Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

MÁSTER UNIVERSITARIO EN DESARROLLO DE VIDEOJUEGOS

Trabajo Fin de Máster

“Rutas patrimoniales en La Laguna, ambientadas en el siglo XVI, para dispositivos Meta Quest 2. Gestión y Optimización de un proyecto en Realidad Virtual.”

“Heritage routes in La Laguna, set in the 16th century, for Meta Quest 2 devices. Management and Optimization in a Virtual Reality Project.”

Autor: Alberto Fariña Barrera

La Laguna, 7 de septiembre de 2022.

Dña. **Isabel Sánchez Berriel**, con N.I.F. 42.885.838-S profesora Contratada Doctora adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora.

D. **Fernando Andrés Pérez Nava**, con N.I.F. 42.091.420-V profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor.

C E R T I F I C A (N)

Que la presente memoria titulada:

“Rutas patrimoniales en La Laguna, ambientadas en el siglo XVI, para dispositivos Meta Quest 2. Gestión y Optimización de un proyecto en Realidad Virtual” ha sido realizada bajo su dirección por D. Alberto Fariña Barrera, con N.I.F. 43.836.555-G.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de septiembre de 2022.

Agradecimientos

A Míriam,

por su apoyo, por tener siempre las palabras exactas para sacarme de momentos bajos y compartir conmigo los momentos altos.

A mis padres y familia,

por apoyarme y animarme a hacer lo que me gusta, aunque sigan diciendo que estar delante del ordenador es jugar.

A mis amigos,

por entender que me perdiera tantos planes los últimos meses y aún seguir estando ahí para lo que necesitara.

A los profesores Isabel y Fernando,

por su dedicación y su entrega, por siempre tener un momento para un consejo y por su paciencia durante la elaboración del proyecto.

Al resto de profesores del máster,

por la pasión que transmiten y por hacernos sentir una pequeña familia entre los miembros de este máster.

Resumen

La ciudad de San Cristóbal de La Laguna fue pionera en un modelo de urbanismo colonial sin murallas y centrado en el comercio. El mapa de la ciudad elaborado por Leonardo Torriani permite el estudio de dicho urbanismo que se convirtió en referente en América Latina.

Por otro lado, la situación de pandemia de los últimos años ha sido clave en el impulso de la tecnología de Realidad Virtual, para permitir a las personas en cuarentena salir de sus hogares al menos virtualmente.

El departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna ha aunado ambos conceptos y elaborado una experiencia en Realidad Virtual que permite al usuario viajar a la ciudad de La Laguna del siglo XVI y verla tal y como era en aquel entonces usando lo plasmado por Torriani en su mapa. El proyecto "Reconstrucción Histórica Virtual de San Cristóbal de La Laguna" junta a profesionales de informática, diseño, historia y demás ámbitos para crear la mejor experiencia histórica virtual para el usuario.

El objetivo de este trabajo de fin de máster consiste principalmente en aportar a dicho proyecto, en la parte de optimización y mejora de rendimiento del juego para el hardware objetivo que son las Meta Quest 2. El trabajo derivó en una optimización más enfocada a la gestión y organización del proyecto, donde se contempla el despliegue del mismo en el servicio de control de versiones Perforce y la elaboración de guías de uso para facilitar la adición de futuros recursos. También se han realizado optimizaciones más tradicionales en el ámbito de las texturas y el Occlusion Culling, así como mejoras más generales en la ruta y audios del guía o en la aparición de los personajes.

Palabras clave: San Cristóbal de La Laguna, realidad virtual, mapa, Torriani, patrimonio cultural, optimización, Meta Quest 2.

Abstract

The city of San Cristóbal de La Laguna was pioneer in a no walled colonial urbanism model and focused on commerce. The map of the city created by Leonardo Torriani allows the study of that very same urbanism that became standard in many cities of Latin America.

On the other hand, the pandemic situation lived this lasts years was key in the boost of virtual reality technology, allowing people in quarantine to go outside at least virtually.

The department of Computer and Systems Engineering of the University of La Laguna has put together those concepts and created a virtual reality experience that allows any person to travel to the sixteenth century city of La Laguna and contemplate how it was then, using the map created by Torriani as a reference. The project “Virtual Historical Reconstruction of San Cristóbal de La Laguna” is made by professionals from computing, design, history and other fields to create the best possible virtual historical experience for the user.

