipo de desarrollo.** El Scrum master debe liderar al equipo de desarrollo y aumentar su efectividad para que generen productos de alto valor.
 - **A la organización.** El Scrum master debe ayudar a todas las partes interesadas a comprender y aplicar Scrum y trabajar con otros Scrum masters de otros equipos para aumentar la efectividad de la aplicación de esta metodología dentro de la organización.
- **Equipo de desarrollo.** Que está formado por los desarrolladores que entregan el trabajo al cliente. Solo los miembros del equipo de desarrollo pueden crear el incrementos en el producto y decir que una historia de usuario se ha completado.
Estos desarrolladores son **auto-organizados y multi-función.** Además, Scrum no reconoce sub-equipos dentro del equipo de desarrollo. Aunque los miembros individuales pueden tener habilidades especializadas y áreas de enfoque, la responsabilidad del producto pertenece al equipo de desarrollo en su conjunto.
- **Stakeholders.** Los interesados pueden ser individuos o empresas que se ven afectados por activa o por pasiva por el desarrollo del proyecto.

5.3. Conclusiones y aplicación de las metodologías

Como se ha visto, las metodologías de desarrollo de videojuegos son complejas, ya que requieren la sincronización y el trabajo conjunto de muchos profesionales. Sin embargo, en este proyecto solo ha trabajado una persona, por lo que se han simplificado estas metodologías, adaptándolas a los

requisitos del proyecto. Una de las principales ventajas de utilizar metodologías ágiles es esto mismo; poder modelar los procesos de desarrollo a las necesidades del proyecto a medida.

Como en los entornos reales de desarrollo de videojuegos, este proyecto se ha desarrollado en torno a su GDD, el documento que contiene toda su información, lo que significa que las iteraciones y el trabajo que conllevan se han basado en él.

Para ello, se ha diseñado un plan de entregas que guíe estas iteraciones y del que se hablará en el capítulo 6. Como se indica en Scrum, el final de cada iteración debe estar marcado por la generación de un entregable. Para probarlo, al final de cada iteración se grabará un vídeo mostrando los avances obtenidos a lo largo de la misma, que durará entre 4 y 6 minutos, y se subirá a YouTube. Además, en el capítulo 7, que habla del desarrollo del proyecto a través de sus iteraciones, se mostrarán estos vídeos y se indicará su URL para que el lector pueda verlos y comprobar que su fecha de subida concuerda con la de su iteración. Por último, como se indica en el GDD al final de juego se generará un tráiler que se utilizaría para tareas de marketing.

A continuación, en la figura 5.4 se presenta una imagen de la lista de reproducción generada al final del proyecto con todos los vídeos, tanto los diarios de desarrollo como el juego completo y el tráiler, para que el lector pueda entender mejor el resultado, que también puede consultarse en el siguiente enlace.

<https://www.youtube.com/playlist?list=PLb5boRKc04iapVxIHKnow08E2WnaLyIcT>

Además, se utilizarán repositorios públicos en GitHub para llevar un control de versiones tanto del proyecto como de su documentación, cuyos respectivos enlaces pueden verse a continuación. Además, una vez terminado el proyecto se subirá compilado al propio repositorio para que cualquier usuario pueda descargarlo y ejecutarlo.

<https://github.com/gomezportillo/mineRVa>

<https://github.com/gomezportillo/mineRVa-doc>

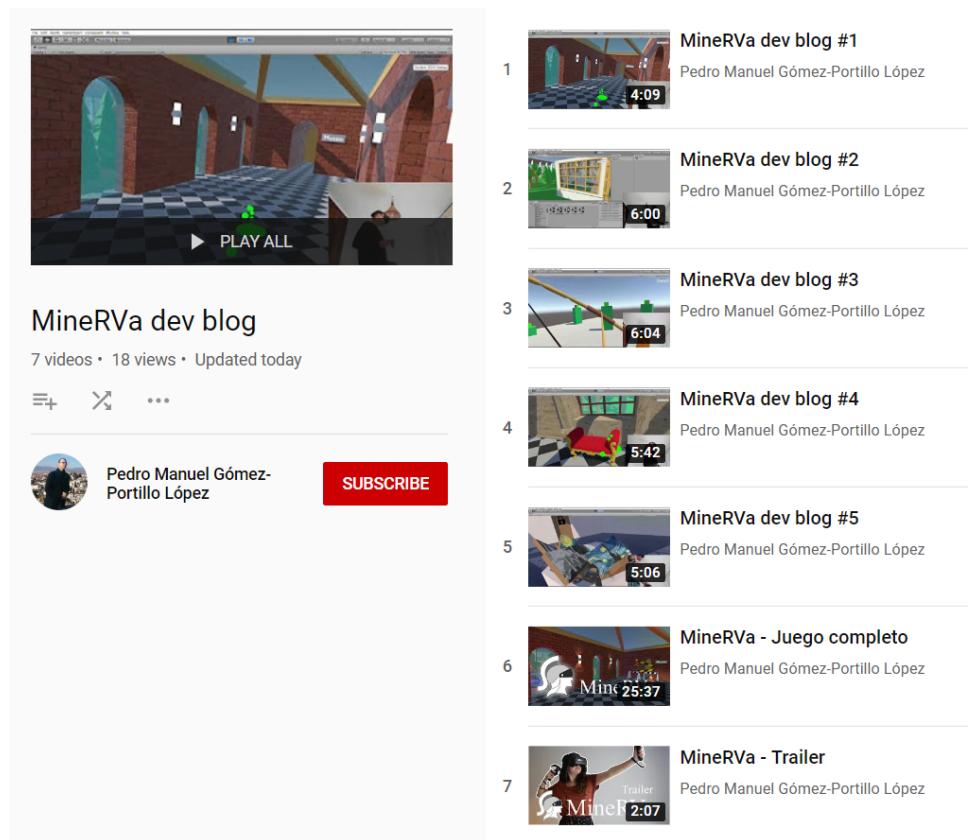


Figura 5.4: Lista de reproducción con los vídeos de *MineRVA*

Capítulo 6

Plan de entregas

En cualquier proyectos es necesario planear de antemano las distintas iteraciones en las que se dividirá el trabajo, pero en los casos en los que se trabaja con nuevas tecnologías esto se vuelve esencial, ya que esto facilita el poder realizar un seguimiento del avance y la evolución del proyecto.

Un plan de entregas es un proceso que permite programar los distintos esfuerzos que se desarrollarán a lo largo de un proyecto, aportando una visión global del mismo.

En este capítulo se presentará y desglosará el plan de entregas seguido para este TFM. Primero se hará en forma de tabla, que se ha usado a lo largo del proyecto para facilitar su seguimiento, y luego se pasará a describir de las entregas en más profundidad.

De aquí en adelante se hablará de *entrega* para definir el esfuerzo que genera un entregable y de *iteración* para describir las subtareas en las que se han dividido cada una de las entregas.

6.1. Plan de entregas

En las tablas siguientes puede verse un resumen de las iteraciones y sus tareas, aunque en la sección posterior se explicarán con más detalle.

Al final de todas las entrega se perfilarán las salas anteriores con el fin de mejorarlas de manera progresiva; como siempre será así, se ha evitado incluir esta iteración en la siguiente tabla, además de grabar un vídeo enseñando los avances obtenidos y subirlo a YouTube.

Por último, la iteración final estará centrada en realizar una evaluación

del proyecto con usuarios reales quieras, tras completar el juego, responderán dos cuestionarios con el objetivo de conocer su grado de satisfacción con el juego y su eficacia para aprender los conceptos que se pretende enseñar.

Entrega	Objetivo	Deadline
0	Tener una idea general de la narrativa del proyecto y tener un entorno de desarrollo configurado	15/2
	Iter.	Descripción
	1	Generar un documento con el prototipo de la narrativa
	2	Configurar el entorno de desarrollo con las librerías seleccionadas y el headset VR proporcionado
1	Tener dos salas que actúen como tutorial para los jugadores	28/2
	Iter.	Descripción
	1	Modelar y texturizar la antesala al museo
	2	Importarla a Unity correctamente
	3	Dotar a la sala de físicas y colisiones
	4	Conseguir que el jugador pueda moverse por ella
	5	Repetirlo todo con la primera sala de tutorial
	6	Modelar varias piezas de fruta y hacer que puedan ser cogidas y lanzadas
	7	Hacer que el jugador pueda moverse entre salas

Cuadro 6.1: Plan de entregas 0 y 1

Entrega	Objetivo	Deadline
2	Tener terminada la segunda sala, ambientada en el Gótico	15/3
	Iter.	Descripción
	1	Modelar, texturizar e importar la segunda sala
	2	Modelar la palanca y hacer que desbloquee la puerta de salida
	3	Comenzar a trabajar en la siguiente sala
	4	Hacer que cuando un objeto toque el suelo vuelva a su posición original
3	Tener terminada la tercera sala, ambientada en el Renacimiento, el prototipo un sistema de interacción avanzado y desarrollar tareas menores	31/3
	Iter.	Descripción
	1	Terminar de modelar la tercera sala
	2	Implementar el sistema/puzzle que haga que se abra la puerta
	3	Elegir qué sistema de interacción se va a hacer para la siguiente sala
	4	Desarrollar un prototipo funcional preparado para ser importado a la siguiente sala
	5	Mostrar la descripción de los cuadros al pulsar un botón
	6	Hacer un fundido a negro al cambiar de sala
4	Completar la cuarta sala, ambientada en el Barroco, comenzar a trabajar en la quinta y seguir trabajando en tareas menores	30/4
	Iter.	Descripción
	1	Modelar y texturizar la cuarta sala
	2	Importar el sistema de interacción avanzado
	3	Integrarlo en la sala para obtener un puzzle interesante
	4	Comenzar a modelar la quinta sala
	5	Añadir un candado a las puertas bloqueadas que desaparezca al abrir las

Cuadro 6.2: Plan de entregas 2, 3 y 4

Entrega	Objetivo	Deadline
5	Terminar la quinta sala, ambientada en el Romanticismo, y la sexta, ambientada en las Vanguardias	15/6
	Iter.	Descripción
	1	Completar el modelado la quinta sala
	2	Desarrollar un sistema de partículas que imiten neblina ambiental
	3	Diseñar e integrar un acertijo con alguna interacción interesante
	4	Modelar la sexta sala
	5	Integrar un puzzle que tenga que ser completado con las piezas de los cuadros elegidos
6	Diseñar y modelar la séptima, el almacén del vigilante y la sala final e integrar el resto de elementos restantes como diálogos	25/6
	Iter.	Descripción
	1	Modelar la séptima sala y elegir el cuadro robado
	2	Modelar la última sala
	3	Incluir un modelo del vigilante, añadirle animaciones e integrarlas con el juego
	4	Desarrollar un sistema de diálogos
	5	Añadir las descripciones al resto de cuadros
	6	Desarrollar un menú de juego inicial
	7	Añadir música de fondo y efectos de sonido

Cuadro 6.3: Plan de entregas 5 y 6

Entrega	Objetivo	Deadline
7	Terminar la documentación y generar un tráiler y un vídeo mostrando todo el juego	11/7
Iter.	Descripción	
1	Realizar una evaluación del juego final con usuarios reales	
2	Grabar un vídeo mostrando todo el juego	
3	Grabar y editar un tráiler de 90 segundos de duración	
4	Terminar de redactar y maquetar la documentación	
5	Realizar la presentación para la defensa con tribunal	

Cuadro 6.4: Plan de entrega 7

6.2. Descripción de las entregas

El desarrollo del proyecto ha sido guiado por un total de 7 entregas, cada una subdividida a su vez en una o varias iteraciones. El resultado de cada una de las entregas ha sido un entregable, permitiendo de este modo realizar un seguimiento preciso del desarrollo del proyecto.

Además, al final de cada entrega se generará un vídeo de entre 4 y 6 minutos mostrando los avances obtenidos a lo largo de la misma y que será subido a la plataforma YouTube. Al final de la descripción del desarrollo de cada entrega, en el capítulo 7, se indicarán los enlaces a cada uno de ellos.

El proyecto se inició a principios del segundo semestre y se planeó terminar a finales de junio, por lo que las fechas de las entregas se deberán ajustar para no sobrepasarlas. Además, se intentará que cada una dure aproximadamente dos semanas, ya que es un tiempo razonable para, aun compaginando el desarrollo de este proyecto con las seis asignaturas del segundo semestre, generar un entregable con el suficiente valor como para marcar el final de la entrega.

Por otro lado, y paralelamente a las entregas, se trabajará en esta documentación para que pueda ser terminada prácticamente al mismo tiempo que el desarrollo software.

6.2.1. Entrega 0

Se utilizará la primera entrega del proyecto como toma de contacto con el entorno de desarrollo, las tecnologías a utilizar y el proyecto en sí.

Como tareas se definirán generar un primer prototipo de la narrativa del proyecto enmarcándola en un museo creíble y realista y configurar el IDE y las librerías.

Los entregables resultantes de esta entrega serán, por un lado, un documento describiendo la narrativa y la estructura de salas del museo, y por el otro un proyecto Unity adecuadamente configurado que se pueda ejecutar con el headset VR proporcionadas y en el que el jugador pueda moverse de manera adecuada.

- **Fecha de inicio.** 1 de febrero de 2019
- **Fecha de fin.** 15 de febrero de 2019

6.2.2. Entrega 1

Una vez que contamos con una narrativa de la que partir y un entorno de desarrollo correctamente configurado, el objetivo principal de la siguiente entrega será aprender a exportar correctamente modelos 3D de Blender a Unity y a trabajar de un modo básico con el framework VRTK.

Esta entrega tendrá varias iteraciones; primero, modelar una antesala al museo relativamente simple en Blender, aplicarle materiales y texturas y aprender a importar todo correctamente desde Unity y una vez conseguido, el siguiente paso será configurar la escena para dotarla de físicas y colisiones para que al iniciar el juego el jugador pueda moverse por ella de un modo similar a como lo haría en una habitación real.

Siguiendo con las salas, y una vez aprendido cómo exportarlas al motor del juego, se modelará la sala que actuará como tutorial al juego y a sus dinámicas y se modelarán y configurarán un par de piezas de fruta con las que el usuario pueda interactuar; cogerlas y soltarlas encima de otros modelos e incluso lanzarlas y que estas se comporten de manera realista.

El resultado de esta entrega será una versión del proyecto con las dos salas explicadas anteriormente y que permita al usuario moverse entre ellas.

- **Fecha de inicio.** 16 de febrero de 2019
- **Fecha de fin.** 28 de febrero de 2019

6.2.3. Entrega 2

La siguiente entrega consistirá en seguir aprendiendo a trabajar con el framework elegido, permitiendo realizar tareas inicialmente más difíciles como mover objetos programáticamente haciendo uso del framework de VR.

En lo que respecta a sus iteraciones, y como la siguiente sala a modelar, ambientada en el gótico y la primera temática del museo, consta de muchos detalles y técnicas más avanzadas de modelado, así que se espera dedicar más tiempo en este trabajo. Por lo tanto, será la única que se modelará. Por otro lado, se volverá a las dos primeras salas ya modeladas para terminarlas y mejorar su apartado gráfico. Además, si sobra tiempo, se comenzará a modelar la siguiente sala.

El entregable por tanto, será una versión del proyecto en la que se puede hacer uso de aspectos más avanzados de las librerías utilizadas para permitir al usuario interactuar con modelos tridimensionales que actúen como interfaces para controlar otros elementos. Siguiendo con el documento inicial, se conseguirá abrir y cerrar una puerta corredera a la que inicialmente el jugador no tiene acceso interactuando con una palanca.

- **Fecha de inicio.** 1 de marzo de 2019
- **Fecha de fin.** 15 de marzo de 2019

6.2.4. Entrega 3

La tercera entrega del proyecto pretende ser algo más práctica. Una vez teniendo más soltura con el framework y el entorno de desarrollo, se valorará la inclusión de técnicas de interacción mucho más avanzadas, que quizás en un principio por falta de experiencia con las tecnologías y las librerías utilizadas ni siquiera se plantearon. Principalmente, se estudiará el incluir elementos como un arco o una pistola que el jugador pueda disparar para acertar a objetivos. Además, se trabajará en tareas que inicialmente podían resultar más difíciles como hacer un fundido a negro al cambiar de sala, mostrar la descripción de las obras de arte de las salas (que serán leídas desde un archivo en disco) con la interacción el jugador con los mandos o teletransportar un objeto de vuelta a su posición original si toca el suelo para que el jugador no tenga que agacharse a recogerlo.

Paralelamente, se seguirá trabajando en el resto de salas, perfilándolas y terminado aquellos modelos a los que les quedaran los últimos retoques.

El resultado de esta entrega, por lo tanto, serán las 4 salas anteriores

prácticamente terminadas y un prototipo funcional con alguna técnica de interacción avanzada para la quinta.

- **Fecha de inicio.** 15 de marzo de 2019
- **Fecha de fin.** 31 de marzo de 2019

6.2.5. Entrega 4

La cuarta entrega del proyecto partirá del prototipo desarrollado en la entrega anterior para modelar la siguiente sala del proyecto y dotarla de la lógica necesaria para que la interacción que se desarrolle en ella sea interesante y divertida para el jugador. Además, se comenzará a trabajar en la siguiente sala, la sexta, modelando su estructura básica.

El principal problema de esta entrega es que dadas las fechas en las que se desarrollará, previsiblemente se solapará con Semana Santa y los distintos trabajos de las asignaturas del máster, por lo que se asignará el doble de tiempo para terminarla, es decir, un mes.

- **Fecha de inicio.** 1 de abril de 2019
- **Fecha de fin.** 30 de abril de 2019

6.2.6. Entrega 5

En esta entrega, la quinta, se terminará de modelar la sala del Romanticismo y se completará el desarrollo de la prueba elegido. Tras ello, se diseñará, modelará y desarrollará de manera íntegra la sexta sala, ambientada en las Vanguardias.

Por tanto, el resultado de esta entrega será una versión jugable del proyecto con todas las salas temáticas finalizadas y sus pruebas terminadas.

Debido a la fecha en la que deberá realizarse, esta entrega coincidirá completamente con los exámenes y las entregas de los trabajos del máster; por ello, se le ha asignado más tiempo que al resto de entregas para poder ser completada correctamente.

- **Fecha de inicio.** 1 de marzo de 2019
- **Fecha de fin.** 7 de junio de 2019

6.2.7. Entrega 6

La sexta entrega del proyecto concluirá la parte software del proyecto; se deberá terminar el desenlace del juego, además de un pequeño epílogo para que el final no sea tan abrupto. Para ello, se incluirá el modelo de un vigilante con el que el jugador pueda interactuar, por lo que también se le añadirán animaciones y se desarrollará un sistema de diálogos.

Para aumentar la sensación de realismo y de inmersión del jugador, se añadirá música de fondo diferente para cada sala, libre de derechos de autor, y efectos de sonido para algunas de las acciones que el jugador más realice.

Además, se terminará de incluir las descripciones de las obras de arte que hay en el museo.

Por último, se trabajará en generar un menú inicial desde el que el jugador pueda empezar una partida, ver los créditos o salir del juego.

Aunque para estas fechas se deberá estar finalizado las clases y las entregas del máster y comenzando con las prácticas de empresa, se deberá compaginar el desarrollo de este proyecto con las prácticas de empresa, por lo que se le ha asignado un tiempo de dos semanas, como a las entregas iniciales.

- **Fecha de inicio.** 7 de junio de 2019
- **Fecha de fin.** 25 de junio de 2019

6.2.8. Entrega 7

Finalmente, la última entrega del proyecto se dedicará a completar la documentación restante y realizar una evaluación de satisfacción del juego final con usuarios reales. Esta evaluación se incluirá en la documentación y se usará como medida de aproximación a la satisfacción de los jugadores con respecto al proyecto.

Además, se grabará un vídeo que muestre el juego completo, de inicio a fin, y se generará un pequeño tráiler, del estilo de un anuncio de televisión, que se usará para promocionar el juego.

Por último, se generará la presentación del proyecto que se utilizará en la defensa del TFM y que resumirá esta documentación.