This Master’s Thesis consists of contributing to the project trying to optimize its performance for the target hardware, the Meta Quest 2 virtual headset. The project resulted in an optimization more focused in management and organization like the deployment of the project on Perforce Virtual Control System and the creation of guides to facilitate the first steps of future collaborators. Classic optimizations were also made in textures and applying Occlusion Culling as well as more general improvements in the sounds and route of the guide and character spawns.

Keywords: San Cristóbal de La Laguna, virtual reality, map, Torriani, cultural heritage, optimization, Meta Quest 2.

Índice

Capítulo 1
Introducción

En el siglo XVI, el arquitecto, ingeniero y cartógrafo militar Leonardo Torriani [1], proveniente de Italia, realizó uno de los mapas más relevantes de la historia de las Islas Canarias: El plano de la ciudad de San Cristóbal de La Laguna (figura 1). Este plano muestra el trazado de las calles de la ciudad capitalina en aquel entonces, así como la localización de los lugares más importantes como centros de culto, hospitales y casas de gobierno. Gracias a la conservación de este mapa, historiadores de las islas y de fuera de ellas han podido entender mejor su planteamiento como ciudad colonial no fortificada, con un diseño abierto al comercio y lejos de la concepción bélica de otras épocas. Además, se ha reconocido su influencia en el desarrollo del diseño de varias ciudades en el continente americano.

Figura 1. Archivo: La Laguna Torriani 1590.jpg.
(fuente: https://es.wikipedia.org/wiki/Archivo:La_Laguna_Torriani_1590.jpg)

La Laguna, como se la conoce coloquialmente, fue la primera capital de la isla de Tenerife tras la conquista española y en el año 1999 se le otorgó el título de Patrimonio de la Humanidad por la UNESCO [2] al conjunto histórico de la ciudad, destacando la conservación de su trazado original y la conservación de sus palacios, iglesias, conventos, calles y casas tradicionales. También es extraoficialmente la capital cultural de la isla, no solo por ser Patrimonio de la Humanidad sino también por albergar la primera universidad del archipiélago,

que en la actualidad cuenta con más de 200 años de historia y más de 30.000 alumnos [3].

Otra parte importante del proyecto es la realidad virtual. La situación de los pasados años durante la pandemia del COVID-19 y el confinamiento consecuente provocó que la tecnología de la realidad virtual se multiplicara como forma de salir de la rutina de dicho confinamiento dentro del hogar. Actualmente, la reducción del coste de los dispositivos utilizados en Realidad Virtual, así como la aparición de algunos modelos capaces de funcionar en un modo independiente que no limitan el movimiento del usuario, han aumentado considerablemente la popularidad de esta tecnología (figura 2).

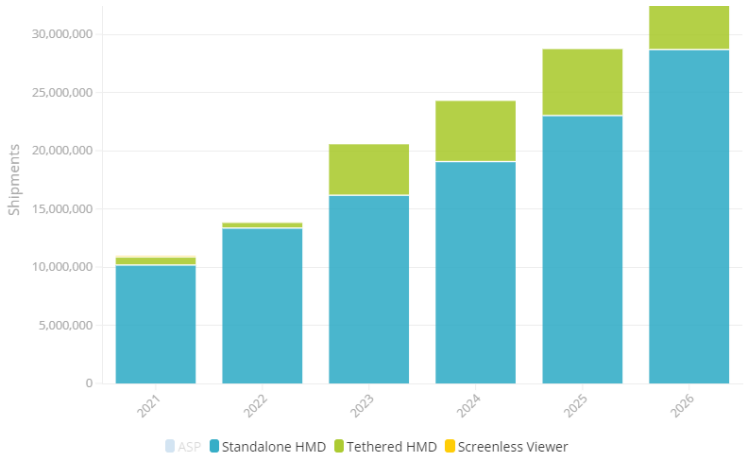

Figura 2. Previsión de crecimiento en las ventas de dispositivos VR por segmento. (fuente: «Meta's Dominance in the VR Market will be Challenged in the Coming Years» [4])

Con los dos conceptos anteriores en mente, el departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna en colaboración con otros departamentos de la universidad, como pueden ser el de Bellas Artes y el de Geografía e Historia, comienzan la creación de un proyecto en realidad virtual que funcione como una guía interactiva a través de una reconstrucción histórica de la ciudad de San Cristóbal de La Laguna. Para ello se ha creado un escenario de la ciudad derivado del ya nombrado plano realizado por Leonardo Torriani en 1588. La aplicación introduce al usuario en la representación de la ciudad en esta época y le permite realizar un viaje en el tiempo que lo llevará no solo a visitar dicho entorno, sino también a disfrutar de visitas guiadas que le permitan conocer los datos y curiosidades de la misma de primera mano.