- **Fecha de inicio.** 25 de junio de 2019
- **Fecha de fin.** 11 de julio de 2019

Capítulo 7

Desarrollo

COMO ya se ha comentado previamente, el desarrollo de este proyecto ha supuesto un reto importante ya que las tecnologías utilizadas eran totalmente desconocidas. En este capítulo se explica el proceso seguido a lo largo de su desarrollo, además de presentar los principales problemas encontrados y las soluciones con las que se han abordado.

Para cada entrega se incluirán los bocetos realizados a mano para los modelos, además del modelo renderizado en Unity y uno de los fragmentos de código desarrollado que se considere más interesante, además de los diagramas pertinentes que ayuden a entenderlos.

Además, como algunos de los elementos desarrollados en una entrega sufrieron cambios en las siguientes, solo se explicará la versión final, por lo que puede que algo de lo explicado para una entrega no corresponda exactamente con lo enseñado en el vídeo de su entrega.

Aunque algunas de las tareas de modelados de las salas han sido bastante complejas, como este trabajo no pretende centrarse en ellas la mayoría se obviarán y en su lugar se hablará de tareas más técnicas relacionadas con el diseño y la programación del código.

Por otro lado, la naturaleza de este proyecto hace que sea muy visual, pero como no se pretende llenar este capítulo de capturas de pantalla, se deja al lector el consultar estos vídeos para ver cada uno de los elementos en acción.

7.1. Entrega 0

El objetivo de esta primera entrega fue, por un lado, establecer la estructura principal del proyecto y esbozar una primera versión de la narrativa, y por otro, realizar una primera toma de contacto con el desarrollo de Unity, las tecnologías VR e instalar y configurar el entorno de desarrollo para poder empezar a trabajar en la siguiente entrega.

7.1.1. Estructuración del proyecto

El primer paso lógico tras proponer al tutor de este TFM el proyecto y ser aprobado fue empezar a definirlo. Para ello, y partiendo de la idea principal de desarrollar una experiencia de juego haciendo uso de tecnologías VR que pusiera en contacto con el mundo del arte a personas que suelen y no visitar museos, se generó el documento que puede verse en el anexo 9.1.

Este documento comienza a detallar la narrativa del juego y la integra en una visita por un museo ficticio y, aunque terminó por sufrir varios cambios importantes, sirvió para definir el punto de partida del proyecto.

A lo largo de este documento se intenta crear una historia interesante para el jugador al mismo tiempo que generar un museo ficticio pero realista y coherente en el que se pueda desarrollar dicha narrativa. Para ello, se ha seguido el orden cronológico por las principales épocas de la historial de arte, en la que a cada sala le corresponde una etapa diferente.

A la hora de elegir los cuadros que se mostrarían en las salas se intentó buscar aquellos más representativos de su época. Para ello, se contó con el asesoramiento de una historiadora del arte, que ha sido quien ha dado el visto bueno al rigor artístico del museo y sus salas.

7.1.2. Primera toma de contacto con Unity

A continuación se presenta un resumen de la interfaz de Unity para que el lector pueda entender algunos de los conceptos de los que se hablarán más adelante.

1. Vista de la escena actual, en la que el desarrollador puede obtener una vista previa de la escena e interactuar con los objetos tridimensionales para colocarlos. Funciona de manera parecida a Blender, aunque se usan controles totalmente diferentes.

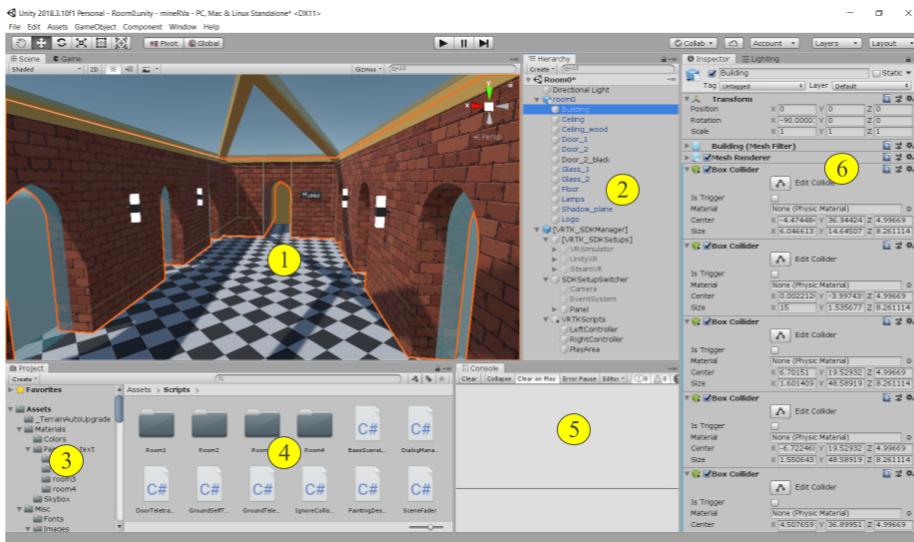


Figura 7.1: Resumen de la interfaz de Unity

2. Vista de la jerarquía de la escena, en la que pueden verse los objetos que hay y sus relaciones, como por ejemplo si están empanteados.
3. Vista del inspector en la que aparece la información de los componentes de un objeto seleccionado. Un componente puede ser prácticamente cualquier cosa desde un script a un material.
4. Vista del proyecto, donde aparecen todas las carpetas disponibles.
5. Vista donde aparecen los elementos de la carpeta seleccionada. En este caso, pueden verse algunos de los scripts con los que se ha trabajado.
6. Consola de salida en la que aparece información del proyecto.

7.1.3. Configuración del entorno de desarrollo

Antes de poder empezar a desarrollar el proyecto fue necesario elegir un IDE y un framework con el que trabajar. Como ya se ha explicado en los capítulos 2 y 4, tras comparar las ventajas y desventajas de los entornos de desarrollo y librerías disponibles en el mercado se decidió trabajar con Unity, el framework VRTK y la librería SteamVR para lo que, tras instalarlas, hubo que importarlas manualmente al proyecto de Unity.

La principal motivación de usar este framework es toda la configuración por defecto que trae; de no haber utilizado estas librerías, hubiera sido necesario implementar de 0 un sistema de Realidad Virtual interactivo, lo que

hubiera sido un TFM por si mismo. En cambio, ha sido posible importar directamente y con relativa poca configuración un sistema funcional preparado para trabajar con él.

Por otro lado, en lo que respecta al movimiento y desplazamiento del usuario, se presentaron dos vertientes.

- Utilizar el **joystick** de los mandos para moverse, como si de un juego clásico se tratara. El problema de este enfoque es que provoca **mareos** en los jugadores, en algunos casos graves. Esto es debido a que cuando los ojos detectan un movimiento que el sistema de equilibrio no, se genera una sensación muy incómoda que obliga al jugador a parar inmediatamente de jugar. En mi caso, por ejemplo, intenté jugar al *Pavlov VR*, un juego que implementa este modelo, y apenas pude jugar tres minutos antes de tener que parar y tumbarme.
- En contraposición, el modelo que están implementando la mayoría de juegos VR es de **lanzar** a los jugadores, quienes utilizan el joystick para apuntar al sitio donde quieren moverse, como si de una catapulta se tratase, y se teletransportan al lugar indicado, evitando de este modo los mareos por completo.

Por tanto, se ha elegido el segundo modelo para utilizar en este juego.

7.2. Entrega 1

Como se indicó en el capítulo 6, el objetivo final de las iteraciones de esta entrega fue aprender a importar modelos a Unity desde Blender y diseñar e implementar el tutorial del proyecto y que éste fuera completamente funcional.

7.2.1. Modelado e importación

Lo primero que se hizo antes de comenzar a modelar en Blender, ya que es mucho menos productivo empezar a trabajar sin una idea previa, fue diseñar un boceto en papel en el poder basar el modelado posterior.

Como se consideró que el museo sería más realista si en lugar de empezar directamente en él el jugador tuviera que recorrer un pequeño pasillo que funcionara de antesala y desde el que se pudiera ver el exterior, fue el primer boceto que se hizo, y tras él se dibujó la sala que actuaría de tutorial. Además, en esta antesala se recibiría una llamada de la directora del museo

en la que presenta al jugador como un detective de renombre y expone el problema del cuadro robado. Tras ello, le indicaría que hablase con el vigilante del museo para más información.

Tras ello, en la primera sala se tendría que presentar un cuadro muy reconocible y una pequeña prueba relacionada con él, por lo que se decidió utilizar el cuadro *El Hijo del Hombre* de René Magritte (1964) y que el jugador tuviera que cambiar de sitio una pieza de fruta relacionada con este cuadro, aprendiendo de este modo que puede interaccionar con los elementos virtuales y que habrá relación entre las pruebas y las obras de arte de tal. De este modo, se pretende que el reto sea darse cuenta de estas ideas y no la propia prueba en sí, ya que es algo trivial. La imagen 7.2 muestra el boceto de estas dos salas, que fue dibujado antes de comenzar a modelarlas para disponer de un punto de partida.

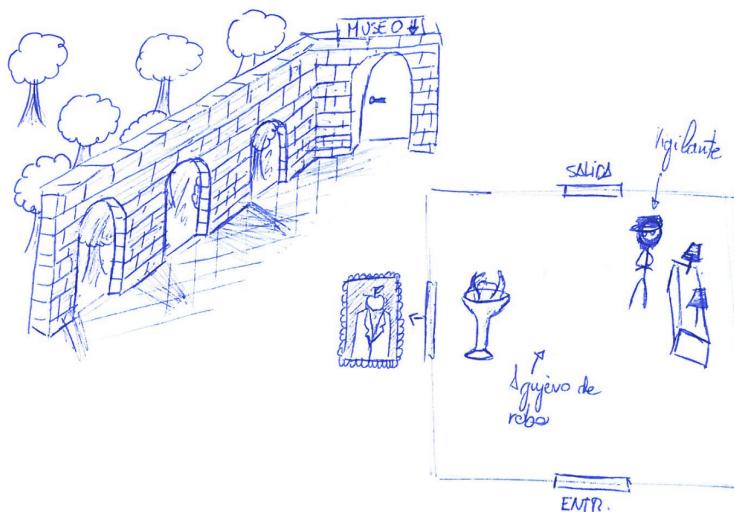


Figura 7.2: Boceto de la antesala y la sala de tutorial

Una vez terminados los bocetos, se modeló la primera sala en Blender y se comenzó a trabajar en importarla desde Unity, para lo que es necesario crear una escena e importar en ella el archivo Blender desde el gestor de archivos. Una vez que lo hagamos, aparecerá como un **Prefab**¹. De este modo, cuando el archivo Blender se modifique Unity lo detectará y actualizará su copia local, aunque por ser un *prefab* no pueden modificarse desde Unity sin perder esta propiedad, por lo que todos los cambios deberán ser hechos desde Blender. La principal ventaja de esto es que si en un futuro se quisiesen portar los modelos a otro motor gráfico, simplemente habría que importar el archivo Blender sin necesidad de acceder al de Unity.

¹<https://docs.unity3d.com/es/current/Manual/Prefabs.html>

Actualmente, Blender cuenta con dos motores de renderizado; Blender Internal y Blender Cycles, y no son compatibles entre sí. Esto quiere decir que si por ejemplo creamos un material en Blender Internal y luego cambiamos a Cycles, éste no aparecerá. Aunque en teoría Unity trabaja mejor con los materiales de Blender Internal, al importarlos no aparecen como deberían y no pueden modificarse sus propiedades como su color, su *metallicidad* o su mapa de normales, por lo que ha sido necesario rehacer todos los materiales de los modelos y volver a aplicarlos a mano.

Vamos a tomar como ejemplo una de las paredes de ladrillos para ver el flujo de trabajo de los materiales; tras modelarla, habría que descargar una textura para ella, para lo que se ha utilizado el sitio web <https://3dtextures.me/> que proporciona texturas procedurales gratuitas y con mapas de normales y rugosidad con licencia libre. Una vez hecho, se crea un material en Blender al que se le aplica la textura para ver cómo quedaría, aunque Blender Internal no da la opción de añadir más mapas a la textura. Tras esto, se importa el modelo desde Unity y se crea un nuevo material, con la misma textura, al que se le añaden y configuran el resto de mapas y que se aplica al modelo.

La imagen 7.3 muestra el resultado final del modelado y la importación a Unity. Como el exterior con árboles, que puede verse a través de las ventanas, se reutiliza en otras salas se ha movido a una escena aparte que se importa cuando es necesario, reduciendo de este modo el peso de las salas desde las que se necesita ver el exterior.



Figura 7.3: Sala 0 renderizada en Unity

Tras ello se hizo lo mismo con la sala de tutorial, que puede verse en la figura 7.4. Para dar a entender al jugador que están relacionadas, se ha utilizado la misma textura de ladrillos para las paredes y el mismo techo

de cristal, lo que ayuda a aumentar la claridad y darles una sensación de amplitud.



Figura 7.4: Sala 1 renderizada en Unity

Además, aunque las cajas de colisiones automáticas de Unity funcionan bien para objetos no lo hacen para habitaciones, ya que estas cajas la rodean y no permiten detectar colisiones interiores. Por ello, ha sido necesario definir manualmente estas colisiones, para lo que se han utilizado los componentes **Box Collider** para cada una de la paredes, el techo y el suelo. Estas cajas de colisiones definen los *límites físicos* de los objetos e impiden que el jugador los atraviese.

7.2.2. Viajar entre salas

Una escena en Unity es una unidad que permite incluir en ella elementos que estén estrechamente relacionados desde el punto de vista del juego, como modelos o scripts. Desde la documentación de Unity se anima al desarrollador a encapsular cada nivel del juego en una escena; así, la equivalencia que se ha usado en este proyecto es de una sala por escena.

Una vez que las dos salas estaban modeladas se trabajó en hacer que el jugador pudiera viajar entre ellas; para ello, se escribió un script en C# que permite cambiar la escena actual a otra cuando el jugador toque una puerta.

Para ello, lo primero que se hizo fue añadir una caja de colisiones sin físicas a la puerta, lo que permite añadir un *listener* para detectar colisiones y poder activar otras funciones. Tras ello, se desarrolló y añadió un script como componente, que puede verse simplificado en el listado 7.1, que usa la clase **SceneManager** para cambiar la escena cuando el jugador colisiona con ella

tras comprobar previamente que no se encuentra cargada. En él se declaran dos variables públicas para poder definirlas desde el propio inspector de Unity más cómodamente, como puede verse en la figura 7.5, lo que añade flexibilidad y reutilización al código. El script entero se encuentra en el archivo `DoorTeletransporter.cs`.

```

1 public string sceneName;
2 public float fadingTime = 10.0f;
3
4 private void OnTriggerEnter(Collider other)
5 {
6     if (sceneName != String.Empty &&
7         !SceneManager.GetSceneByName(sceneName).isLoaded)
8     {
9         SceneManager.LoadScene(sceneName,
10             LoadSceneMode.Single);
11    }
12 }
```

Listing 7.1: Fragmento del script para viajar entre salas



Figura 7.5: Script para cambiar de salas desde el inspector

Además, Unity utiliza la convención *CamelCase* para nombrar sus variables, que es un estilo de escritura que se aplica a frases que omite los espacios y hace que cada palabra empiece en mayúsculas . Por ello, si la implementamos en nuestro código es capaz de reconocer las palabras y representar el nombre de las variables en el inspector correctamente.

Por otro lado, cada script puede implementar dos funciones, `Start()` y `Update()`, que se ejecutan automáticamente al inicio de la escena y en cada frame, respectivamente. Este paradigma es totalmente distinto a la programación lineal, ya que es necesario realizar cualquier cambio de manera iterativa; por ejemplo, si queremos mover un objeto un metro durante un segundo, deberemos parametrizar las distancias y los tiempos para que esta posición se actualice sesenta veces cada segundo, suponiendo una tasa de refresco de pantalla de 60Hz.

7.2.3. Interacción con objetos virtuales

Una vez modeladas las dos salas, se comenzó a trabajar en hacer que el jugador pudiera interactuar con los objetos virtuales. Al estar utilizando el framework VRTK, se han podido hacer uso de sus funciones para facilitar mucho el trabajo.

Lo primero que se hizo, tras modelar las tres piezas de fruta (una manzana, un plátano y una pera) e importarlas a Unity, fue dotarlas de físicas, para lo que se utilizaron los componentes `Box Collider` para definir sus límites y `Rigidbody` para hacerlas responder a fuerzas como la gravedad. Tras ello se hizo que interactuarán con los mandos del jugador con ayuda de los componentes `VRTK_Interactable_Object`, `VRTK_Child_Of_Controller`, `VRTK_Interact_Haptics` y se hizo que apareciera un borde amarillo cuando su caja de colisión detectara el mando con ayuda del componente `VRTK_Outline_Object_Highlighter`. Tras ello, se utilizó una *snap drop zone* o zona en la que poder colocar objetos, para lo que se adaptó una de las *prefabs* proporcionadas por el framework. Tras configurarla adecuadamente, fue posible colocar sobre esta zona piezas de fruta y que estas automáticamente adquirieran la posición y rotación adecuadas.

Lo que se explica a continuación no se hizo hasta dos entregas posteriores, pero se contará ahora por estar estrechamente relacionado con ello. Para evitar que el jugador saliese de la sala sin hacer caso al vigilante, se añadió un script a la *snap drop zone* antes mencionada que detectara cuando el jugador dejaba un objeto con la etiqueta `Apple` para desbloquear la puerta. Por un lado, para conseguir detectar cuando el jugador colocaba objetos en la zona se añadió un *listener* al evento `ObjectSnappedToDropZone`, emitido por el componente `VRTK_SnapDropZone`, que permite saber programáticamente cuándo ocurre esto. Por otro, para deshabilitar la puerta y evitar que el jugador viaje entre salas, se deshabilita inicialmente su caja de colisiones y solo se vuelve a activar cuando el jugador coloca la manzana en su sitio. Más adelante, también se utilizaría este sistema para activar uno de los diálogos del vigilante que felicitase al jugador por haberlo hecho bien.

Como resultado de esta entrega se generó el primer entregable y, por tanto, se grabó un vídeo presentando el proyecto y enseñando los avances que puede verse en el siguiente enlace.

7.3. Entrega 2

La segunda entrega del proyecto estuvo más centrada en el modelado que la primera. Tras diseñar el boceto de la sala del gótico, que puede verse en la figura 7.6, se modeló el Blender. Al ser un modelo lleno de detalles y diseños y formas muy específicos, como los arcos o las ventanas, se tardó bastante en terminar. El resultado final puede verse en la figura 7.7.

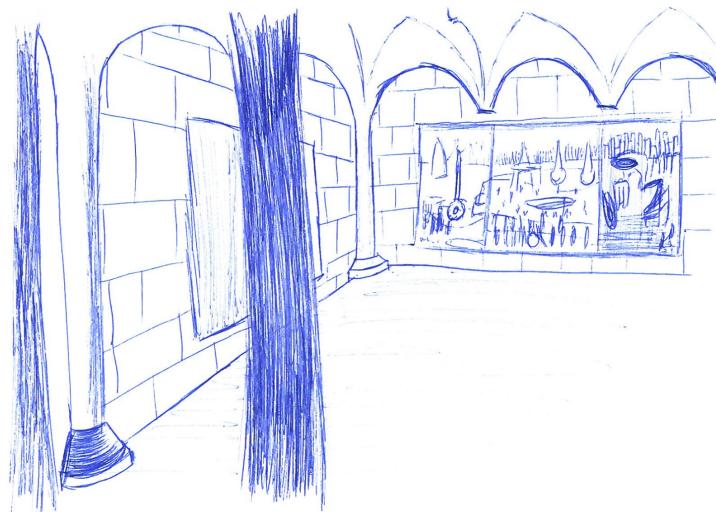


Figura 7.6: Boceto de la segunda sala



Figura 7.7: Sala 2 renderizada en Unity

7.3.1. Activación de una palanca virtual

Una vez se contaba con el modelo, se pasó a implementar la prueba. Para esta sala se decidió esconder una palanca tras uno de los cuadros laterales que, al activarla, abriera en dos el *Jardín de las Delicias*. Para ello, una vez modelado el hueco en la pared con la palanca detrás del cuadro elegido, se utilizó el script del framework VRTK `VRTK_ArtificialRotator` y se integró con un script propio para dotar de dicha funcionalidad tanto a la palanca como a la puerta. A continuación, en la figura 7.8 puede verse un fragmento de este último script desde el inspector de Unity y cómo contiene al primero. Además, también puede verse cómo se han asignado el resto de elementos mencionados para que pueda acceder a ellos y modificarlos.

Por ejemplo, desde este script es posible definir el contenido del cuadro de texto que muestra si el cuadro está abierto o cerrado, su velocidad de apertura y la referencia a las dos piezas que forman el cuadro, que se utilizará para acceder a ellas y modificar su posición.

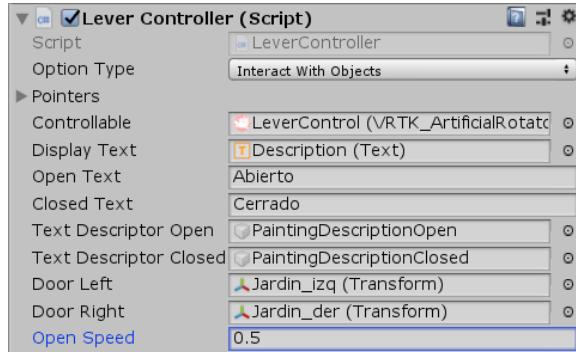


Figura 7.8: Script para controlar la palanca desde el inspector

A continuación, en el listado 7.2 se presenta simplificado el script utilizado para abrir y cerrar la puerta, que se puede ver completo en el archivo `LeverController.cs`. Como puede verse, todo se realiza dentro del método `Update()`, actualizando en cada frame la posición de los elementos utilizando la función `Lerp` de la clase `Vector3`, que implementa una función de interpolación lineal que permite modificar un vector tridimensional.

```

1 public Transform doorLeft;
2 public Transform doorRight;
3 public float openSpeed;
4
5 private void Update()
6 {
7     doorLeft.position = Vector3.Lerp(doorLeft.position,

```

```

8             leftPosition ,
9             Time.deltaTime *
10            openSpeed);
11
12         doorRight.position = Vector3.Lerp(doorRight.position,
13                                              rightPosition,
14                                              Time.deltaTime *
15                                              openSpeed);
16     }

```

Listing 7.2: Fragmento del script para abrir y cerrar una puerta abatible con una palanca

Además, multiplicando la velocidad de apertura por el tiempo transcurrido desde el último frame ejecutado (obtenido con la función `Time.deltaTime`), volvemos dependientes estas dos variables y podemos asegurar que en todos los ordenadores en los que se ejecute se abrirá a la misma velocidad, evitando así que el juego se ejecute más rápido en ordenadores más potentes que renderizan más imágenes por segundo.

Por otro lado, para permitir al jugador abrir y cerrar el cuadro que esconde la palanca como si fuera una ventana, se ha utilizado el script `VRTK_Physics_Rotator` al que indicándole un vector que actúa como bisagra y el ángulo de apertura, además de otros parámetros, es capaz de permitir rotar el objeto que lo contiene.

7.3.2. Teletransportar los objetos que toquen el suelo

Para evitar que el jugador tuviera que agacharse a recoger los objetos virtuales que caen al suelo (lo que puede ser peligroso, ya que con las gafas no se ve el mundo real y se puede llegar a chocar contra algún mueble), se diseñó un pequeño script que hace que cuando estos objetos chocan contra el suelo vuelvan a su posición original. Este script hace que cuando la caja de colisión de un objeto detecta un choque comprueba si es el suelo, y si sí cambia la posición de dicho objeto con la que tenía al cargar la escena, que había almacenado previamente, automatizando la tarea.

En el listado 7.3 puede verse el fragmento de este script que detecta las colisiones, cambiando la posición y la rotación del objeto para que coincida con su inicial. Además, también se resetean sus velocidades iniciales y angulares, ya que de otro modo el objeto mantendría la velocidad y la aceleración que tenía mientras caía, lo que antes de implementarlo hacía que rebotaran contra las superficies y volvieran a caer cada vez más y más rápido.

```

1 private void OnTriggerEnter(Collider other)

```

```

2 {
3     if (other.tag == GroundTag && instance != null)
4     {
5         Rigidbody rb =
6             instance.GetComponent<Rigidbody>();
7
8         instance.position = initialPosition;
9         instance.rotation = initialRotation;
10        rb.velocity = initialVelocity;
11        rb.angularVelocity = initialAngularVelocity;
12    }

```

Listing 7.3: Fragmento del script para teletransportar un objeto si toca el suelo

Y por último, se comenzó a prototipar la siguiente sala, aunque como esta tarea apenas se empezó en esta entrega, su explicación se deja para la siguiente.

Por tanto, como resultado de esta entrega se generó una versión del proyecto que contenía la antesala y las dos primeras salas funcionalmente terminadas y un prototipo con la estructura básica de la tercera. El vídeo que enseña los avances puede verse a continuación.

<https://youtu.be/kfEnxP5dHU4>

7.4. Entrega 3

A continuación se comenzó a trabajar en la tercera entrega. Como la anterior sala se terminó y se empezó a modelar la tercera, lo primero que se hizo fue terminar el modelo de la misma. En la figura 7.9 puede verse el boceto que se dibujó antes de comenzar a modelar para tener un punto de partida.

A continuación, en la figura 7.10 puede verse la sala ya terminada e importada desde Unity. Finalmente, la prueba de esta sala consiste en ayudar al vigilante a recoger restos que otros visitantes han dejado en el museo. En concreto son 4; una lata de refresco, una botella de cristal, una taza de café y una tela que tapa uno de los cuadros que ha sido modelada utilizando el motor de físicas de tejidos de Blender para que fuera realista. Una vez que el jugador los ha depositado todos en la papelera que puede verse al fondo, la sala se desbloquea y puede continuar visitando el museo.

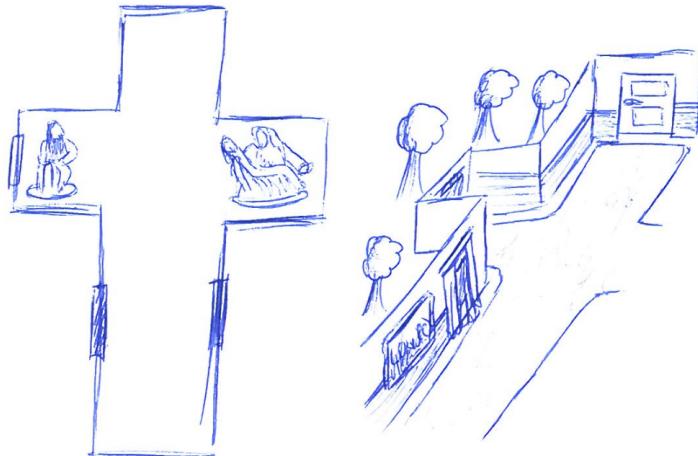


Figura 7.9: Boceto de la tercera sala



Figura 7.10: Sala 3 renderizada en Unity

7.4.1. Recogiendo basura virtual

Para hacer los elementos presentados anteriormente interactivos se utilizó un procedimiento similar a las piezas de fruta de la primera sala, utilizando la funcionalidad y las interfaces que provee el framework VRTK. Por tanto, se omitirá su explicación y se pasará directamente a presentar el funcionamiento de la papelera.

Aunque inicialmente se consideraron varios enfoques para hacer funcionar esta papelera, como hacer que el jugador tuviera que depositar dentro

los objetos, tras varias pruebas se vio que no siempre caían dentro, y como a estos objetos también se les añadió el script para que volvieran a su sitio al caer, el jugador tenía que volver a la posición de origen de nuevo a por ellos, lo que era bastante tedioso. Por ello, finalmente se decidió que simplemente con soltar los objetos cerca de la papelera, automáticamente se metieran dentro de ella y desaparecieran. Para ello se volvieron a utilizar las *snap drop zones* de VRTK.

Por tanto, se colocaron cuatro zonas concéntricas, una por cada objeto, y se configuraron de tal manera que al dejar un objeto dentro hacían que su tamaño se encogiese hasta ser prácticamente invisible, tras lo que se eliminaban. Por otro lado, para detectar cuándo una de estas zonas detectaba un objeto, se utilizaron eventos de Unity. Estos eventos permiten suscribirse a ellos y resultar notificado cuando se activen, y las *snap drop zones* ofrecen sus propios eventos. Por tanto, una vez es posible acceder a las 4 zonas desde el código basta con utilizar el operador `+=` para suscribir un método que implemente la interfaz necesaria al evento de un objeto.

En el listado 7.4 pueden verse simplificadas las fragmentos más relevantes de este script. La primera línea declara un vector público de *snap drop zones*, lo que permite definir desde el inspector su tamaño y sus elementos. Posteriormente, cuando la escena comienza, se recorre este vector con el iterador `foreach` y se suscribe un método propio a los eventos de todas las zonas que anuncian que un objeto le ha sido agregado. Así, cada vez que una zona adquiera un objeto se llamará a la función de la línea 15, que comprobará si el objeto añadido corresponde con aquellos que nos interesan y podrá de este modo llevar un registro de cuántos elementos se han depositado en la papelera y cuándo desbloquear la puerta de salida.

```
1 public VRTK_SnapDropZone[] snapDropZones;
2
3 private readonly string TRASH_TAG = "Trash";
4
5 private void Start()
6 {
7     foreach (VRTK_SnapDropZone sdz in snapDropZones)
8     {
9         sdz.ObjectSnappedToDropZone += OnTrashSnapped;
10    }
11 }
12
13 internal void OnTrashSnapped(object sender,
14                               SnapDropZoneEventArgs e)
15 {
16     if (e.snappedObject.tag == TRASH_TAG)
```

```

17         trashCounter++;
18     }
19 }
```

Listing 7.4: Fragmento del script para detectar piezas de basura

En la imagen inferior, la figura 7.11, puede verse el script desde el inspector de Unity y cómo es posible definir de manera flexible y reutilizable el tamaño y los elementos del vector para que puedan ser accedidos después desde el código. Si se quisiese añadir otro objeto, bastaría con crear otra *snap drop zone* y añadirla al script, que ha sido diseñado para ser escalable.

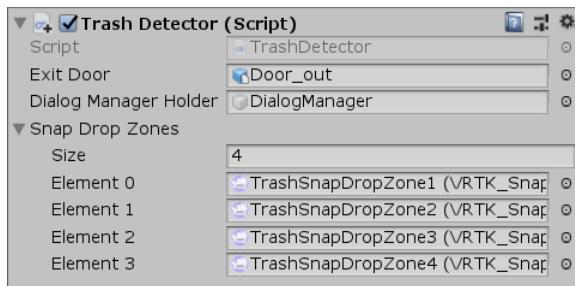


Figura 7.11: Script para detectar piezas de basura desde el inspector

7.4.2. Mostrando la descripción de los cuadros

Lo siguiente en lo que se trabajó fue en mostrar la descripción de los cuadros cuando el jugador estuviera cerca y pulsara un botón. Se dedicó bastante tiempo a pensar la solución más orgánica, desde poner al lado de cada una cuadro un pequeño marco con la descripción, como en los museos de verdad, a cambiar la textura del cuadro a un texto con su descripción, pero ninguna terminaba de ser cómoda para el jugador. Por lo tanto, finalmente se decidió superponer un cuadrado semitransparente delante de la cámara con la descripción del cuadro, que apareciese y desapareciese cuando se pulsara un botón. Además, esta solución permitía reutilizarse para el sistema de diálogos que se implementaría en un futuro, y por lo tanto ahorrar tiempo de desarrollo, por lo que se decidió elegirla. Este cuadro semitransparente sigue siempre al jugador, pero está desactivado por defecto, por lo que solo es visible cuando se activa.

En la figura 7.12 se muestra cómo funciona. Como puede verse, se define un cuadro delante de la cámara que copia su localización y rotación para seguirla mientras el jugador se mueve. De este modo siempre está a la misma distancia de la cámara. Además, se hicieron pruebas colocando el plano a

distintas distancias y con distintos tamaños de letra hasta encontrar las que se ha considerado óptimas, que permiten leer sin forzar la vista.

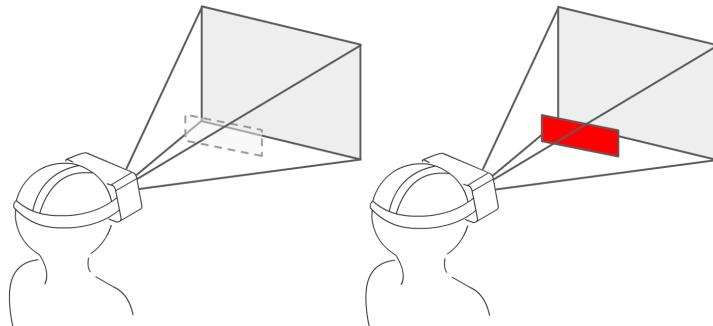


Figura 7.12: Cuadro de texto superpuesto al mundo

Además, Unity trabaja por defecto con letras rasterizadas; es decir, hace que puedan verse muy fácilmente sus píxeles. Esto, aunque en el caso de las texturas no es tan importante, hace que sea difícil leer texto, y más aún con unas gafas de realidad virtual. Por tanto, y tras investigar, se decidió utilizar la librería de Unity **TextMeshPro** que proporciona soporte para letras vectoriales, lo que hace que de igual a la distancia o el tamaño al que se rendericen, que siempre aparecerán tan nítidas como permita la resolución de la pantalla en la que se rendericen. Como único inconveniente, es necesario convertir las fuentes que se estén utilizando en el proyecto a su formato propio, aunque la propia librería proporciona una herramienta para poder hacerlo cómodamente.

Por tanto, una vez que se disponía de letras legibles que seguían al jugador, fue momento de hacer que el cuadro apareciera con la descripción de un cuadro (cargada desde un archivo de texto en disco) cuando el jugador estuviera cerca de él y pulsara un botón.

Para ello, cada cuadro tiene un script llamado `PaintingDescriptionManager.cs`, que utiliza una esfera de colisión para detectar cuándo se encuentra cerca el jugador y, cuando lo hace, activa un pequeño ícono enfrente del jugador para que éste sepa que tiene disponible la descripción de un cuadro pulsando un botón.

A continuación, cuando el jugador pulsa el botón el script se encarga de escribir su descripción, que previamente ha leído de disco, en el cuadro delante del jugador, y después lo vuelve visible para que pueda ser leído. Este cuadro vuelve a ser invisible tanto si el jugador deja de pulsar el botón como si se aleja del cuadro.

Este script necesita saber la sala en la que se encuentra y el nombre de

su cuadro para poder encontrar el directorio con el archivo que tiene que leer. También necesita las referencias a ambos mandos para saber cuándo se pulsa y libera el botón correspondiente, y las referencias a su cuadro, al ícono de disponibilidad y al objeto TextMeshPro para poder modificar su texto. De este modo, se consigue que el jugador solo tenga un cuadro de texto que es utilizado por todos los cuadros para mostrar sus descripciones.

En el listado 7.5 puede verse el fragmento de código de este script encargado de detectar si el jugador está cerca y activar el ícono de disponibilidad y, si además pulsa el botón adecuado, activará el cuadro de texto y escribirá en él su descripción.

```

1 private void OnTriggerEnter(Collider other)
2 {
3     if (other.name.Contains(PLAYER_TAG))
4     {
5         playerIsNear = true;
6         availabilityIcon.SetActive(true);
7     }
8 }
9
10 private void ControllerEvents_ButtonTwoPressed(object
11     sender, ControllerInteractionEventArgs e)
12 {
13     if (playerIsNear)
14     {
15         textObject.text = paintingDescription;
16         availabilityIcon.SetActive(false);
17         textBackground.SetActive(true);
18         PlaySoundEffect();
19 }

```

Listing 7.5: Fragmento del script para activar la descripción de los cuadros

Además, en las últimas líneas puede leerse cómo se llama al método encargado de reproducir un pequeño efecto de sonido, que no sería implementado hasta una de las últimas entregas.

7.4.3. Fundido a negro entre salas

Además, como otro de los objetivos de esta entrega era realizar un fundido a negro al cambiar de salas, se utilizó un plano negro colocado frente a la cámara de manera similar que el de la descripción de los cuadros, aunque más cerca para que no dejara nada de luz. Este plano comienza la escena completamente opaco y ve reducida su opacidad gradualmente hasta ser

completamente transparente, lo que da al jugador el efecto de un fundido desde negro. Como vuelve a ser necesario realizar la interpolación lineal de un parámetro entre dos valores a lo largo del tiempo, en este caso la opacidad, se ha utilizado la función `Lerp` de la clase `Color` para actualizar el color del plano.

7.4.4. Prototipado de un arco

Para finalizar esta entrega, la última tarea que falta es desarrollar el prototipo de un sistema de interacción avanzado que se finalizaría y perfilaría en la siguiente entrega. La idea era implementar un sistema que permitiera al jugador disparar y, aunque se valoraron varios tipos como pistolas, finalmente se decidió que el más interactivo para el jugador, además de el que más cuadraba con las temáticas del museo, sería un arco.

Para ello se utilizaron diversos componentes genéricos del framework VRTK, además de otros desarrollados específicamente para arcos, como animaciones de tensado o un script que permitía apuntar, `BowAim.cs`. Además, el modelo del arco, que también provee la librería, se adaptó para coincidir con la estética del juego.

Tras ello, se añadió un carcaj y una flecha y se utilizó el script `ArrowSpawner.cs` para hacer que el jugador pudiera sacar flechas infinitamente del mismo, lo que se hace creando múltiples instancias de la flecha original. Por último, se desarrolló un script que permitía detectar las colisiones de estas flechas con objetos de un modo similar a los explicados anteriormente.

Por lo tanto, el resultado de esta entrega fue una versión del proyecto con todas las salas completas hasta el barroco, que sería la siguiente, y un prototipo funcional de un arco listo para ser utilizado. Además, el usuario puede consultar la descripción de los cuadros y hay un fundido a negro al cambiar de escena, aparte de otras características menores. En el vídeo que se presenta a continuación pueden verse estos avances.

<https://youtu.be/MLn0r248dUs>

7.5. Entrega 4

Una vez que se disponía del prototipo funcional de un arco capaz de disparar flechas, se comenzó a modelar la siguiente sala cuyo boceto, dibujado antes de comenzar para disponer de un punto de partida, puede verse en la figura 7.13

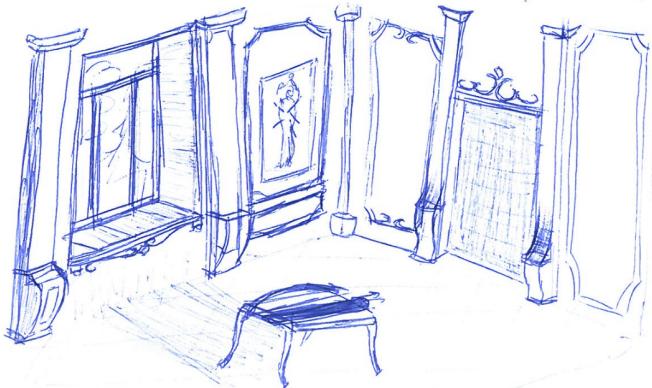


Figura 7.13: Boceto de la cuarta sala

En este boceto se propuso colocar una mesa en el centro con el arco y el carcaj encima. De este modo, sería fácil para el jugador visualizar todos los objetivos que hubiera en la sala y dispararles. Además, también pueden verse grandes ventanas que permiten que entre más luz. Aunque en el boceto no se ve, porque aún no se tenía claro, finalmente el techo también fue abierto, como puede verse en la figura 7.14.

7.5.1. Limitando el peso por texturas

Por otro lado, al comienzo del modelado de esta sala se empezó a ver que el espacio dedicado a las texturas era cada vez más y más grande. Inicialmente, para cada textura de Unity se utilizaban todos los mapas descargados, y la mayoría de ellos llegaba a una resolución de 4K, llegando a pesar cada uno varios megas, cuando no era necesaria para nada tanta resolución. Por ello, y tras comprobar que se veían bien, se decidió reducir todas las texturas a 1024x1024 píxeles y trabajar para cada textura únicamente con 3 mapas diferentes, que han sido los siguientes.