Para este Trabajo de Fin de Máster, se parte de la aplicación en desarrollo para el proyecto de Reconstrucción Histórica Virtual de San Cristóbal de La Laguna. El

trabajo que se llevará a cabo será conocer los pormenores del estado actual del proyecto e intentar aportar optimizaciones en el ámbito de la gestión y documentación, así como mejoras extras que surjan durante el desarrollo, tras la pertinente aprobación del equipo principal del proyecto. También será apropiada cualquier corrección de errores que puedan surgir durante el desarrollo o errores que se detecten durante la familiarización con el proyecto.

También destacar que este trabajo ha sido llevado a cabo paralelamente y en colaboración con el Trabajo Fin de Máster del compañero Jonathan Merayo de Caso. Su aportación al proyecto fue también imprescindible para poder alcanzar el objetivo propuesto.

1.1. Antecedentes

La Realidad Virtual (VR) como la conocemos en la actualidad se remonta a la Segunda Guerra Mundial y la creación de simuladores de vuelo para entrenamiento de pilotos de bombarderos. Su relación con el mundo de los videojuegos y el entretenimiento comenzó a finales de los 80 cuando Sega[5] y Nintendo[6] en la gran batalla que mantenían apostaron por lanzar cascos de realidad virtual con lentes de obturador y la más famosa y posterior Virtual Boy[7] del gigante nipón en 1995.

Sin embargo, el inicio de la Realidad Virtual moderna y el gran avance que está experimentando da comienzo en 2012 cuando Palmer Luckey, CEO de Oculus, presenta la campaña de mecenazgo de Oculus Rift[8], que comienza a comercializarse en 2015.

Tras esto, un gran número de marcas se embarcó también en la realidad virtual centrada en el entretenimiento, como Sony con la Playstation VR[9] o Valve y HTC que lanzaron conjuntamente las HTC Vive[10].

Los HMD se podrían dividir en tres categorías:

- HMD Independientes: cascos de realidad virtual que pueden ejecutar software sin necesidad de conectarse a PC o consolas. Ejemplo: Oculus Quest 2.
- HMD Dependientes: cascos de realidad virtual que dependen de la potencia gráfica de un PC o consola al que deben estar conectados. Ejemplo: Oculus Rift S (figura 3).
- HMD para dispositivos externos: cascos de realidad virtual preparados para que un dispositivo externo (normalmente un smartphone) sea introducido en los mismos y usado como pantalla. Ejemplo: Google Cardboard

Hemos visto, en los últimos años, el despegue lento pero seguro de esta tecnología, capaz de hacernos percibir una realidad alternativa a la que estamos acostumbrados, tanto visual, como auditiva o háptica.

Figura 3. Oculus Rift S
(fuente: https://versus.com/es/oculus-rift-s)

El software específico de juegos para VR intentó innovar en cuanto a usabilidad y mecánicas y no ser simplemente una versión de un juego generalista en el que puedes mover la cabeza con un casco (aunque también existen). Varios ejemplos pueden ser juegos de ritmo como Beat Saber[11] que aprovecha el movimiento de los brazos para seguir el ritmo de una canción eliminando cajas. Otro ejemplo curioso es una versión de un juego de sobra conocido como Tetris. Tetris Effect VR [12] intenta crear una experiencia sensorial mientras se juega a tetris, con piezas que caen al ritmo de la música, y entornos creados con millones de partículas que acrecientan la experiencia.

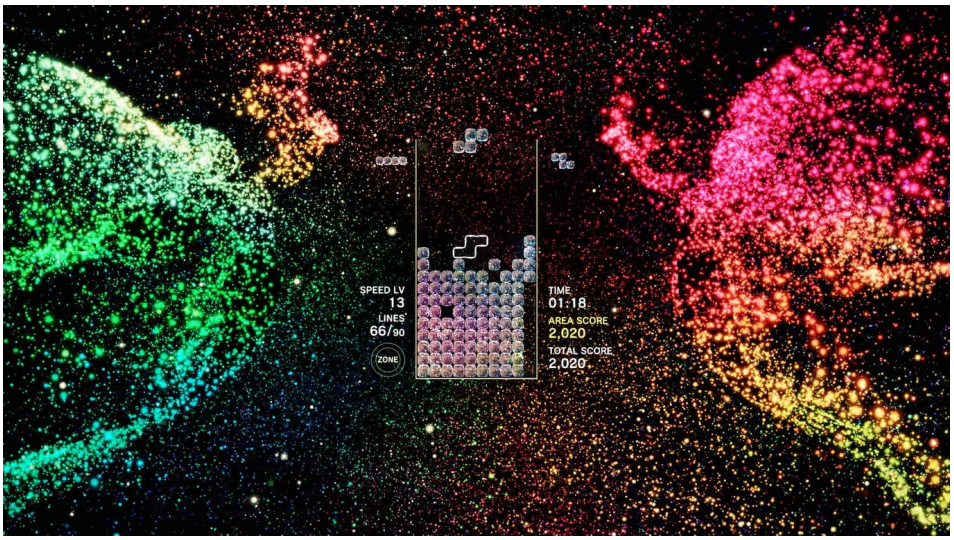

Figura 4. Captura de pantalla del videojuego Tetris Effect
(fuente: https://www.vidaextra.com)

Más específicamente con el proyecto que nos incumbe, la aplicación cultural y turística de la realidad virtual es más común cada día que pasa [13]. Lo más normal es su uso para permitir el acceso a patrimonio y arte desde cualquier parte del mundo de una manera inmersiva, ya que no es lo mismo ver una imagen estática de una obra de arte en un museo, que estar en esa sala del museo, girar la cabeza y poder ver todas las obras que te rodean. Un gran ejemplo de esto son las visitas virtuales inmersivas del Museo Nacional Thyssen-Bornemisza [14].

1.2. Contexto Histórico y Justificación

Como se ha mencionado anteriormente, la ciudad de San Cristóbal de La Laguna posee un diseño muy característico que la ha hecho un referente para otras ciudades de latinoamérica.

El plano de Torriani anteriormente comentado muestra la ciudad así como su entorno natural, entre los que se encuentra la laguna que da nombre a la ciudad. También es destacable la leyenda que se muestra en la esquina inferior derecha que indica los lugares más importantes de la ciudad, siendo la mayoría de ellos lugares de culto. La tabla 1 muestra dicha leyenda de manera más legible.

Leyenda de puntos de interés del mapa de Torriani	
(A) Los Remedios (Catedral)	(M) Hospital
(B) La Concepción	(N) San Sebastián
(C) San Agustín	(O) San Juan
(D) Santo Domingo	(P) San Cristóbal
(E) San Francisco	(Q) San Roque
(F) Ermita de San Miguel	(R) Santa Clara
(G) Plaza del Adelantado	(S) Villa Vieja
(H) Plaza de los Remedios	(T) Fuente Seca
(I) Entrada de Agua	(V) Molino
(L) Salida de Agua	(X) Casas del Cabildo

Tabla 1. Leyenda de puntos de interés del mapa de Torriani (1588)

Teniendo disponibles los datos anteriores y viendo las posibilidades que nos ofrece la realidad virtual de cara a una inmersión total en cualquier tipo de entorno, es una idea bastante interesante, académica y culturalmente, utilizar la tecnología disponible a nuestro alcance para virtualizar la ciudad de La Laguna de la época y dar la posibilidad, a cualquier persona que lo desee, de transportarse a otro tiempo a conocer una ciudad referente para muchas otras, visitar sus calles y conocer su arquitectura, el origen de sus edificaciones más emblemáticas y las vestimentas y elementos patrimoniales característicos de este entorno.