- Un mapa de color, que sería la textura propiamente dicha. En Unity, este mapa se llama **albedo**, que en Física es el porcentaje de radiación que refleja una superficie respecto a la que incide sobre ella.
- Un mapa de normales, que simula relieve en la superficie modelo y ayuda a evitar que se vea plano, aportándole realismo, sin aumentar el tiempo de renderizado.
- Un mapa de oclusión, que define cómo reflejará la luz cada parte del modelo. Esto es especialmente útil en texturas con varios materiales

juntos, como una pared pintada con trozos en los que pueden verse los ladrillos de debajo, haciendo que parezcan realmente diferentes.

Para seguir reduciendo el número de texturas, y tras comprobar que casaba relativamente bien con el color de las paredes, se decidió repetir la textura del suelo de la primera sala en ésta. La versión final de esta sala, con los cuadros colocados, puede verse en la figura 7.14.



Figura 7.14: Sala 4 renderizada en Unity

7.5.2. Disparando a los cuadros

Seguidamente, se importó el arco desde la escena de prueba en la que se desarrolló hasta la definitiva y se comenzó a trabajar en un sistema capaz de informar al usuario a qué cuadro debe disparar de manera orgánica y detectar sus disparos. Finalmente se ha implementado un sistema que hace brillar parpadeando el borde del cuadro que el jugador tiene que disparar y lo ilumina en verde completamente cuando acierta con una flecha, para volver tras unos segundos paulatinamente a su color original. Finalmente, cuando todos los cuadros han sido acertados las veces necesarias y el juego ha terminado, todos comienzan a parpadear con luz amarilla durante unos segundos, y luego vuelven a la normalidad, anunciando de este modo al jugador que ha completado la prueba.

Con el fin de dividir el trabajo entre varios scripts, se ha diseñado el siguiente sistema; por un lado, cada cuadro tiene un script, `PaintingCollider-Detector.cs`, que le permite definir todos los parámetros necesarios para cada cuadro, como el material al que tienen que cambiar, el tiempo de

fundido, si están activos o no... que se comunican con un gestor global, `ArrowGameManager.cs`, que elige qué cuadro será objetivo, y al que informa el cuadro activo cuando es disparado para que pueda llevar la cuenta de la puntuación, y finalmente activar la luz de victoria en todos los cuadros y desbloquear la puerta. En entregas futuras, este script también activaría determinados diálogos del vigilante. Además, para evitar que los cuadros estuvieran brillando cuando el jugador entrara a la sala y antes de haber tenido tiempo de hablar con el vigilante para que le explique el juego, se ha hecho que el gestor del juego sea notificado cuando el jugador toque el arco para empezar, lo que se ha conseguido con el script `BowInformer.cs`, que utiliza una esfera de colisión en el mango para saber cuándo el jugador lo sostiene por primera vez. Además, el arco también contiene el script para teletransportarse de vuelta a la mesa si toca el suelo.

A continuación, la figura 7.15 contiene el diagrama de secuencia que intenta presentar de un modo más visual lo que se ha explicado anteriormente. Aunque en Unity la vida de un objeto es más amplia, ya que se inician cuando la escena comienza y se destruyen cuando acaba, se ha adaptado para representar de manera más intuitiva cuándo interviene cada clase.

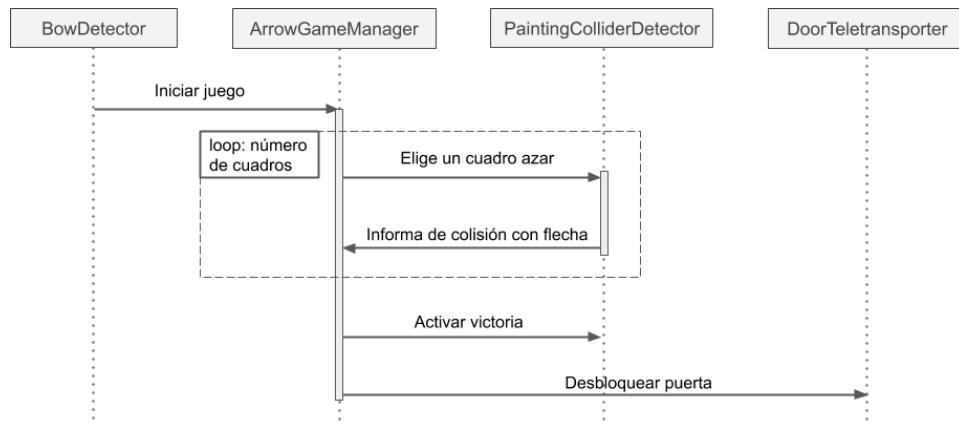


Figura 7.15: Diagrama de secuencia del juego del arco

Para gestionar el estado de un cuadro, que puede ser `Disabled`, `Enabled`, `FadingToDisabled` y `Winning`, se ha utilizado una máquina de estado con `switch` dentro del método `Update()` que comprueba en cada frame su estado y actúa en consecuencia. Para cambiar su material, de nuevo, se ha utilizado la función `Lerp` de interpolación lineal.

En el listado 7.6 puede verse cómo funciona el `switch` mencionado anteriormente, cambiando su comportamiento dependiendo del estado, que es actualizado de forma asíncrona por el gestor del juego, y cómo se informa al gestor del juego cuando una flecha colisiona con un cuadro cuando éste está

activo.

```
1 private void Update()
2 {
3     switch (state)
4     {
5         case PaintingState.Disabled:
6             break;
7
8         case PaintingState.Enabled:
9             pingpong = Mathf.PingPong(Time.time,
10                             FadingInDuration);
11            fadingLerp = pingpong / FadingInDuration;
12            GetComponent<Renderer>().materials[1].Lerp(
13                LightMaterial,
14                FrameMaterial,
15                fadingLerp);
16            break;
17
18         case PaintingState.Winning:
19             ...
20     }
21 }
22
23 private void OnTriggerEnter(Collider other)
24 {
25     if (state != PaintingState.Disabled &&
26         other.tag == ARROW_TAG)
27     {
28         gameManagerScript.
29         AnnounceDetectedArrow(this.name);
30     }
31 }
```

Listing 7.6: Fragmento del script para actualizar un cuadro

Como puede verse, al colisionar con un objeto que resulta ser una flecha se anuncia al gestor del juego lo ocurrido para que se encargue de ello. En el listado 7.7 puede verse este método y cómo comprueba cuántos cuadros se han acertado y si quedan más. De ser así, elige un cuadro aleatoriamente (que no puede ser el mismo) y le hace saber que es el elegido. y éste actualiza su estado; por el contrario, si el juego ha terminado, lo anuncia a todos los cuadros para que ellos lo gestionen y parpadeen en amarillo durante unos instantes.

```
1 public void AnnounceDetectedArrow(string painting_name)
2 {
3     painting_scripts[currentIndex].Disable();
```

```

4     successfulShots++;
5
6     if (successfulShots >= MaxShoots)
7     {
8         foreach (PaintingColliderDetector pcd
9                 in painting_scripts)
10        {
11            pcd.Win();
12        }
13    }
14    else
15    {
16        currentIndex = GenerateNewIndex();
17        painting_scripts[currentIndex].Enable();
18    }
19 }
```

Listing 7.7: Fragmento del script para gestionar el juego del arco

7.5.3. Mostrando un candado en las puertas bloqueadas

Por otro lado, al empezar a probar el proyecto con jugadores reales se vio que no había ningún tipo de retroalimentación en el momento en el que la puerta se desbloqueaba; es decir, el jugador no sabía cuando podía pasar a la siguiente sala ni qué puertas estaban bloqueadas. Por ello, se situó la imagen de un candado delante de las puertas bloqueadas que, gestionado por el propio script que activa y desactiva el teletransporte, es visible cuando el jugador no puede utilizarla y es deshabilitado y desaparece cuando la puerta es usable.

Por último, antes de pasar a la siguiente entrega, se comenzó a trabajar en la siguiente sala, la quinta, ambientada en el romanticismo. Como tal, se intentó dar un aspecto místico utilizando poca luz y un sistema de partículas de niebla, aunque éste no fue implementado hasta la siguiente entrega.

Como tanto la cuarta sala, con el arco y las flechas, como la sexta, que previsiblemente implementaría un juego de piezas de cuadros en forma de puzzle, se consideraban bastante activas, se decidió diseñar la quinta para que fuera más tranquila y el jugador pudiera relajarse. Por tanto, se pensó en esconder la llave para desbloquear la puerta de salida en los cajones de un mueble y colocar varios cuadros con pistas escondidas que llevarán al jugador hasta la llave, obligándole así a examinarlos con detalle.

Así, el boceto generado para esta sala puede verse en la figura 7.16, que contiene el mueble con los cajones (y un par de sillas para no hacerlo tan

evidente), una alfombra que finalmente se descartó y unas gruesas cortinas que tapan la luz del exterior.



Figura 7.16: Boceto de la quinta sala

Pero apenas se comenzó a modelarla cuando llegó la fecha de finalización de la entrega, por lo que se paró de trabajar en la misma y se grabó el vídeo mostrando el resultado hasta la fecha, que incluía las cuatro primeras salas terminadas y pequeños añadidos como los candados de las puerta. Dicho vídeo puede verse a continuación.

<https://youtu.be/w83YTZuz6tQ>

7.6. Entrega 5

Tras dar por finalizada la entrega 4, la primera tarea en la que comenzó a trabajar fue el terminar el modelado de la quinta sala. Como ya se ha comentado antes, finalmente la alfombra de en medio se eliminó y fue sustituida por la estatua de *El pensador* de Auguste Rodin, además de situar los cajones al fondo para intentar ocultarlos.

7.6.1. Niebla virtual

Para generar niebla realista se han utilizado dos componentes simultáneamente. Por un lado, se ha utilizado un sistema de partículas que, a partir

de un *tileset* o conjunto de imágenes de humo, consigue generar imágenes translúcidas y en movimiento a lo largo de la sala. Además, se ha configurado este sistema para que roten y cambien de dirección ligera y aleatoriamente, dando la sensación de bruma cambiante.

Por otro, se ha utilizado el sistema de iluminado para simular lo que se conoce como *deferred fog*² o niebla diferida. Este efecto superpone un color a los objetos dependiendo de lo lejos que estén de la cámara; así, si un objeto está cerca se verá tal cual pero si se aleja se irá volviendo poco a poco del color indicado. En este caso el color elegido ha sido gris, lo que junto al sistema de partículas simula de un modo lo suficientemente realista el efecto óptico que genera la niebla real.

El resultado final de esta sala puede verse en la figura 7.17.

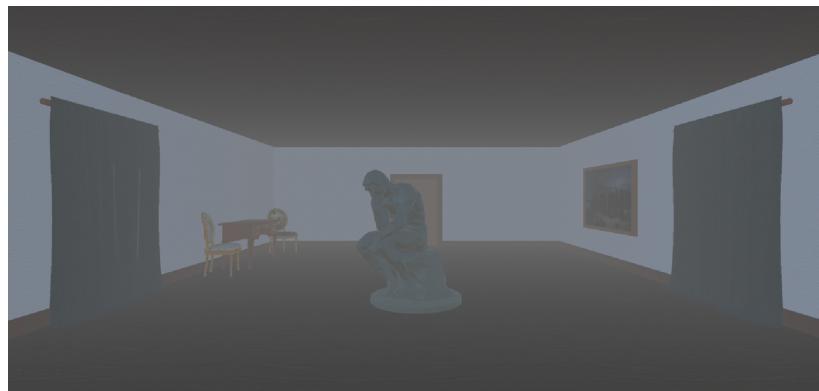


Figura 7.17: Sala 5 renderizada en Unity

7.6.2. Cuadros con pistas y cajones interactivos

Una vez que la sala estaba terminada se pasó a implementar el acertijo elegido. Como ya se introdujo anteriormente, la quisicosa para esta sala sería un cajón interactivo escondiendo la llave de la puerta de salida y las sílabas de la palabra *cajones* en cada uno de los cuadros. Por ello, en cada uno de ellos se colocó un cuadro de texto con la librería TextMeshPro para que pudieran leerse siempre nítidamente. El cuadro *Doña Juana la Loca* contiene la sílaba *CA* cerca del ataúd de Felipe el Hermoso, el cuadro *El caminante sobre un mar de nubes* contiene la sílaba *JO* en el banco de niebla central y por último el cuadro *El Fusilamiento de Torrijos y sus compañeros* contiene la sílaba *NES* cerca del general. Además, cada uno de los textos tiene un

²<https://docs.unity3d.com/Manual/PostProcessing-Fog.html>

color muy parecido a la parte que sobre la que están colocados para que no sean tan fáciles de ver.

Para terminar con esta sala, se procedió a implementar los cajones interactivos. Para ello, tras modelar una pequeña mesa con cinco cajones y una llave antigua en Blender, se importaron a Unity y se utilizó el script `VRTK_Physics_Slider.cs` para hacer que los cajones se pudieran deslizar en una dirección y se colocó la llave dentro de uno de ellos. El problema con este script fue que lo que en teoría debería haber sido algo relativamente directo se convirtió en un interminable ajuste de parámetros para conseguir que el sistema funcionara correctamente, e incluso fue necesario bucear en el código fuente del framework para averiguar por qué no lo hacía. Finalmente fue necesario escribir un pequeño script, llamado `CollisionIgnorer.cs`, para hacer que la llave ignorara la caja de colisión con la mesa. Como puede verse en el listado 7.8, este script hace uso de la interfaz al motor de físicas de Unity y comprueba cuando colisiona con un objeto si corresponde con el que se ha indicado que ignore para indicarle que lo haga.

```
1 public GameObject ignoredObject;
2
3 void OnCollisionEnter(Collision collision)
4 {
5     if (collision.gameObject.name == ignoredObject.name)
6     {
7         Physics.IgnoreCollision(
8             collision.collider,
9             GetComponent<Collider>());
10    }
11 }
```

Listing 7.8: Fragmento del script para ignorar colisiones

Por último, para hacer que el jugador pudiese colocar la llave en la cerradura de la puerta, que no era más que una imagen como se explica en la entrega anterior, se utilizó una *snap drop zone* estratégicamente colocada en el ojo de la cerradura para que pareciese que realmente estaba activando su mecanismo, y como su uso ya ha sido explicado anteriormente esta vez se omitirá. Por último, para detectar cuando la llave es colocada se ha desarrollado el script `KeyDetector.cs` que se suscribe al evento que emite el objeto `VRTK_SnapDropZone` de la cerradura cuando se le coloca la llave, comprobando que se trata de ella y comunicándose al script de la puerta para que oculte el candado y se active para permitir al usuario acceder a la siguiente sala.

7.6.3. Descripción de la sala de las vanguardias

Así, una vez que se terminó la quinta sala, se empezó a trabajar en la sexta y, como en el resto, se comenzó dibujando su boceto, que puede verse en la figura 7.18. Como se ha introducido anteriormente, en esta sala hay cuatro cuadros, divididos en cuatro piezas cada uno, que el jugador tiene que colocar en su sitio para poder avanzar a la siguiente sala. En cada pared, además del hueco para las piezas y una pequeña miniatura del cuadro que el jugador puede usar como referencia, hay una bombilla que indica mediante su color si el cuadro está sin terminar, brillando en amarillo, si el cuadro se ha terminado pero mal, brillando en rojo, y si el cuadro es correcto, con lo que brillará en verde. El utilizar una bombilla que cambie de color es la interfaz más natural con el jugador y la manera más orgánica de proporcionarle retroalimentación que se ha valorado.

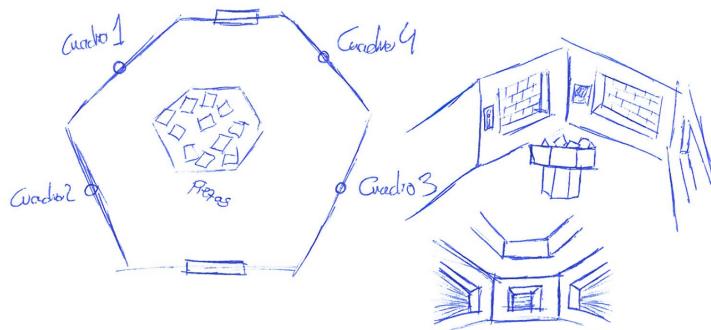


Figura 7.18: Boceto de la sexta sala

Como está ambientada en las vanguardias del siglo XX, se ha intentado dar un aspecto más moderno a esta sala modelándola con amplias paredes blancas con forma hexagonal, lo que coincide con el tener cuatro cuadros, uno en cada pared, más dos para las puertas de entrada y salida. Además, también se ha utilizado una textura de azulejos con forma pseudo-hexagonal para el suelo. Además, en el centro de la sala se encuentra un pequeño recipiente con las 12 piezas de los cuadros que el jugador debe coger para colocarlas.

Por otro lado, encima de la puerta de salida que, como indicará el vigilante, lleva a su almacén (en el que es finalmente desenmascarado como el ladrón), se ha colocado un pequeño letrero en el que puede leerse *Staff*.

La imagen con la versión final de la sala puede verse en la figura 7.19 con todos los elementos explicados anteriormente. Aunque en el juego evidentemente las piezas no pueden aparecer puestas y en el contenedor del centro

a la vez, en la imagen se muestran ambas para que el lector pueda hacerse una idea de cómo quedan tanto colocadas como antes de hacerlo.



Figura 7.19: Sala 6 renderizada en Unity

7.6.4. Puzzles virtuales con piezas de cuadros

Una vez explicado el modelado de la sala, se pasará a contar cómo se ha implementado el sistema de puzzles con las piezas. Lo primero que es necesario explicar es que, al igual que en todos los casos anteriores en los que se requería que el usuario colocase objetos en lugares específicos, se ha usado la combinación de objetos interactivos (en este caso las piezas de los cuadros) con *snap drop zones* (los huecos en los que es necesario colocarlos).

Para implementar este sistema se ha utilizado un diseño *master/slave* parecido al del arco que dispara a cuadros, en el que hay un único componente que sabe lo que está pasando globalmente encargado de controlar al resto, que simplemente le informan de lo que pasa y actúan cuando y como se lo ordena.

En este caso, los *slaves* son las cuatro instancias del script `Painting-Checker.cs`, una por cada cuadro. Este script es el encargado de manejar las cuatro *snap drop zones* de las que se compone un cuadro (una para cada pieza), detectando cuándo se coloca y se retira una pieza. Además, como la nomenclatura que se ha seguido para las piezas y las zonas ha sido llamarlas en ambos casos `Cuadro_Pieza` es posible para el script saber si una pieza ha sido colocada en su lugar correcto siempre que ambos nombres coincidan. Para detectar qué número contiene el nombre se han utilizado **expresiones regulares**, concretamente, el método `Match` de la clase `Regex`, que indicándole qué patrón ha de seguir la expresión que buscamos es capaz de devolver la cadena que coincide.

Para este puzzle ha sido de vital importancia colocar las *snap drop zones* justo en el lugar donde corresponden, para que cuando el cuadro esté completamente no se noten los cortes entre ellas. Esta tarea ha sido relativamente tediosa y ha sido necesario apuntar manualmente las posiciones exactas desde Blender y hacer el cálculo de cambio de ejes a Unity, ya que los ejes X, Y, Z en Blender corresponden a X, Z, -Y en Unity.

En la figura 7.20 puede verse cómo se ilumina una zona al pasar el cuadro por encima, que por defecto es invisible; de este modo es posible para el jugador saber en qué sitio está colocando la pieza, que al soltarla se coloca automáticamente en su sitio.



Figura 7.20: Pieza de un cuadro sobre uno de los posibles huecos

Como puede verse, la pieza es tres veces menor que la zona, y una vez puesta triplicará su tamaño. Esto se ha hecho así porque al principio del diseño se vio que las piezas del cuadro eran demasiado grandes para ser manejadas con comodidad, lo que no era posible hasta que medían aproximadamente un tercio de su tamaño original. Así, se ha configurado la zona para que al colocar un objeto sobre ella que sea del tipo esperado haga aumentar su tamaño tres veces.

Para poder detectar cuándo se colocan objetos, el script suscribe la misma función al evento de todas sus zonas que informan de éste hecho. Aunque se podría haber suscrito un método a cada uno y se sabría de antemano qué zona es la que lo ha emitido, este diseño es mucho más compacto y reutilizable e implica menos duplicidad de código.