La aplicación desarrollada en el proyecto “Reconstrucción Histórica Virtual de San Cristóbal de La Laguna” está actualmente en un estado avanzado de desarrollo. Los esfuerzos de este Trabajo de Fin de Máster irán encaminados principalmente a utilizar los últimos avances tecnológicos y los conocimientos adquiridos en el desarrollo del proyecto para hacer crecer la aplicación, corrigiendo errores, optimizando su funcionamiento para la plataforma de destino elegida (Meta Quest 2) y realizando las actualizaciones y añadidos que se vean convenientes.

1.3. Objetivo General

El proyecto de este Trabajo de Fin de Grado se centra en mejorar el proyecto en su punto actual de desarrollo mediante la gestión del mismo, añadiendo documentación o realizando un despliegue en un sistema de control de versiones

que permita el trabajo colaborativo. Además de optimizar el rendimiento, otras mejoras y sin olvidar el gran esfuerzo de integración en un proyecto ya en desarrollo.

1.4. Objetivos Específicos

Los objetivos específicos del Trabajo Fin de Máster se pueden separar en cuatro categorías:

- **Gestión y organización**: Se tratará de mejorar la gestión del proyecto, algo que comprende desde organizar la estructura de carpetas del mismo hasta su despliegue en un sistema de control de versiones. Además se añadirá documentación que permita una incorporación más sencilla de nuevos recursos en un futuro.
- **Optimización**: se tratará de alcanzar un funcionamiento óptimo, conociendo que la plataforma de destino será el dispositivo de realidad virtual "Meta Quest 2"[15] y basándonos en las características técnicas de su hardware y las limitaciones y ventajas que este pueda presentar.
- **Corrección de errores**: Se buscará tratar los errores ya conocidos y referidos por el equipo del proyecto, así como detectar y corregir los posibles errores no conocidos o aquellos que pudiesen surgir de las modificaciones aplicadas por nuestra intervención.
- **Actualizaciones y mejoras**: secundario a los puntos anteriores se abre la puerta a aplicar las actualizaciones, modificaciones y añadidos que se consideren adecuados, consensuado con el equipo del proyecto original.

1.5. Estado Actual del Tema

En el estado actual de desarrollo de la aplicación desarrollada en el proyecto de Reconstrucción Histórica Virtual de San Cristóbal de La Laguna se dispone de:

- **Modelos de las edificaciones**: tanto de casas y mobiliario de la época, como modelos detallados de los edificios singulares representativos de la ciudad. Se inicia este trabajo con versiones no definitivas de los modelos, las versiones estables de los mismos, se han ido integrando durante el periodo de desarrollo de este trabajo.
- **Diferentes modos de navegación por el modelo**: el juego dispone de varias opciones de navegación por el mismo, con opción de movimiento básico y libre por toda la ciudad, opción de "viaje rápido" mediante mecánicas de teletransporte a puntos de interés y opción de movimiento siguiendo a un guía.

- **Mecánicas de selección de escenarios**: para reducir la carga de la ejecución del proyecto, se ha separado este en una escena principal que contiene el terreno, entorno, lógica del juego, etc... y que permanecerá siempre cargada, y 6 escenas con carga aditiva de elementos individuales como edificaciones, modelos de personajes y puntos de interés. La gestión de carga de una u otra escena se ejecuta mediante el uso de un mapa interactivo.
- **Modelos de personajes con vestimentas propias de la época**: diseñados con un alto nivel de detalle. Algunas con animación y otras que se pueden visitar en un museo virtual.

Capítulo 2
Herramientas utilizadas

En este capítulo se documentan todas las herramientas usadas durante el desarrollo de este trabajo de fin de máster, dividiéndolas en herramientas de software y herramientas de hardware.

2.1. Herramientas de Software

Las herramientas de software abarcan todos los programas informáticos usados en el proyecto para cualquier tipo de acción. Desde lo más común como el motor gráfico utilizado para el desarrollo, el IDE para la escritura del código, hasta el sistema de control de versiones o Oculus Developer HUB, el programa propietario de Meta para los desarrolladores interesados en su sistema.

2.1.1. Unity (versión 2019.4.9f1)

Unity[16] es un motor de videojuegos multiplataforma creado por Unity Technologies y disponible para Windows, Mac OS y Linux. La empresa comenzó siendo una empresa de desarrollo de videojuegos, pero tras el fracaso de su primer lanzamiento, reconocieron el valor de las herramientas de desarrollo que habían creado y decidieron centrarse en evolucionarlas y desarrollar un motor asequible y que cualquiera pudiera usar. Su primera versión se lanzó en 2005.

Este proyecto utiliza la versión 2019.4.9f1 de Unity. Este motor parece la elección obvia para este proyecto, ya que siendo un proyecto de la universidad y contando la universidad con licencia PUE[17], permite el acceso a material educativo exclusivo que puede ser de gran utilidad para colaboradores y los propios profesores encargados del proyecto, así como tener la posibilidad de realizar exámenes de certificación.

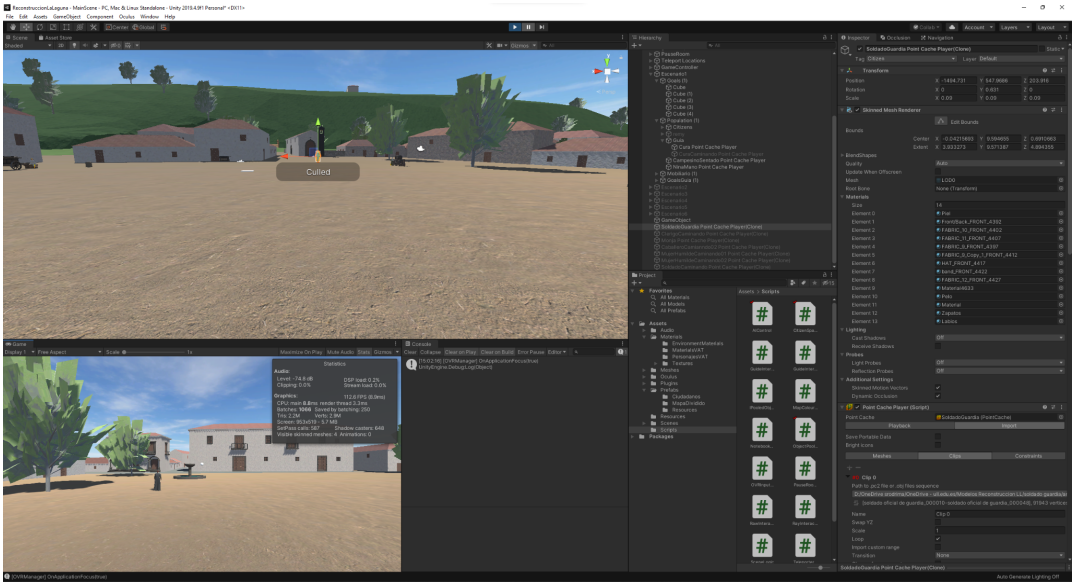

Figura 5. Pantalla del editor de Unity.
(Fuente: Elaboración propia)

Al ser un proyecto en un punto intermedio de desarrollo no hubo elección de motor ni de versión por nuestra parte ya que se tuvo que continuar con la que ya se estaba usando. Se intentó actualizar el proyecto a una versión más reciente de Unity para acceder a varias mejoras en el ámbito de la Inteligencia Artificial y mejoras generales que pueden mejorar el rendimiento del programa, pero fue imposible debido a ciertas dependencias del proyecto con la versión utilizada. Pero aparte de eso, no ha habido mayor problema ya que la versión usada (2019.4.9f1) es una versión LTS (Long Time Support), por lo que Unity asegura un soporte a largo plazo de dicha versión. En la figura 5 se puede observar una vista del editor de Unity que se ha usado para este proyecto.

A parte del motor en sí, se han usado ciertos añadidos al mismo, como assets, herramientas, plugins, etc que han extendido las funcionalidades del motor y que han permitido resolver con una mayor facilidad problemas que de otra forma hubieran conllevado un gran tiempo de estudio y corrección:

- **Vertex Animation Tools (VAT)**[18]: Asset que permite optimizar la gestión de los vértices de las mallas de las vestimentas animadas de los personajes.

- **XR Plugin Management y Oculus XR Plugin**[19]: plugins imprescindibles, el primero para la creación de proyectos en realidad virtual y el segundo para la integración de proyectos de realidad virtual con gafas Oculus (Actual Meta).

2.1.2. Microsoft Visual Studio Code y C#

Para el desarrollo de software en Unity es común el uso del lenguaje C# y de un IDE compatible.