A continuación, en el listado 7.9 puede verse cómo cuando se coloca una pieza en una de las cuatro *snap drop zones* que gestiona cada script se llama al siguiente método, que comprueba, primero, que el cuadro no esté ya terminado. Si no lo está, aumenta en uno el número de piezas que tiene y, si la pieza ha sido colocada en su sitio, aumenta en uno el número de piezas correctas, pero si no y si está lleno enciende la luz roja. Por el contrario, si todas las piezas han sido colocadas correctamente el cuadro ha sido terminado, por lo que cambia la bombilla a verde, deshabilita las cajas

de colisión de todas las piezas para que no puedan ser retiradas e informa al *master* de que el cuadro ha sido terminado correctamente.

```
1 internal void OnPieceSnapped(object sender,
2                               SnapDropZoneEventArgs e)
3 {
4     if (!isCorrectlyFinished)
5     {
6         currentPieces++;
7
8         string objectName = e.snappedObject.name;
9         string zoneName = sender.ToString();
10
11        if (PieceIsCorrect(objectName, zoneName))
12        {
13            correctPieces++;
14            if (correctPieces == MAX_PIECES)
15            {
16                isCorrectlyFinished = true;
17
18                ChangeBulbColor(LightGreen);
19                LockAllPiecesInPlace();
20
21                gameManagerScript.PaintingFinished();
22            }
23        }
24
25        else if (currentPieces == MAX_PIECES
26                  && !isCorrectlyFinished)
27        {
28            ChangeBulbColor(LightRed);
29        }
30    }
31 }
```

Listing 7.9: Fragmento del script detectar piezas de cuadros

Por otro lado, el *master* del sistema es el script `PaintingPiecesGameManager.cs`. Este componente es informado de los eventos que suceden y actúa en consecuencia. Además, de manera similar a la sala del arco, cuando el jugador entra en esta sala todas las bombillas están apagadas y no se encienden hasta que el jugador se acerca al contenedor central. Este recipiente tiene un script, `GameStarter.cs`, que informa al *master*, quien a su vez le indica a todos los *slaves* que enciendan las luces. Finalmente, cuando ha recibido un número de mensajes de completitud igual al tamaño de su vector de *slaves*, puede saber que todos los cuadros han sido resueltos correctamente y desbloquea la puerta de salida para que el jugador pueda

continuar.

En entregas posteriores se integraría el gestor de diálogos del vigilante para que hablara cada vez que un cuadro era completado.

Finalmente, al terminar la entrega, y como en el resto de los casos, se grabó un vídeo demostrativo explicando todos los cambios implementados a lo largo de la misma, que puede consultarse en el siguiente enlace.

<https://youtu.be/WDkoSrzh24o>

7.7. Entrega 6

Lo primero que se comenzó a hacer en la entrega 6 fue modelar la séptima sala, el almacén del vigilante, en la que el juego llega a su desenlace y se desenmascara finalmente al ladrón del cuadro.

7.7.1. El almacén del vigilante y el cuadro robado

Para el modelo de esta sala, como en comparación con el resto es más simple, no fue necesario realizar un boceto y se pasó directamente a modelarla. Es una habitación cuadrada, relativamente pequeña, para la que se ha utilizado una textura de pintura desconchada para las paredes y de cemento para el suelo para que, en contraposición con el resto del museo, parezca vieja y abandonada, en un intento de destapar al vigilante como alguien malvado. Además de las estanterías, la sala cuenta con varias cajas y palés, algo que temáticamente encajaría en un almacén, que puede verse terminada en la figura 7.21.



Figura 7.21: Sala 7 renderizada en Unity

Además, para dar más realismo a la sala, se han colocado diversos objetos temáticos, como cajas, un cubo de fregar, un limpiacristales o a Suzanne, el icónico logo de Blender con forma de cabeza de mono. Además, el jugador puede interactuar con todos ellos.

En el centro de la sala, encima de una silla y tapado por diversos objetos como cajas y cepillos que el jugador tendrá que quitar de en medio para llegar hasta él, se encuentra el cuadro robado. Se ha dedicado bastante tiempo en pensar qué cuadro elegir como el robado, si debía ser real o ficticio, y finalmente se ha inventado a un pintor, Banksy (un juego de palabras con el pintor Banksy), quien resulta ser el artista más famoso del mundo y cuyas obras valen millones de euros. Este pintor es el autor del cuadro *Universo* que fue robado por el vigilante y que, como confesará más adelante, lo hizo porque él es el único que sabe reconocer el verdadero valor de la pieza, por lo que es el único que merece tenerlo, haciéndose ver como alguien egocéntrico y megalómano.

Este cuadro, en una referencia al arte conceptual, ha sido adaptado de una imagen de una mancha de vino con licencia libre a la que se le han aplicado diversos filtros en GIMP, como desaturación, posterización o el filtro artístico *oilify*. La versión final de este cuadro puede verse en la figura 7.22.



Figura 7.22: Imagen del cuadro robado

Antes de pasar a la siguiente sala, se implementó un pequeño script que detectaba cuándo el jugador interactuaba con los elementos que tapaban el cuadro, ya que debe retirarlos para llegar hasta él, y cuándo lo cogía. Este script sería más tarde integrado con el gestor de diálogos para que el vigilante hablara con el jugador mientras llegaba poco a poco su fin. Este script será descrito más adelante.

7.7.2. Sala final y desenlace

Para no terminar el juego tan abruptamente se decidió diseñar una última sala, con elementos como coches de policía, en la que el jugador tuviera

un sentimiento de desenlace y de que el vigilante había sido finalmente detenido.

Para ello se decidió adaptar la antesala del museo, en la que había empezado todo, y que habría cambiando para tener ahora diversos elementos relacionados con la policía. Aunque inicialmente se pensó que utilizar la misma escena de Unity que la antesala, finalmente se decidió que la sala final tenía la suficientemente autonomía e independencia como para ser una escena en sí misma.

En esta versión de la sala es de noche, para hacer ver al jugador que había pasado todo el día investigando para descubrir al ladrón, y hay varias patrullas con las luces encendidas aparcadas fuera, que pueden verse desde los ventanales. Además, las luces del pasillo están encendidas y hay conos, balizas y una cinta de policía que impide físicamente la entrada de vuelta al museo. La sala terminada puede verse en la figura 7.23.



Figura 7.23: Sala 8 renderizada en Unity

Para el modelado de los coches de policía se ha usado el paquete de modelos de Unity *Low Poly Police Car Pack*³, que puede ser descargado gratuitamente desde su tienda con licencia libre. Este paquete incluye varios modelos *low poly* de coches de policía, lo que permite probarlos rápidamente en la escena y ver cuál queda mejor.

En esta sala, cuando el gestor de diálogos estuviera implementado, se recibiría una última llamada de la directora del museo agradeciéndole su trabajo y despidiéndose hasta la siguiente vez, abriendo la posibilidad a una segunda entrega del juego.

³<https://assetstore.unity.com/packages/3d/vehicles/land/low-poly-police-car-pack-59458>

7.7.3. Modelo del vigilante y animaciones

Como el vigilante juega un papel primordial en la narrativa del juego y su interacción con el jugador era constante, era necesario contar con una representación visual del mismo. Inicialmente se barajaron opciones mucho más simples, como utilizar una imagen bidimensional con un dibujo del mismo, totalmente estático, que simplemente estuviera en su sitio pero, tras probarlo, el resultado no fue para nada convincente y fue evidente que había que utilizar un modelo tridimensional para el mismo.

Al principio se buscaron modelos de ancianos con aspecto confiable y bonachón, para ocultar la cleptomanía del vigilante, pero aunque se encontraron algunas opciones perfectas, como los que se pueden ver en la figura 7.24, por *3Dcartoon* (izquierda) y *danielmgt* (derecha) costaban entre 80 y 120 euros, y como uno de los objetivos no funcionales de este proyecto era utilizar modelos gratuitos con licencia libre no fue posible usarlos.



Figura 7.24: Algunos de los modelos que se valoraron para vigilante

En su lugar, y tas mucho buscar, se decidió utilizar el modelo de un hombre *low poly* con licencia libre de Denys Almaral⁴, que se adaptó en Blender para hacerle parecer mayor, simplificar su topología y añadir elementos como la placa que tiene en el pecho, ajustar su esqueleto y cambiar el color de su ropa. Este modelo finalizado e importado ya ha podido verse en algunas de las salas anteriormente mostradas.

Tras ello, se importó a Unity se probó, pero era simplemente un modelo estático, lo que lejos de aportar realismo al juego se lo quitaba. Por ello, se

⁴<https://www.turbosquid.com/3d-models/3d-style-couple-casual-man-model-1387761>

decidió añadirle animaciones.

Unity está bastante bien preparado para utilizar animaciones; tiene integrada una herramienta capaz de importar y normalizar automáticamente esqueletos tridimensionales desde un modelo, utilizando una nomenclatura constante para los huesos. De este modo, es posible importar animaciones ya hechas al juego y aplicarlas a un modelo con un esqueleto normalizado para que, con relativamente poca configuración, sea posible utilizarlas con un resultado muy bueno.

Inicialmente se intentó implementar animaciones simples para el vigilante, pero mis habilidades como animador dejan mucho que desear y el resultado no era para nada convincente, por lo que se decidió importarlas ya hechas. El problema de las animaciones universales de Unity es que hay un mercado muy importante a su alrededor, y todos los paquetes medianamente completos que se encontraron eran de pago, llegando a costar cientos de euros. Por ello, ha sido necesario utilizar varios ejemplos de prueba con licencia libre, cada uno conteniendo una de las animaciones buscadas, como el paquete *Idle MoCap* de Morro Motion para el estado de reposo del vigilante, o el paquete *Basic Motion* de 3D-Brothers para el saludo y la negación.

Una vez que se importaron y se comprobó que funcionaban correctamente, se integraron en el juego. Para ello, lo primero que se hizo fue añadir un componente *Animator* al objeto del vigilante, que relacionaba su esqueleto o *avatar* en Unity con un **controlador de animaciones**.

Este controlador implementa una máquina de estados, asignando una animación a cada uno de ellos, y permite gestionar de un modo intuitivo las transiciones entre ellos. Esto puede verse en la figura 7.25, que presenta en controlador de animaciones final de guarda, y en la que se ha seleccionado la transición desde el estado *Greeting* hasta *Idle*. A su derecha aparece la configuración de la transición, en la que podemos decidir cuánto durará la transición, en qué momento ocurrirá y qué **condiciones** han de darse.

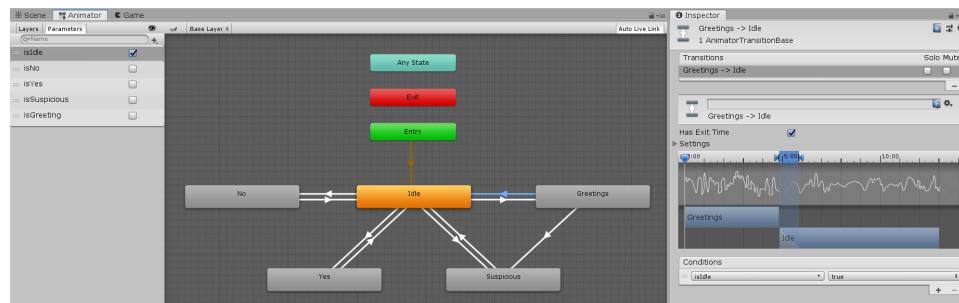


Figura 7.25: Máquina de estados del controlador de animaciones

Estas condiciones pueden configurarse a través de variables desde el propio editor, como puede verse a su derecha. En este caso, se ha asignado una variable booleana a cada transición, lo que permite activarlas cuando la variable cambia. En este caso el estado por defecto, en naranja, es *Idle*, así que cuando la variable `isGreeting` sea *true* la transición de estado desde *Idle* hacia *Greeting* se activará. Estas variables pueden modificarse desde el código fuente, lo que hace el controlador de animaciones una herramienta muy potente.

Estas animaciones están estrechamente ligadas con el sistema de diálogos; de hecho, es el propio gestor de diálogos el encargado de activar las animaciones. Por tanto, se explicará con más detalle en la siguiente sección.

7.7.4. Implementación de un sistema de diálogos

Como el guardia debía disponer de algún método de comunicar información al jugador, se ha diseñado un sistema de diálogos para ello.

Como en el caso de las descripciones de los cuadros, y como ha podido verse en la figura 7.12, se ha utilizado un cuadro de texto superpuesto al mundo para que el jugador pueda leer lo que el guarda le dice. Además, como se explica más adelante, se ha añadido un efecto de sonido para que se reproduzca cuando habla estilo *8 bit*, similar a los que utilizaban los videojuegos antiguos, para hacerlo más inmersivo.

Como controlar el estado en el que se encontraba el guarda era muy complejo y abstracto, se ha implementado una máquina de estados para ello, que se aprovecha para controla la animación que está reproduciendo el vigilante. El diagrama de la misma aparece en la figura 7.26, y como puede verse hay 5 estados en total.

Al empezar el juego, el vigilante está callado y con la animación *idle*, pero cuando el jugador se acerca a él (hecho que se detecta con una esfera de colisión), cambia al estado **SPEAKING** y empieza a hablar con él. Cuando cada uno de los diálogos termina, cambia de estado a **WAITING_INPUT** y se queda callado esperando la interacción del jugador con el mando, y cuando pulsa empieza el siguiente diálogo. Sin embargo, cuando el jugador se aleja el guardia se callará hasta que vuelva a acercarse, momento en el que comenzará de nuevo el diálogo por el que iba. Cuando ha terminado todos los archivos de diálogo, pasa al estado **FINISHED**, en el que permanece hasta que la escena termina. A menos, claro, que se active algún diálogo especial, como una felicitación o una reprimenda por fallar alguna prueba, momento en el que volvería al estado de hablar.

Para los archivos de diálogo se ha utilizado la siguiente nomenclatura;

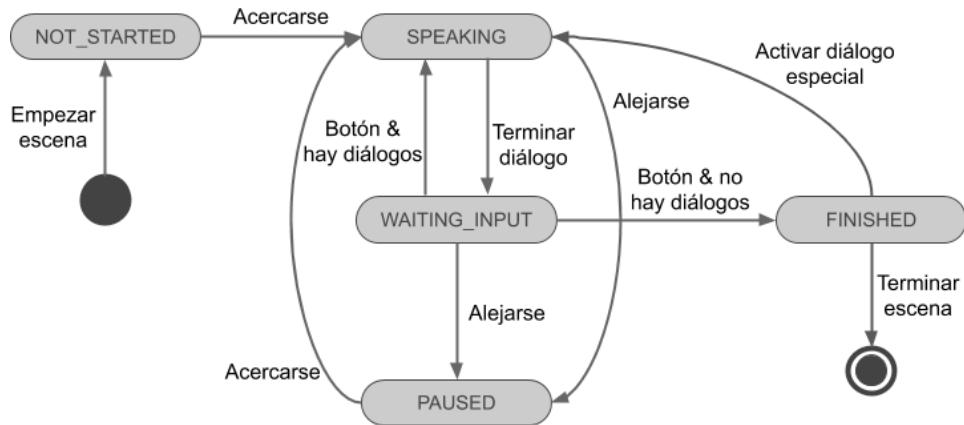


Figura 7.26: Diagrama Máquina de diálogos del gestor de diálogos

desde el punto de vista de la implementación, cada sala tiene una directorio en la que se encuentran sus diálogos, que comienzan por el numero 0, (e.g. 0.txt), y aumenta una unidad para denotar el siguiente. Cuando no hay más archivos de texto, el gestor de diálogos lo da por terminado y pasa al estado FINISHED.

Además, se han añadido dos diálogos con la directora del museo, que es la que llama al detective para pedirle su ayuda inicialmente. El primero sucede cuando el jugador entra en la antesala, con un retraso de 4 segundos, para que no sea tan inmediato, en la que le explica la situación y le indica que hable con el vigilante. El segundo ocurre en la sala final, con el mismo retraso, en el que le agradece la ayuda prestada. En ese momento, cuando el jugador termina de hablar con ella se desbloquea la puerta de salida del museo, y el jugador vuelve al menú principal.

Por otro lado, para no hacer que todo el texto apareciese de forma inmediata, se ha hecho que cada letra del texto aparezca cada dos décimas de segundo, intentando imitar la velocidad de un diálogo real. Para ello, y como puede verse en el listado 7.10, cada frame del juego se comprueba si el vigilante está hablando, y si sí se comprueba si ha pasado suficientemente tiempo como para mostrar otra letra. Si ya se han mostrado todas, se activa el estado de espera y se muestra un icono para que el jugador sepa que tiene que pulsar el botón correspondiente.

```

1 private void Update()
2 {
3     if (speaking == SpeakingState.SPEAKING)
4     {

```

```

5         lastDeltaTime += Time.deltaTime;
6         if (lastDeltaTime > TimeBetweenLetters)
7         {
8             if (currentLetterIndex <
9                 currentDialog.Length)
10            {
11                DialogObject.text =
12                    currentDialog.Substring(0,
13                     currentLetterIndex);
14                currentLetterIndex++;
15                lastDeltaTime = 0f;
16            }
17            else
18            {
19                speaking = SpeakingState.WAITING_INPUT;
20                NextDialogIcon.SetActive(true);
21            }
22        }

```

Listing 7.10: Fragmento del script del gestor de diálogos

Por otro lado, para ejemplificar la integración con el controlador de animaciones con el gestor de diálogos, en el listado 7.11 se muestra cómo, desde el mismo script que anteriormente, se activa la animación de vigilante sospechoso, lo que se hace en la última y penúltima sala. En él, y como se ha explicado anteriormente, se utilizan las variables ligadas a las transiciones entre estados para activar determinadas animaciones. Uno de los problemas que se ha encontrado es que si simplemente se modifica la variable, la animación actual ha de terminar para que se active su transición; en cambio, si se obliga a reproducirla, como puede verse en la línea 6, cambia automáticamente y sigue respetando las variables para decidir qué hacer cuando dicha animación termine.

```

1 public void ActivateSuspiciousAnimation()
2 {
3     Animator animator = Guard.GetComponent<Animator>();
4     if (animator)
5     {
6         animator.CrossFade("Suspicious", 0.5f);
7         animator.SetBool("isIdle", false);
8         animator.SetBool("isSuspicious", true);
9     }
10 }

```

Listing 7.11: Fragmento del script para activar animaciones

Este script, `DialogManager.cs`, ha resultado ser bastante extenso y complejo, y aunque tiene otras muchas partes interesantes, como la gestión de la interacción del jugador con el mando o de su acercamiento al vigilante dependiendo del estado, se deja al lector el consultar.

El resto del trabajo de los diálogos fue de índole narrativa para diseñar todos los diálogos a lo largo de las salas y los de confesión de vigilante, por lo que se omitirán. Si el lector los quiere consultar, se encuentran en la ruta `Assets/Text/Dialogs/spanish` del proyecto.

7.7.5. Menú principal del juego

Una vez que el proyecto estaba prácticamente completo, se procedió a diseñar e implementar el menú principal del juego. Aunque inicialmente se pensó en hacer un menú 2D que no aprovechara el potencial de la VR, al investigar se descubrió que el framework VRTK soportaba los botones nativos de Unity, por lo que se decidió implementar un menú virtual.

Para ello, se ha tomado como referencia el menú principal del juego *Beat Saber*⁵, en el que el jugador se encuentra en un punto sin poder desplazarse, aunque puede mover la cabeza, y utiliza láseres que salen desde sus mandos para señalar los botones que quieren pulsar, pulsando un botón para activar su selección. En un intento por lograr que hasta el menú del juego sea inmersivo, como en el caso del *Beat Saber*, se ha decidido modelar una pequeña ciudad con edificios *low poly* en la que el jugador se encuentra en el medio y desde la que ve tanto el propio museo desde fuera como varios botones tridimensionales con los que puede interactuar, como puede verse en la figura 7.27. Estos botones le permiten empezar una nueva partida, ver los créditos del juego o salir.

Como el modelar edificios es una tarea trivial, y el tiempo de la entrega estaba llegando a su fin, se decidió buscar paquetes de modelos que pudieran ser utilizados para este fin, y el paquete *Low poly European City Pack* de *karboosx*, que contiene 8 edificios diferentes que coinciden con la estética del juego.

Después se colocaron los botones y el logo del juego y se limitó la funcionalidad de los mandos para que el jugador no pudiera moverse con ellos y se utilizaron y configuraron los componentes `VRTK_UI_Pointer.cs` y `VRTK_Straight_Pointer_Renderer.cs` del framework para hacer que el jugador pudiera utilizar los mandos para seleccionar elementos.

Tras ello, se creó el script `MenuManager.cs` desde el que se añadió un

⁵<https://beatsaber.com/>



Figura 7.27: Menú principal del juego

listener diferente a cada botón para poder controlar su funcionamiento. En el listado 7.12 se muestra, en un fragmento de este script, cómo se añade un listener al botón de salir y cómo este script comprueba si se trata del editor de Unity o del juego compilado, es decir, si estamos desarrollando o jugando, para decidir cómo cerrarse.

```

1 private void Start()
2 {
3     if (exitButton != null)
4     {
5         exitButton.onClick.AddListener(OnClickExit);
6     }
7 }
8
9 private void OnClickExit()
10 {
11     PlaySoundEffect();
12     #if UNITY_EDITOR
13         UnityEditor.EditorApplication.isPlaying = false;
14     #else
15         Application.Quit();
16     #endif
17 }
```

Listing 7.12: Fragmento del script gestionar el menú

7.7.6. Música de fondo y efectos de sonido

Por último, como el juego quedaba muy vacío y no había verdadera retroalimentación entre el jugador y su interacción, se decidió integrar música

y efectos de sonido.

Para toda la música se ha utilizado la página *SoundImage*⁶, un sitio web mantenido por donaciones que proporciona música gratuita y con licencia libre especialmente pensada para videojuegos. A continuación se presenta una lista con la canción elegida para cada sala.

- **Menú.** *Magic Clock Shop.*
- **Antesala y sala 1.** *Peaceful Evening.*
- **Sala 2.** *Puzzle Game.*
- **Sala 3.** *Clippity Clop.*
- **Sala 4.** *Gentle closure.*
- **Sala 5.** *Peaceful Mind.*
- **Sala 6.** *Winding Down.*
- **Sala 7.** *Thumbing Across Alaska.*
- **Sala 8.** *The Jazz man.*

Además, para la música de la última sala, se ha utilizado Audacity para solapar sonido de sirenas de policía, ya que en esa sala hay varias patrullas, como ha podido verse en la figura 7.23.

Para toda la música de fondo se ha usado el componente nativo de Unity *Audio Source*, al que basta indicarle la canción que queremos reproducir, si ha de hacerlo en bucle y su volumen para hacerlo.

Por otro lado, se han añadido cuatro efectos de sonido distintos que se enumeran a continuación.

- Cuando el jugador selecciona un elemento del menú, suena un pequeño click, añadiendo retroalimentación.
- Cuando alguien está hablando con el jugador, ya sea la directora o el vigilante, suena un efecto de sonido estilo *8 bit*. Este efecto se ha modificado en Audacity para hacerlo más grave en el caso del vigilante y más agudo en el caso de la directora, imitando la diferencia de tonos que suele haber entre voces de hombres y mujeres.

⁶<https://soundimage.org/>

- Cuando la directora llama al jugador, ya que hablan siempre por teléfono, suena el timbre clásico de un teléfono.
- Por último, cuando el jugador activa la descripción de un cuadro suena un sonido de papel arrugado.

Como los efectos de sonido se reproducen en situaciones concretas, ha sido necesario activarlos de manera programática. Para ello, en los scripts que los utilizan, como el gestor de diálogos o el gestor de descripciones de cuadros, se ha añadido el código necesario para ello.

A modo ejemplificativo, en el listado 7.13 se muestra el código simplificado del gestor de diálogos que activa el efecto de sonido de una llamada. Como puede verse, recupera su componente para reproducir audio y detiene cualquier sonido que estuviera sonando para no solaparlo. Más adelante, y por convención, si el diálogo en cuestión contiene la palabra *RING*, hace sonar el respectivo sonido al 50 % de volumen.

```
1 private void PlaySoundEffect(string dialog)
2 {
3     AudioSource audioSrc = GetComponent<AudioSource>();
4     audioSrc.Stop();
5
6     if (dialog.Contains("RING"))
7     {
8         audioSrc.PlayOneShot(ringSound, 0.5f);
9     }
10 }
```

Listing 7.13: Fragmento del script para activar efectos de sonido

Finalmente, el resto de trabajo para esta entrega consistió en completar los archivos de texto con las descripciones de los cuadros. Si el lector quiere consultarlos, se encuentra en la ruta `Assets/Text/Painting_descriptions-spanish`.

Como una de las tareas para la entrega final era grabar un vídeo mostrando el juego entero, no tiene sentido grabar otro enseñando los avances en esta entrega, ya que serían lo mismo, por lo que en esta ocasión no se ha hecho.

7.8. Entrega 7

Esta última entrega se ha dedicado a realizar una evaluación del juego con usuarios reales, grabar un vídeo mostrando el juego entero, grabar y edi-

tar un tráiler del mismo, terminar de redactar e imprimir la documentación y realizar la presentación del proyecto para la defensa con tribunal.

7.8.1. Evaluación con usuarios reales

Con el fin de saber qué opinión tiene usuarios reales, se ha diseñado un cuestionario para que estos jugadores lo realicen tras completar el juego. Ha sido necesario esperar hasta esta entrega para poder ofrecer una versión terminada y pulida del juego, sin errores, para que los usuarios no pierdan el hilo del juego y su narrativa ni se sientan fastidiados porque las cosas no funcionen como deberían.

El cuestionario está compuesto de tres partes, un **pre-test** para conocer a los usuarios, en el que se introducirán a ellos mismos a través de preguntas cortas, un **test** propiamente dicho cuyo objetivo es que completen el juego con la menor ayuda posible y un **post-test**, a efectos prácticos la parte más importante, en la que los usuarios podrán aportar sus impresiones inmediatas tras terminar el juego en forma de preguntas que deben ser puntuadas del 0 al 10.

Aunque los test completos pueden verse en el anexo 3, a continuación se pasará a presentar a los usuarios, las preguntas y los resultados de las mismas.

Presentación de los usuarios

Aprovechando que el fin de semana del día 28 al 31 de junio vinieron varios amigos a ver Granada, y que tenían un bagaje tecnológico muy distinto, se realizó el test con ellos. A continuación, se presentan brevemente estos usuarios.

- **Ramona Romero de Ávila.** 22 años. Estudiante del máster de Historia del Arte de Granada. Aunque disfruta viendo a otra gente jugar, se declara nula para los videojuegos y apenas ha jugado a ninguno. Ya había probado el juego antes de su versión final previamente, por lo que estaba más confiada que los demás.
- **Eduardo Labián.** 22 años. Estudió de Lenguas Inglesas y actualmente trabaja en Londres. Está muy interesando en los videojuegos y las nuevas tecnologías, aunque nunca había probado un juego VR.
- **Álvaro Briñas.** 23 años. Estudiante del máster de Economía en Madrid. Apenas ha jugado a videojuegos, y solo conoce los títulos más famosos. Tampoco había utilizado nunca un casco de VR.

- **Jaime García.** 23 años. Estudiante del máster de Historia del Arte en Granada. Suele jugar a videojuegos en su tiempo libre, aunque no con mucha asiduidad. Tampoco había probado nunca tecnologías VR.

Sus respuestas a la fase pre-test, como se ha indicado antes, pueden verse en el anexo 3.

Preguntas

Las preguntas a las que han tenido que contestar con una valoración del 0 al 10 tras completar el test han sido las siguientes.

1. ¿Está contento con el juego?
2. ¿Ha sido una experiencia inmersiva?
3. ¿Se ha sentido mareado en algún momento?
4. ¿Le ha gustado la estética del juego?
5. ¿Le ha resultado sencillo adaptarse a los controles del juego gracias al tutorial?
6. ¿Le ha parecido intuitiva la interacción con los objetos virtuales?
7. ¿Le ha resultado complicado descubrir la palanca en la segunda sala?
8. ¿Le ha parecido complicado disparar con el arco?
9. ¿Le ha costado trabajo encontrar las letras en la sala de la niebla?
10. ¿Le ha parecido tedioso resolver los puzzles con las piezas de los cuadros?
11. Cuando por fin lo ha descubierto, ¿esperaba que fuese el vigilante el ladrón del cuadro?
12. ¿Le ha parecido satisfactorio ser un detective y resolver el caso del cuadro robado?
13. ¿Se ha sentido perdido en algún momento del juego?
14. ¿Volvería a jugar?
15. ¿Ha aprendido algo de arte con este juego?

El objetivo de las mismas, como es fácil suponer, es saber si los jugadores han estado a gusto a lo largo de la partida, si alguna de las pruebas del museo es innecesariamente complicada y si el sentimiento final tras completar el juego es positivo.

Resultados

Aunque los siguientes resultados pueden ser consultados en la hoja de respuesta de los distintos usuarios, en la figura 7.28, se presentan en forma de gráfico para que sea más cómodo para el lector consultarlos. Estos resultados pertenecen a un conjunto muy reducido de usuarios, por lo que no deberían ser entendidos como globalmente representativos; en su lugar, han servido simplemente para tener una idea general del resultado final del juego y de los objetivos que se intentaban obtener.

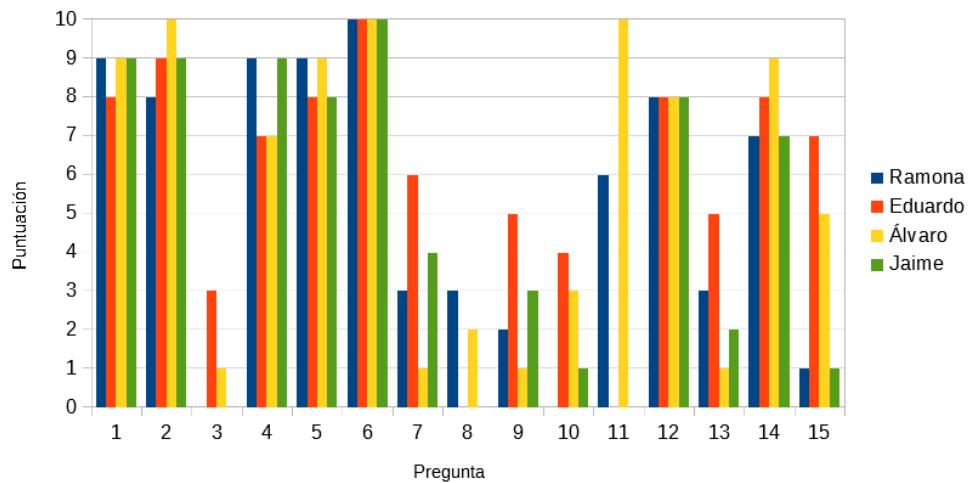


Figura 7.28: Resultados de la evaluación con usuarios

Como puede verse, hay muchos datos juntos, lo que dificulta la tarea de entender correctamente el gráfico. Para ello, se ha decidido realizar la media de cada pregunta, que puede verse en la figura 7.29. Además, para que el lector pueda saber qué valores hubieran sido óptimos en cada respuesta, se ha añadido una línea roja con ellos.

Como se ha indicado antes, estos resultados no son muy representativos, ya que por ejemplo el contar con dos historiadores del arte entre los candidatos a la evaluación ha hecho que las respuestas a la pregunta 15 sean muy dispares, y las personas que ya habían utilizado un headset VR se acostumbrarán mucho más rápido al juego, lo que les puede hacer creer que el

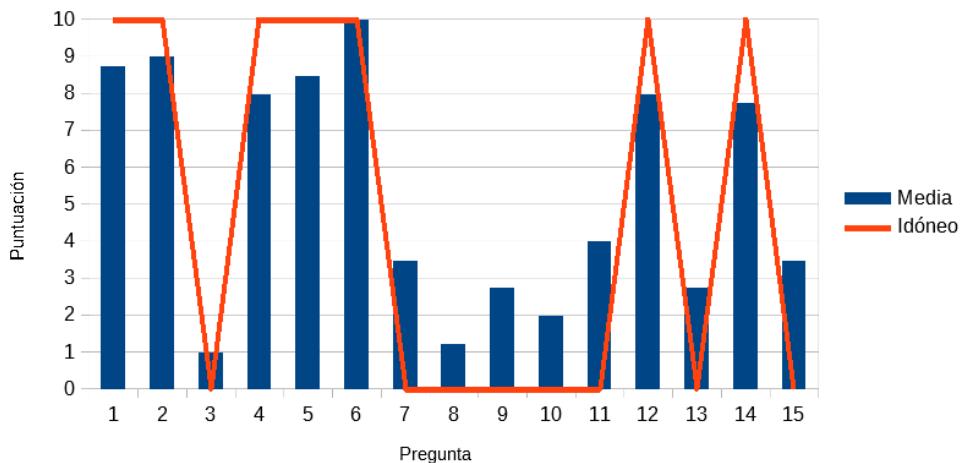


Figura 7.29: Media de los resultados de la evaluación

tutorial les ha sido de más ayuda. Aún así, si se contara con un conjunto de usuarios mayor se habría llevado a cabo una metodología de evaluación similar, que es por lo que se considera que esta, aunque solo exacta en la medida de lo posible, es representativa.

En general, las respuestas han resultado ser bastante buenas, así que aunque se esperaba tener que modificar algunos aspectos del juego, como esconder más la palanca de detrás del cuadro en la sala del gótico o hacer más obvias las letras escondidas en los cuadros de la sala del renacimiento, se ha decidido no hacerlo, ya que en un intento de mejorar cosas que funcionan se podrían empeorar, y ya no sería tan sencillo encontrar usuarios con los que repetir la evaluación.

7.8.2. Versión final del juego y trailer

A continuación se presenta el vídeo enseñando el juego finalizado. Se ha esperado hasta este momento para grabarlo porque, como se ha explicado antes, era necesario conocer la opinión de los usuarios, ya que había varios aspectos susceptibles a ser modificados. El vídeo puede ser consultado en el siguiente enlace.

<https://youtu.be/YJHA07s5M0k>

Por otro lado, también se ha grabado un tráiler que muestre de manera resumida el estilo, la narrativa, objetivo y el potencial de este proyecto. El

público objetivo de este tráiler podrían ser desde el propio tribunal de este TFM a un usuario valorando si adquirir o no el juego, pasando por posibles inversores. Para este vídeo se ha utilizado la canción *The Jazz man*, la misma que para la última sala.

<https://youtu.be/sYCLIU3ZRdw>

Por otro lado, se ha finalizado la documentación y se ha realizado una la presentación que utilizar para la defensa del tribunal, completando así la fasa de desarrollo de este proyecto.

Capítulo 8

Conclusiones y Trabajo futuro

En este capítulo se detallarán las conclusiones obtenidas del desarrollo de este proyecto, y se valorará si los objetivos propuestos en el capítulo 1 han sido cumplidos. Además, se propondrán posibles líneas de trabajo futuro para mejorar este proyecto o que tomar como referencia para una posible segunda entrega.

8.1. Conclusiones

Este TFM ha simulado el ciclo de desarrollo de un videojuego real en el que se ha implementado un videojuego completo con tecnologías VR. Este desarrollo se ha dividido entregas, al final de las cuales se ha generado un entregable que se ha documentado a través de vídeos periódicos.

Tomando como referencia los resultados de la evaluación con usuarios reales realizada al final, se puede afirmar que se han cumplido los objetivos planteados al principio del proyecto; crear una experiencia inmersiva en la que el jugador se sintiera gratificado al resolver los diferentes acertijos que se le proponen. Además, esta experiencia puede motivarles para visitar museos reales. Por otro lado, si el usuario de *MineRVa* padece algún tipo de impedimento físico que le impida hacerlo, se le ofrece una manera de emularlo haciendo uso del potencial motivador de un videojuego.

Por otro lado, con respecto a los objetivos secundarios, de índole más personal, se considera que este proyecto ha servido como primera toma de contacto tanto al desarrollo con tecnologías VR y a sus principales frameworks y librerías como a los principales procesos y metodologías de desarrollo

de videojuegos, así como al documento GDD y las evaluaciones con usuarios.

Además, el trabajo de investigación realizado para el capítulo 2 ha servido para entender cómo han nacido las tecnologías utilizadas para desarrollar este proyecto y la motivación de sus creadores, como Blender o VRTK, y todos problemas que tuvieron que superar para lograr sus objetivos.

Gracias a la división de trabajo al inicio del proyecto, la planificación por entregas y la utilización de metodologías ágiles ha sido posible terminarlo en el tiempo establecido, evitando así tener que realizar una fase de *crunch* en la etapa final. Este término, que desafortunadamente está actualmente de moda, hace referencia al período en el que desarrolladores de un proyecto, normalmente informático, se ven obligados a trabajar por encima de las horas establecidas. Estos período suelen darse al final de los proyectos y se utilizan como única manera para poder llegar a la fecha de entrega estipulada. En su lugar, ha sido posible compaginarlo con el máster y las prácticas y completarlo a tiempo.

El desarrollo de este proyecto me ha permitido afianzar los conocimientos obtenidos a lo largo de las distintas asignaturas estudiadas en el máster, además de recordar conceptos aprendidos a lo largo de la carrera relacionados con interfaces y sistemas interactivos y de tiempo real.

En definitiva, considero que este proyecto es el principio de mi carrera profesional, que espero poder dedicar al desarrollo de sistemas interactivos y en la que espero poner en práctica los conocimientos adquiridos tras la realización de este TFM.

8.2. Trabajo futuro

A pesar de haber completado todos los objetivos propuestos inicialmente y haber aportado algunos nuevos a lo largo del desarrollo, hay algunas opciones que no se han llevado a cabo o bien por falta de tiempo o bien porque no se cree que actualmente aporten suficiente valor al producto en comparación con su carga de trabajo.

Además, a diferencia de otro tipo de proyectos informáticos, los videojuegos son especialmente susceptibles a, en lugar de mejorar una versión, reunir todas estas mejoras y desarrollar una nueva entrega, teóricamente mejor que la anterior, por lo que es especialmente interesante plantear un posible trabajo futuro en proyectos de esta naturaleza.

A continuación se presentan las líneas de trabajo futuro más relevantes que podrían seguirse de este proyecto.

- **Traducir a diferentes idiomas.** Uno de los objetivos de un videojuego es que sea jugado por el mayor número de personas posibles, y una de las mayores barreras de entrada, si no la que más, es el lenguaje.

Por ello, se ha añadido el esqueleto de lo que en un futuro podría ser el soporte multilenguaje. Para lograr esto, se ha añadido un selector en los script de gestión de los diálogos y las descripciones de los cuadros en el que se puede seleccionar un lenguaje, que por debajo simplemente de qué carpeta leer los archivos de texto. En la figura 8.1 puede verse el caso de este último, que actualmente solo soporta el español.

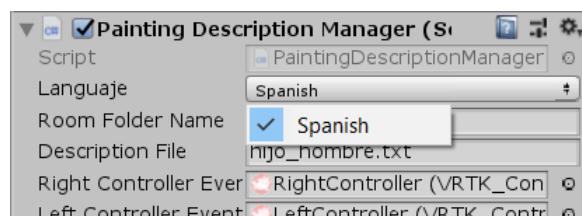


Figura 8.1: Script para cambiar el idioma desde el inspector

Siguiendo este enfoque, cada lenguaje tendría una carpeta dentro del directorio de diálogos y descripciones. Para completarlo, en el menú principal sería necesario incluir una sección desde la que el usuario pudiera seleccionar su lenguaje, y ésto actualizase el resto de scripts.

De este modo sería suficiente con proporcionar a un traductor los archivos de texto del juego e incluirlos directamente dentro de su carpeta, una vez traducidos, sin necesidad de modificar el código fuente. Además, sería posible que las traducciones de los diálogos se dividieran en un número mayor o menor de archivos, ya que el sistema continua leyéndolos hasta que no encuentra más, lo que les proporcionaría mayor flexibilidad y les permitiría a los traductores adaptar estas traducciones de mejor manera.

- **Guardar y cargar partidas.** Quizá uno de las mayores ausencias de funcionalidad del juego es la habilidad para poder guardar la partida y poder continuarla en un futuro. Esto no ha sido una prioridad a lo largo del desarrollo por varios motivos, siendo el principal que en un juego relativamente corto (si tomamos como referencia el vídeo mostrando el juego terminado, aproximadamente media hora) y que la dificultad de una sala reside en saber qué hay que hacer para avanzar, y una vez que se ha completado es inmediato, se considera que no es algo esencial y que el su lugar se podría añadir en un futuro.

Además, por la naturaleza del juego sería relativamente sencillo guardar una partida en disco, bastando simplemente con almacenar la sala

en la que se quedó el jugador, lo que si quisiéramos simplificar aún más podría traducirse en un selector de salas en el menú principal.

- **Aumentar la rejugabilidad.** Como ya ha podido verse en el punto anterior y en los resultados de las evaluaciones con usuarios, el principal problema del juego es su falta de rejugabilidad; esto es, una vez que el usuario completa el juego y conoce la historia y las salas y sus pruebas no tiene ninguna motivación por volver a jugar.

Esto podría intentar evitarse sin desviarse excesivamente de la naturaleza del juego variando, por ejemplo, los cuadros que hay en cada sala. Sería algo relativamente sencillo, y simplemente sería necesario elegir aleatoriamente qué cuadros va a tener una sala en el momento de cargarla. No sería una generación procedural completa, ya que la sala en sí sería la misma siempre; simplemente se sustituirían unos cuadros por otros que habrían sido modelados e incluidos previamente.

- **Publicación.** El paso lógico de un videojuego terminado es ser publicado, ya sea gratuito o de pago, y la plataforma en la que lo hacen la mayoría de los juegos es Steam, aunque por cuestión de agenda no ha sido posible hacerlo. Aún así, la versión compilada del juego está disponible públicamente en la sección *releases* del repositorio del proyecto o, de manera equivalente, siguiendo el siguiente enlace.

<https://github.com/gomezportillo/mineRVa/releases>

Capítulo 9

Apéndices

9.1. Apéndice 1: Guía de salas

EN las páginas siguientes se adjunta la primera versión de la estructura de las salas del museo que fue propuesta al inicio del proyecto junto a su disposición y sus cuadros.

MineRVa - Generador de museos virtuales

Guía de salas

La idea principal es que, tras desaparecer la obra maestra del museo, te llaman para descubrir quién la ha robado. Tras llegar al museo, el vigilante, un simpático anciano, te explica la situación y comienza a guiarte a través de las salas en las que se descubren pistas tras resolver pequeños puzzles. Finalmente se descubre que el vigilante ha sido el ladrón.



Concept art del vigilante,
https://wordgirl.fandom.com/wiki/Museum_Guard

El museo tendrá 7 salas, ordenadas cronológicamente por las obras que contienen, que además tendrán al lado una pequeña descripción para que el jugador pueda leerla.

La estética general será *low poly*, y cada sala estará ambientada ligeramente en su época.

Sala 1 – Tutorial.....	2
Sala 2 – Gótico/Flamenco.....	3
Sala 3 – Renacimiento.....	5
Sala 4 – Barroco.....	7
Sala 5 – Romanticismo.....	8
Sala 6 – Vanguardias del siglo XX.....	10
Sala 7 – Final.....	11

Sala 1 – Tutorial

Esta sala funciona como un pequeño tutorial para que el jugador se familiarice con los controles y la mecánica del juego. Esta sala tendrá un hueco para un cuadro vacío, y el vigilante le contará explicará el contexto al jugador.

Además, en la sala estará *El hijo del hombre*, una cesta con varias piezas de frutas diferentes y la puerta de salida cerrada con llave.

Tras contarle la historia al jugador, el vigilante le dirá que ha perdido su merienda y que, aunque no recuerda exactamente qué es, sí sabe que tiene algo que ver con el cuadro que hay. Si le damos la pieza de fruta correcta (una manzana), nos da la llave para ir a la siguiente sala.



El hijo del hombre, René Magritte, 1964

Sala 2 – Gótico/Flamenco

Esta sala será cuadrada y tendrá tres cuadros; a los lados, *El Matrimonio Arnolfini* y *El descendimiento de van der Weyden*, a modo representativo del género, y en la pared del fondo *El jardín de las delicias*.



El Matrimonio Arnolfini, Jan van Eyck, 1434



El descendimiento de la cruz, Rogier van der Weyden, 1436



El jardín de las delicias, El Bosco, ~1500

La idea de esta sala será resolver algún puzzle relacionada con el tríptico, tras lo que se partirá por la mitad mostrando la puerta hacia la siguiente sala. Por ejemplo, se podrían desordenar los paneles (poner el Infierno en la izquierda y el Edén en la derecha). Tras colocarlos correctamente, el panel del medio se partiría por la mitad abriendo el camino a la siguiente sala.



Sala 3 – Renacimiento

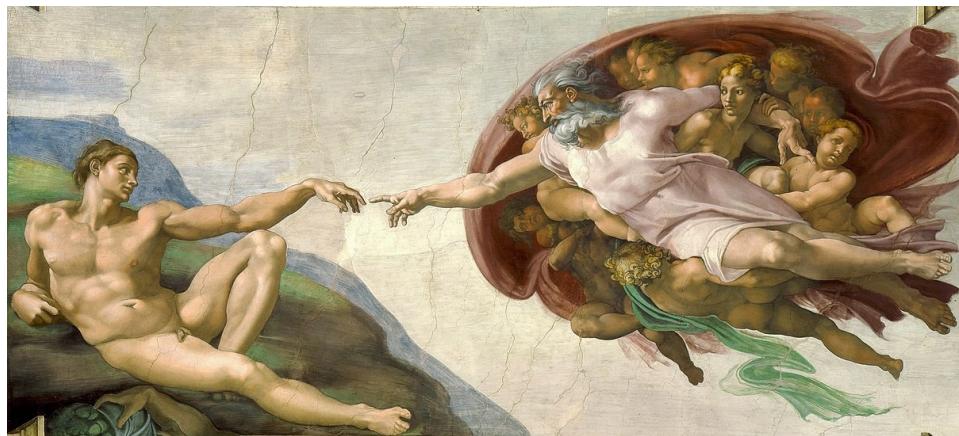
En esta sala habrá 3 cuadros, *La última cena*, *La escuela de Atenas* y *La creación de Adán*.



La última cena, Leonardo da Vinci, ~1495



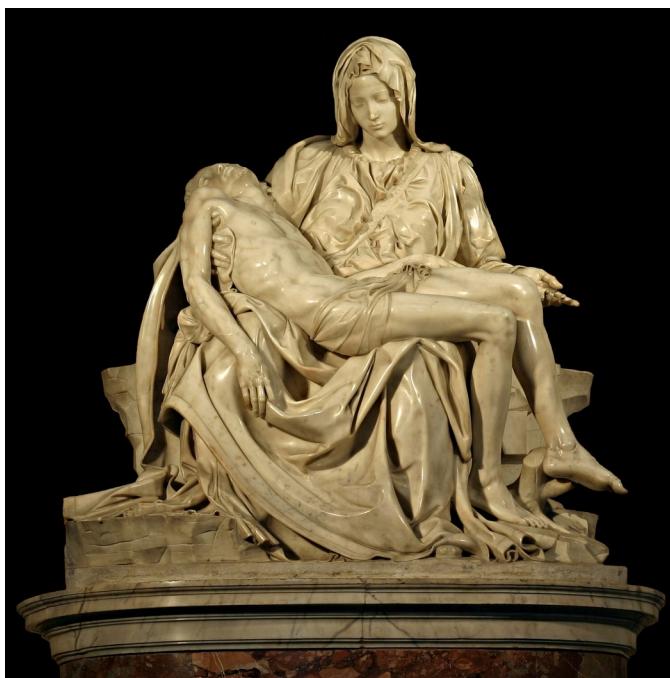
La escuela de Atenas, Rafael, 1511



La creación de Adán, Miguel Angel, 1511

El objetivo de estas salas será encontrar un objeto anacrónico en los cuadros. Por ejemplo, el Judas de *La última cena* (con el que el vigilante simpatizará con algún comentario casual) puede tener un móvil en la mano, en *La escuela de Atenas* puede haber un balón de fútbol reglamentario y en *La creación de Adán* un avión en el cielo. Tras encontrarlos todos los objetos (para lo que podría bastar seleccionarlos con el mando) el vigilante abrirá la puerta a la siguiente sala.

Para ambientar la sala también se buscará alguna escultura 3D, como la *Piedad* o el *Moisés*, ambas de Miguel Ángel, ya modelada y con licencia libre.



Sala 4 – Barroco

El Barroco fue una época especialmente fructífera en España lo que refiere a la pintura. Por eso, se ha pensado que en esta sala aparezcan 3 cuadros al azar de una batería y el vigilante le pregunte al jugador por sus nombres dándole a elegir entre 3 opciones. Si falla puede volver a intentarlo y cuando acierte los 3 pasa a la siguiente sala.

Algunos de los cuadros que se pueden seleccionar son, por autor, los siguientes.

- Diego Velázquez
 - *La fragua de Vulcano*, 1630
 - *Viejariendo huevos*, 1618
 - *El bufón don Sebastián de Morra*, 1644
- Murillo
 - *Sagrada Familia del pajarito*, 1648
 - *Niños comiendo uvas y melón*, 1650
 - Inmaculada de Soult, 1678
- Francisco de Zurbarán
 - Cristo en la cruz, 1627
 - *Agnus dei*, 1635
 - *Bodegón con cacharros*, 1633
- Francisco de Goya
 - *El quitasol*, 1777
 - *El sueño de la razón produce monstruos*, 1797
 - *La familia de Carlos IV*, 1800
 - *Los fusilamientos del 3 de mayo*, 1813
 - *Saturno devorando a su hijo*, 1820

Sala 5 – Romanticismo

La estética de esta sala será oscura, con pocas fuentes de luz y muy ligeras, solo para iluminar los cuadros. Además, se probará a incluir partículas de humo con Unity a la altura del suelo, lo que la hace perfecta para esconder cosas. Por eso, se ha pensado en esconder el botón que abra la puerta a la siguiente sala y dejar pistas en los cuadros que indiquen su posición. Concretamente, el botón estará en el techo y los cuadros esconderán la palabra ARRIBA.

Los cuadros elegidos han sido los siguientes.

- *El Fusilamiento de Torrijos y sus compañeros*, Antonio Gisbert, 1888, que esconderá las letras **AR**
- *Juana la Loca*, Francisco Pradilla, 1877, que esconderá las letras **RI**.
- *El caminante sobre un mar de nubes*, Caspar David Friedrich, 1818, que esconderá las letras **BA**.

Las letras pueden estar situadas en las esquinas de los cuadros, incluso giradas, obligando de este modo al jugador a tener que mirar detenidamente los cuadros.



El Fusilamiento de Torrijos y sus compañeros



Juana la Loca



El caminante sobre un mar de nubes

Sala 6 – Vanguardias del siglo XX

Como en el caso del Barroco, la cantidad de obras y estilos de este período dificulta la tarea de elegir 3 o 4 representativos. Por eso, se ha pensado, reutilizando parte del código hecho para la sala del Barroco, mostrar dos cuadros que comparten estilo y autor y que el vigilante pida al jugador que lo identifique. Estos cuadros podrían ser los siguientes

- Cubismo
 - *Las señoritas Avignon*, Pablo Picasso
 - *Guernica*, Pablo Picasso
- Expresionismo
 - *El grito*, Edvard Munch
 - *Autorretrato con una botella de vino*, Edvard Munch
- Surrealismo
 - *La persistencia de la memoria*, Salvador Dalí
 - *La tentación de San Antonio*, Salvador Dalí
- Postimpresionismo
 - *La noche estrellada*, Vincent van Gogh
 - *Los girasoles*, Vincent van Gogh

Por ejemplo, si el estilo es cubismo, se ha elegido a Pablo Picasso como su representante. Se mostrarán dos cuadros, uno a cada lado de la sala, dejando la pared del fondo libre para que se coloque el vigilante, y éste le preguntará al jugador por el autor o el estilo de las obras (aún por decidir). Como anteriormente, se dará a elegir entre 3 opciones y si falla podrá volver a intentarlo.

Además, para adornar la sala, se colocará un modelo 3D de la Sagrada Familia, de Antoni Gaudí.

Sala 7 – Final

Una vez superadas todas las salas anteriores, el jugador podrá acceder a la sala final del museo, en la que ocurrirá el desenlace del juego. Esta sala estará vacía a excepción de un único caballete con un cuadro tapado. Al acercarse el jugador, el vigilante confesará que fue él quien robó el cuadro y que lo escondió en la última sala pensando que nadie podría pasar sus pruebas para llegar hasta él y encontrarlo.

Tras esto, habrá una pantalla en negro en la que se le dará las gracias al jugador por encontrar el cuadro y terminará el juego.



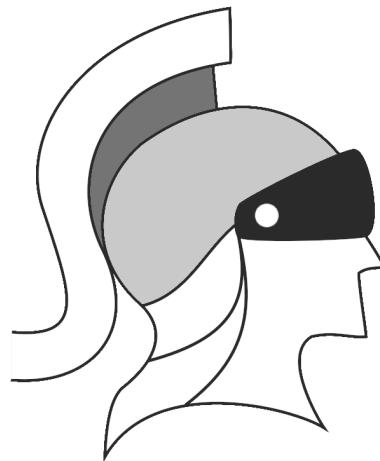
9.2. Apéndice 2: Game Design Document

A continuación se adjunta el GDD final de *MineRVa*.

Game Design Document

MineRVa

Experiencia de juego en Realidad Virtual en un museo



Pedro Manuel Gómez-Portillo López

Revisón

1.0.1

Fecha

31 de mayo de 2019

Índice

1. Información general	3
1.1. Resumen del juego	3
1.2. Objetivos a alcanzar por el juego	3
1.3. Justificación del juego	4
1.4. Core gameplay	4
1.5. Características del juego	4
1.5.1. Género	4
1.5.2. Público objetivo	5
1.5.3. Plataforma de destino	5
1.5.4. Estética y estilo artístico	5
1.6. Características del jugador	5
1.6.1. Arquetipo 1	5
1.6.2. Arquetipo 2	6
1.6.3. Arquetipo 3	6
1.7. Recursos iniciales	6
2. Mecánicas	7
2.1. Elementos de juego. Núcleo	7
2.2. Reglas de interacción	7
2.3. Elementos de juego. Mundo	7
2.4. Elementos de registro y progreso	8
2.5. Elementos de jugabilidad y experiencia del jugador	8
3. Dinámica	9
3.1. Mundo de juego. Universo virtual	9
3.1.1. Historia detallada	9
3.2. Descripción de niveles	9
3.2.1. Objetivos	9
3.2.2. Recompensas	9
3.3. Interfaz del juego	9
3.3.1. Controles de la interfaz	10
3.4. Aprendizaje del juego	10
4. Estética y arte	11
4.1. Estética	11
4.2. Recursos artísticos	11

5. Marketing	12
5.1. Portada del juego	12
5.2. Anuncios	12
5.3. PEGI	12
6. Información del documento	13
6.1. Acrónimos y abreviaturas	13

1. Información general

1.1. Resumen del juego

MineRVa es un proyecto que utiliza la Realidad Virtual para crear una experiencia de juego en un museo a través del que el jugador debe resolver pruebas y puzzles para llegar al final y descubrir con ayuda del vigilante quién ha robado el cuadro más importante del museo.

Este museo está compuesto por varias salas, cada una ambientada en una de las principales fases de la Historia del Arte; Gótico, Renacimiento, Barroco, Romanticismo y Vanguardias. Cada sala será temática y tendrá varias obras de arte, como cuadros y estatuas, con las que el jugador tendrá que interactuar para abrirse paso hasta llegar al final y resolver el misterio del cuadro robado.

1.2. Objetivos a alcanzar por el juego

El objetivo de *MineRVa* es crear una experiencia inmersiva y en la que el jugador se sienta gratificado al ir resolviendo los pequeños acertijos que se le proponen. Con ello se pretende alcanzar dos objetivos diferentes; por un lado, se busca conseguir que nativos digitales que no suelen visitar museos encuentren en *MineRVa* una excusa para hacerlo de manera divertida, y por el otro que usuarios más mayores a los que les gusta visitarlos encuentre en la temática de este proyecto una excusa para jugarlo y disfrutarlo de la misma forma. En cualquiera de los dos casos se espera que, tras completar el juego, la mayoría de usuarios se interesen en visitar museos reales.

Por otro lado, uno de los nichos de jugadores más pequeños y a la vez uno en el que más se ha pensado es en aquellas personas discapacitadas o con movilidad reducida que no pueden físicamente visitar museos. Se espera ayudar especialmente a este colectivo consiguiendo una experiencia equivalente en la medida de lo posible, además de intentar hacerla más amena enmarcándola en el contexto en un videojuego.

1.3. Justificación del juego

A fecha de hoy, *MineRVa* supone el primer videojuego en Realidad Virtual que se desarrolla en un museo y que no tiene como principal objetivo visitar el museo. Ya hay varios proyectos de Realidad Virtual que intentan conseguir una experiencia realista utilizando modelos digitalizados, pero ninguno lo usa como excusa para desarrollar un videojuego con pruebas para el jugador, y aunque son proyectos muy interesantes puede que no satisfagan todas las expectativas de un usuario que va buscando un videojuego.

Por esto, se considera que *MineRVa* es un proyecto pionero en la manera de unir videojuegos, museos y Realidad Virtual.

1.4. Core gameplay

Una vez comience el juego, el jugador deberá usar los mandos para moverse por cada sala, que inicialmente tendrá la puerta de salida bloqueada, e ir investigando las pistas de las que dispone. Además, podrá interactuar con el vigilante del museo, que se utilizará como elemento de ayuda por si el jugador se atasca, para obtener información adicional.

Una vez encuentre y resuelva el puzzle que cada sala le propone la puerta de salida se desbloqueará y podrá pasar a la siguiente hasta que finalmente llegue a la última y resuelva el misterio del cuadro robado.

Los puzzles estarán basados en interacciones simples enmarcadas en el contexto de la Realidad Virtual; el jugador deberá coger, accionar, tirar o empujar objetos virtuales utilizando los mandos.

1.5. Características del juego

1.5.1. Género

El género principal de este juego sería simulación, pero como también incorpora mecánicas de resolución de acertijos podríamos definirlo como una mezcla de los dos; es decir, **simulación/puzzles**.

1.5.2. Público objetivo

Como ya se ha indicado antes, tanto por la temática como por la estética y las mecánicas de *MineRVa* se espera que su nicho de mercado sea muy variado, desde niños a adultos. Además, como no se tiene previsto incorporar mecánicas multijugador, será un juego únicamente de un solo jugador.

1.5.3. Plataforma de destino

La única plataforma para la que en principio se espera lanzar el juego es para ordenador.

1.5.4. Estética y estilo artístico

Como el tiempo de desarrollo de este proyecto va a ser muy escaso, será necesario decantarse por un estilo artístico que sea simple y que permita un modelado relativamente rápido de todos los objetos necesarios para el juego. Por tanto, se ha elegido una estética principalmente **low poly**, aunque se intentará hacer especialmente inmersiva usando texturas realistas.

1.6. Características del jugador

A continuación se presentan los tres principales arquetipos de jugadores en los que se ha pensado a la hora de diseñar el juego, aunque evidentemente prácticamente cualquier persona puede jugarlo.

1.6.1. Arquetipo 1

Nativos digitales de hasta 15 años que nunca han visitado un museo ni tienen apenas conocimiento de la Historia del Arte, pero que están en continuo contacto con nuevas tecnologías y son especialmente susceptibles a los videojuegos. Para ellos, *MineRVa* puede ser una primera toma de contacto perfecta con las fases del Arte y algunas de sus obras de arte más importantes. Además, como todas las pruebas han sido diseñadas teniendo esto en cuenta, no requieren de complicadas interacciones sino que se ha intentado que sean intuitivas.

1.6.2. Arquetipo 2

Adultos de entre 30 y 50 años aproximadamente que disfruten realizando actividades culturales como visitar museos y que no tengan miedo de probar nuevas tecnologías como la Realidad Virtual. Para ellos, las sencillas mecánicas y las intuitivas interacciones de *MineRVa* pueden ser perfectas para sumergirse en la experiencia de un mundo virtual y resolver el misterio del robo del museo.

1.6.3. Arquetipo 3

Por último, y como ya se ha introducido antes, se tendrá especialmente en cuenta a las personas discapacitadas o con movilidad reducida que quieran pero por motivos físicos no puedan visitar museos. Para ellos, *MineRVa* puede ser un modo de sumergirse en un museo, aunque sea virtual, y obtener una experiencia por lo menos parecida.

Además, puede ser hasta más entretenido para ellos el resolver los pequeños acertijos que las salas les proponen, ya que se ha huído de las interacciones complejas y engorrosas pensando especialmente en ellos.

1.7. Recursos iniciales

Los recursos de este proyecto son muy reducidos. Como se enmarca dentro de un Trabajo Fin de Máster, el presupuesto será lo más próximo a los 0 euros posible, por lo que si es necesario usar texturas y modelos de internet deberán ser gratuitos.

Por otro lado, solo se cuenta con un desarrollador que se hará cargo del proyecto entero en un máximo de 6 meses, que serán los recursos temporales.

2. Mecánicas

2.1. Elementos de juego. Núcleo

- **Sala.** Cada una de las divisiones con personalidad propia del juego. Cada sala es una habitación ambientada en un período histórico distinto y que propone distintas obras de arte y puzzles al jugador.
- **Museo.** Es el conjunto de todas las salas en las que transcurre el juego.
- **Obra de arte.** Cada uno de los elementos que existen fuera del juego como obras reales y con las que el jugador puede interactuar para obtener más información sobre ellas. Pueden presentarse en forma de cuadro o escultura.
- **Puzzle.** Cada uno de los acertijos que el jugador tiene que resolver para desbloquear el acceso a la siguiente sala.
- **Vigilante.** Persona encargada del museo y que guiará al usuario a través de las salas, haciendo a su vez de mentor y ayudándole cuando se quede atascado.

2.2. Reglas de interacción

- Cada nueva sala a la que el usuario acceda estará bloqueada y solo podrá avanzar cuando resuelva su puzzle.
- Los puzzles estarán basados en interacciones sencillas del usuario con los controladores, simulando interacciones en el mundo real como coger un objeto o tirar de él.
- El jugador podrá moverse hacia atrás en las salas para volver a aquellas que ya visitó con anterioridad.

2.3. Elementos de juego. Mundo

El mundo de este juego es un *universo paralelo* en el que existen las mismas obras de arte que en éste y tienen la misma historia pero han ido a parar al museo, que es custodiado por un vigilante ficticio.

2.4. Elementos de registro y progreso

El jugador salir del juego en cualquier momento y guardar su partida para luego volver a acceder, cargarla, y mantener el mismo progreso que tenía antes de salir.

2.5. Elementos de jugabilidad y experiencia del jugador

Toda la experiencia del jugador gira en torno a la Realidad Virtual y sus interacciones, que intentarán realistas para que esta experiencia sea lo más inmersiva posible. Todas estas interacciones se harán por medio de los mandos, a través de las cuales el jugador podrá comunicarse con el mundo virtual.

3. Dinámica

3.1. Mundo de juego. Universo virtual

3.1.1. Historia detallada

La historia del juego coloca al jugador el rol de un detective infalible que ha sido contratado por el museo para resolver el misterio de la desaparición de su mayor obra de arte, y todo apunta a que ha sido robado.

El encargado de recibir al jugador en el museo es su vigilante, un vetusto simpático y campechano señor que ha pasado toda su vida trabajando en él. Este personaje será el encargado de introducir al jugador la historia y las mecánicas del juego, de explicarle su misión y de guiarle a través de las distintas salas, ayudándole para que no se quede atascado.

3.2. Descripción de niveles

3.2.1. Objetivos

El objetivo de cada nivel o sala será resolver el puzzle propuesto para acceder a la siguiente. El objetivo secundario será que el usuario interactúe con las obras de arte y aprenda sobre ellas, ya que en alguno de los casos le será necesario.

3.2.2. Recompensas

Cada sala individual no tiene recompensa más allá de desbloquear el acceso a la siguiente; sin embargo, cuando el jugador complete todos los puzzles podrá resolver el misterio del museo y descubrir dónde está el cuadro y quién lo robó.

3.3. Interfaz del juego

La interfaz del juego intentará ser lo menos intrusiva posible, manteniéndose al mínimo. Durante el juego, la interfaz más intrusiva será el cuadro de texto en el que el jugador puede leer la información de los cuadros; el resto de elementos estarán camuflados en el entorno de manera realista.

3.3.1. Controles de la interfaz

La interfaz, como intenta ser mínimamente intrusiva, tendrá que ser manejada usando modelos de interacción inmersivos como señalar con el mando.

3.4. Aprendizaje del juego

Como el nicho de mercado de este juego no son jugadores experimentados, ni el propio juego es especialmente complejo, se intentará que la curva de aprendizaje sea lo más liviana y guiada posible para que el número de personas que comienzan el juego y lo dejan por no entenderlo sea mínimo.

Por ello, la primera sala a la que el jugador acceda será un tutorial en el que se le expliquen tanto la historia como las dinámicas de manera clara y ligera. Por otro lado, los primeros puzzles que resuelva serán muy sencillos, y no se planteará hacer algo mínimamente complejo hasta el último tercio del juego.

4. Estética y arte

4.1. Estética

Como el presupuesto de este proyecto es nulo y el tiempo disponible para completarlo es especialmente bajo, la estética deberá ser acorde a estos factores. Por ello, se ha elegido una estética *low poly*, con lo que los tiempos de modelado se verán muy reducidos.

Aún así, se intentará que no sea especialmente plana con la ayuda de texturas medianamente realistas.

4.2. Recursos artísticos

Aunque el modelado de las salas deberá ser hecho por el equipo, se podrá importar de internet modelos tridimensionales especialmente complejos, como las estatuas que se crean necesarias para las salas; estos modelos suelen ser digitalizaciones exactas, que se adaptarán para reducir el número de polígonos, reducir el peso total del juego y hacer que no sea necesario un equipo hardware potente para jugarlo.

Por otro lado, se podrán importar texturas de internet para ayudar a dar personalidad a las salas, pero todas ellas deberán ser gratuitas.

5. Marketing

5.1. Trailer

Cuando el juego se termine se generará un tráiler narrado de entre 90 y 120 segundos que se subirá a la plataforma YouTube y que servirá como anuncio del juego.

5.2. PEGI

El Sistema PEGI es el mecanismo de autorregulación diseñado por la industria para dotar a sus productos de información orientativa sobre la edad adecuada para su consumo. Este juego será **PEGI 3**, ya que, según el propio sitio web¹,

“El contenido de los juegos con una clasificación PEGI 3 se considera adecuado para todos los grupos de edad. El juego no debe contener sonidos o imágenes que puedan asustar a los niños pequeños. Una forma muy leve de violencia (en un contexto cómico o en un entorno infantil) es aceptable. No se debe escuchar un lenguaje soez.”



¹ <https://pegi.info/es/node/59>

6. Información del documento

6.1. Acrónimos y abreviaturas

- **VR.** Realidad Virtual
- **PEGI.** Pan European Game Information

9.3. Apéndice 3: Evaluación de satisfacción con usuarios

A continuación se adjuntan los cuestionarios resueltos de las evaluaciones de satisfacción de los usuarios.

Nombre y apellidos	Ramona Romero
Fecha	29 de junio de 2019

Evaluación de satisfacción de MineRVA

1. Pre-test

Por favor, responda las siguientes preguntas para permitirnos conocerlo mejor.

ID	Pregunta	Respuesta
1	¿Qué edad tiene?	22
2	¿Es usted hombre o mujer?	Mujer
3	¿Cuál es su ámbito de estudio o trabajo?	Máster de Historia del Arte en Granada
4	¿Suele visitar museos?	Sí
5	¿Está familiarizado con el mundo de los videojuegos?	No mucho
5	¿Con qué frecuencia juega a videojuegos?	Casi nunca
6	¿Ha jugado alguna vez a algún juego en Realidad Virtual?	Sí
7	Si sí, ¿le ha gustado?	Sí
8	Por lo poco que sabe de antemano, ¿está interesado en probar este proyecto?	Sí

2. Test

Por favor, complete el juego con la menor ayuda posible y luego responda a las preguntas que se encuentran a continuación.

3. Post-test

Tras haber realizado las tareas indicadas anteriormente, por favor responda las preguntas valorando su respuesta del 0 al 10, donde 0 es el mínimo y 10 es el máximo.

ID	Pregunta	Valoración
1	¿Está contento con el juego?	9
2	¿Ha sido una experiencia inmersiva?	8
3	¿Se ha sentido mareado en algún momento?	0
4	¿Le ha gustado la estética del juego?	9
5	¿Le ha resultado sencillo adaptarse a los controles del juego gracias al tutorial?	9
6	¿Le ha parecido intuitiva la interacción con los objetos virtuales?	10
7	¿Le ha resultado complicado descubrir la palanca en la segunda sala?	3
8	¿Le ha parecido complicado disparar con el arco?	3
9	¿Le ha costado trabajo encontrar las letras en la sala de la niebla?	2
10	¿Le ha parecido tedioso resolver los puzzles con las piezas de los cuadros?	0
11	Cuando por fin lo ha descubierto, ¿esperaba que fuese el vigilante el ladrón del cuadro?	6
12	¿Le ha parecido satisfactorio ser un detective y resolver el caso del cuadro robado?	8
13	¿Se ha sentido perdido en algún momento del juego?	3
14	¿Volvería a jugar?	7
15	¿Ha aprendido algo de arte con este juego?	1

Gracias por completar el test

Nombre y apellidos	Eduardo Labián
Fecha	29 de junio de 2019

Evaluación de satisfacción de MineRVA

1. Pre-test

Por favor, responda las siguientes preguntas para permitirnos conocerlo mejor.

ID	Pregunta	Respuesta
1	¿Qué edad tiene?	22
2	¿Es usted hombre o mujer?	Hombre
3	¿Cuál es su ámbito de estudio o trabajo?	Trabajo en un colegio en Londres
4	¿Suele visitar museos?	Apenas
5	¿Está familiarizado con el mundo de los videojuegos?	Sí
5	¿Con qué frecuencia juega a videojuegos?	Siempre que puedo
6	¿Ha jugado alguna vez a algún juego en Realidad Virtual?	No
7	Si sí, ¿le ha gustado?	-
8	Por lo poco que sabe de antemano, ¿está interesado en probar este proyecto?	Sí

2. Test

Por favor, complete el juego con la menor ayuda posible y luego responda a las preguntas que se encuentran a continuación.

3. Post-test

Tras haber realizado las tareas indicadas anteriormente, por favor responda las preguntas valorando su respuesta del 0 al 10, donde 0 es el mínimo y 10 es el máximo.

ID	Pregunta	Valoración
1	¿Está contento con el juego?	8
2	¿Ha sido una experiencia inmersiva?	9
3	¿Se ha sentido mareado en algún momento?	3
4	¿Le ha gustado la estética del juego?	7
5	¿Le ha resultado sencillo adaptarse a los controles del juego gracias al tutorial?	8
6	¿Le ha parecido intuitiva la interacción con los objetos virtuales?	10
7	¿Le ha resultado complicado descubrir la palanca en la segunda sala?	6
8	¿Le ha parecido complicado disparar con el arco?	0
9	¿Le ha costado trabajo encontrar las letras en la sala de la niebla?	5
10	¿Le ha parecido tedioso resolver los puzzles con las piezas de los cuadros?	4
11	Cuando por fin lo ha descubierto, ¿esperaba que fuese el vigilante el ladrón del cuadro?	0
12	¿Le ha parecido satisfactorio ser un detective y resolver el caso del cuadro robado?	8
13	¿Se ha sentido perdido en algún momento del juego?	5
14	¿Volvería a jugar?	8
15	¿Ha aprendido algo de arte con este juego?	7

Gracias por completar el test

Nombre y apellidos	Álvaro Briñas
Fecha	29 de junio de 2019

Evaluación de satisfacción de MineRVa

1. Pre-test

Por favor, responda las siguientes preguntas para permitirnos conocerlo mejor.

ID	Pregunta	Respuesta
1	¿Qué edad tiene?	23
2	¿Es usted hombre o mujer?	Hombre
3	¿Cuál es su ámbito de estudio o trabajo?	Estudiante de Económicas
4	¿Suele visitar museos?	Sí
5	¿Está familiarizado con el mundo de los videojuegos?	No mucho
5	¿Con qué frecuencia juega a videojuegos?	De vez en cuando al FIFA
6	¿Ha jugado alguna vez a algún juego en Realidad Virtual?	No
7	Si sí, ¿le ha gustado?	-
8	Por lo poco que sabe de antemano, ¿está interesado en probar este proyecto?	Mucho

2. Test

Por favor, complete el juego con la menor ayuda posible y luego responda a las preguntas que se encuentran a continuación.

3. Post-test

Tras haber realizado las tareas indicadas anteriormente, por favor responda las preguntas valorando su respuesta del 0 al 10, donde 0 es el mínimo y 10 es el máximo.

ID	Pregunta	Valoración
1	¿Está contento con el juego?	9
2	¿Ha sido una experiencia inmersiva?	10
3	¿Se ha sentido mareado en algún momento?	1
4	¿Le ha gustado la estética del juego?	7
5	¿Le ha resultado sencillo adaptarse a los controles del juego gracias al tutorial?	9
6	¿Le ha parecido intuitiva la interacción con los objetos virtuales?	10
7	¿Le ha resultado complicado descubrir la palanca en la segunda sala?	1
8	¿Le ha parecido complicado disparar con el arco?	2
9	¿Le ha costado trabajo encontrar las letras en la sala de la niebla?	1
10	¿Le ha parecido tedioso resolver los puzzles con las piezas de los cuadros?	3
11	Cuando por fin lo ha descubierto, ¿esperaba que fuese el vigilante el ladrón del cuadro?	10
12	¿Le ha parecido satisfactorio ser un detective y resolver el caso del cuadro robado?	8
13	¿Se ha sentido perdido en algún momento del juego?	1
14	¿Volvería a jugar?	9
15	¿Ha aprendido algo de arte con este juego?	5

Gracias por completar el test

Nombre y apellidos	Jaime García
Fecha	29 de junio de 2019

Evaluación de satisfacción de MineRVA

1. Pre-test

Por favor, responda las siguientes preguntas para permitirnos conocerlo mejor.

ID	Pregunta	Respuesta
1	¿Qué edad tiene?	32
2	¿Es usted hombre o mujer?	Mujer
3	¿Cuál es su ámbito de estudio o trabajo?	Estudio el máster de Patrimonio Histórico-Artístico en Granada
4	¿Suele visitar museos?	Sí
5	¿Está familiarizado con el mundo de los videojuegos?	Sí
5	¿Con qué frecuencia juega a videojuegos?	Cuando estoy en casa, cuando no apena
6	¿Ha jugado alguna vez a algún juego en Realidad Virtual?	No
7	Si sí, ¿le ha gustado?	-
8	Por lo poco que sabe de antemano, ¿está interesado en probar este proyecto?	Sí

2. Test

Por favor, complete el juego con la menor ayuda posible y luego responda a las preguntas que se encuentran a continuación.

3. Post-test

Tras haber realizado las tareas indicadas anteriormente, por favor responda las preguntas valorando su respuesta del 0 al 10, donde 0 es el mínimo y 10 es el máximo.

ID	Pregunta	Valoración
1	¿Está contento con el juego?	9
2	¿Ha sido una experiencia inmersiva?	9
3	¿Se ha sentido mareado en algún momento?	0
4	¿Le ha gustado la estética del juego?	9
5	¿Le ha resultado sencillo adaptarse a los controles del juego gracias al tutorial?	8
6	¿Le ha parecido intuitiva la interacción con los objetos virtuales?	10
7	¿Le ha resultado complicado descubrir la palanca en la segunda sala?	4
8	¿Le ha parecido complicado disparar con el arco?	0
9	¿Le ha costado trabajo encontrar las letras en la sala de la niebla?	3
10	¿Le ha parecido tedioso resolver los puzzles con las piezas de los cuadros?	1
11	Cuando por fin lo ha descubierto, ¿esperaba que fuese el vigilante el ladrón del cuadro?	0
12	¿Le ha parecido satisfactorio ser un detective y resolver el caso del cuadro robado?	8
13	¿Se ha sentido perdido en algún momento del juego?	2
14	¿Volvería a jugar?	7
15	¿Ha aprendido algo de arte con este juego?	1

Gracias por completar el test

9.4. Agradecimientos

Me gustaría dar por concluido este máster con unas pequeñas líneas en las que tener la oportunidad de recordar a todas las personas que han hecho posible de un modo u otro este TFM.

Antes de nada, me gustaría agradecer al doctor Francisco Gutiérrez su orientación, motivación y tutela a lo largo del desarrollo de este proyecto.

También me gustaría recordar a todos los amigos que he conocido durante este año, tanto en clase como en las prácticas. Especialmente, a Juan Carlos Serrano y a Felipe Peiró; sin ellos, este curso no habría sido lo mismo.

Agradecer también a todos los amigos que han estado conmigo desde pequeño; a Álvaro, Jaime, Edu y a todos los demás. Y, por supuesto, a mi historiadora del arte personal.

Y por último, me gustaría dedicarles estas últimas líneas a mis padres, por todo lo que me han enseñado.

Gracias a todos.

Bibliografía

[nex, 1997] (1997). So what do they teach you at videogame school? *Next Generation*, 25.

[Aguado et al., 2008] Aguado, G. et al. (2008). *Organización y gestión de la empresa informativa*. Síntesis.

[Bethke, 2003] Bethke, E. (2003). *Game Development and Production*. Wordware Publishing.

[Boehm, 1979] Boehm, B. (1979). Software engineering as it is.

[Brockwell, 2016] Brockwell, H. (2016). Forgotten genius: the man who made a working vr machine in 1957.

[Cadena, 2008] Cadena, R. (2008). Diseño e implementación de un motor de realidad virtual escalable para escenarios 3d.

[Dille and Zuur, 2007] Dille, F. and Zuur, J. (2007). *The Ultimate Guide to Video Game Writing and Design*. Lone Eagle Publishing Company.

[Fernández et al., 2002] Fernández, R. et al. (2002). De la realidad virtual a la realidad aumentada.

[Geig, 2018] Geig, M. (2018). *Unity 2018 game development in 24 hours*. Pearson.

[Glover, 2018] Glover, J. (2018). *Unity 2018 Augmented Reality projects*. Packt.

[James and Walter, 2010] James, M. and Walter, L. (2010). SCRUM reference card. Last checked on 26/05/2017.

[Karamian, 2018] Karamian, V. (2018). *Bulding an RPG in Unity 2018*. Packt.

[Linowes, 2018] Linowes, J. (2018). *Unity Virtual Reality Projects*. Packt.

- [Lomba, 2013] Lomba, C. (2013). El paraninfo, símbolo de la cultura universitaria.
- [López, 2018] López, V. (2018). La realidad virtual como recurso educativo en las ciencias experimentales.
- [Manrubia, 2014] Manrubia, A. M. (2014). El proceso productivo del videojuego: fases de producción.
- [Mazuryk and Gervautz, 1999] Mazuryk, T. and Gervautz, M. (1999). Virtual reality - history, applications, technology and future.
- [Morales et al., 2010] Morales, G. et al. (2010). Procesos de desarrollo para videojuegos.
- [Murray, 2017] Murray, J. (2017). *Building Virtual Reality with Unity and Steam VR*. CRC Press.
- [Ramírez, 2010] Ramírez, J. A. (2006-2010). *Historia del Arte*. Alianza.
- [Rouse III, 2005] Rouse III, R. (2005). *Game Design: Theory & Practice*. Wordware Publishing.
- [Schell, 2017] Schell, J. (2017). *Virtual Reality blueprints*. Packt.
- [Schnipper, 2017] Schnipper, M. (2017). *Seeing is Believing: The State of Virtual Reality*. The Verge.
- [Schwaber and Sutherland, 2013] Schwaber, K. and Sutherland, J. (2013). The scrum guide.
- [Smith, 2018] Smith, M. (2018). *Unity 2018 cookbook*. Packt.
- [Sutherland, 1965] Sutherland, I. (1965). A head-mounted three dimensional display.
- [Thorn, 2018] Thorn, A. (2018). *Unity 2018 by example*. Packt.