En primer lugar, como IDE se usó Microsoft Visual Studio Code[20] o VS Code para abreviar. A pesar de que mucha gente lo considera un IDE o Entorno de Desarrollo Integrado, Microsoft se refiere a él como un editor de código ligero, al contrario que su hermano mayor Microsoft Visual Studio[21] que sí es considerado un IDE. Ambos programas fueron creados para la integración con la plataforma .NET[22] de Microsoft y edición de código para el mismo. Puesto que la compilación del código para este proyecto es realizada por el propio motor Unity cada vez que un script es modificado, no fue necesario usar una herramienta tan pesada como Microsoft Visual Studio, por ello se optó por una opción más ligera como VS Code.

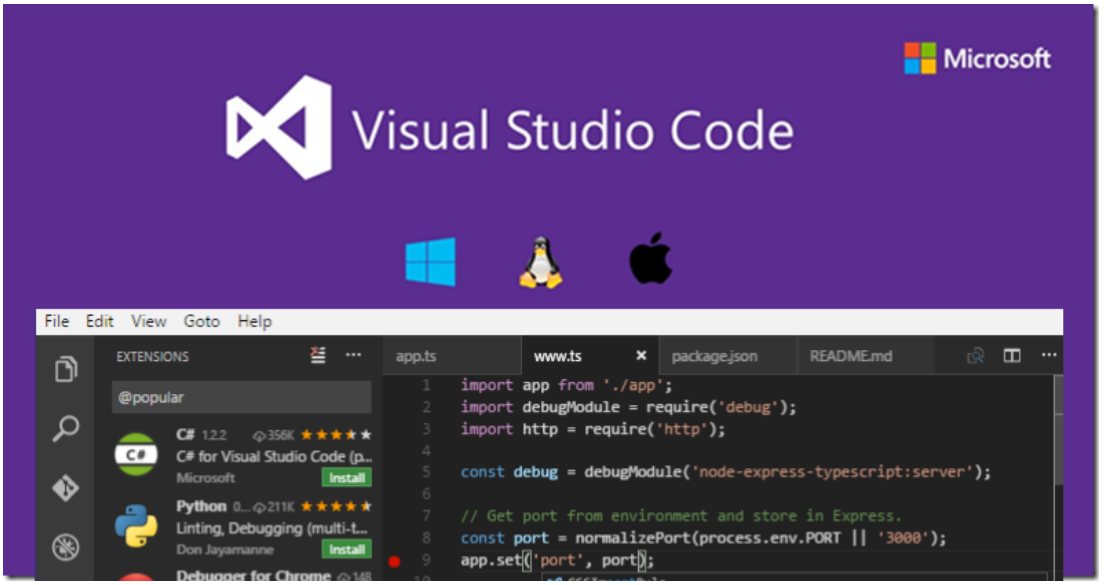

Figura 6. Banner publicitario de Visual Studio Code.
(Fuente: https://openexpoeurope.com)

VS Code permite la adición de plugins para labores concretas, entre ellas existe un plugin para el desarrollo en Unity, que incluye ayudas y autocompletado para hacer más cómodo el desarrollo para este motor de videojuegos.

C#[23] (pronunciado C Sharp) es un lenguaje de programación que, al igual que VS Code, fue desarrollado por Microsoft para su uso en su plataforma .NET. Su sintaxis básica deriva de C[24] y C++[25], pero utilizando el modelo de objetos de la plataforma .NET que es similar al de Java[26] e incluyendo a la vez mejoras derivadas de otros lenguajes. Es el lenguaje principal para el desarrollo en Unity, junto con la librería MonoBehaviour[27] de la cual heredan la gran mayoría de clases creadas para un juego en Unity.

2.1.3. Android Studio y Android SDK Manager

Android Studio[28] es el IDE oficial desarrollado por Google para la elaboración de software para Android[29] y está basado en el software IntelliJ de JetBrains[30]. A pesar de incluir aquí Android Studio, no fue utilizado como IDE para el desarrollo o escritura de código. Su instalación fue necesaria para el uso de una de las herramientas que incluye y que es imposible descargar aparte: Android SDK Manager[31].

Figura 7. Logo de Android Studio.
(Fuente: https://www.xatakandroid.com)

El hardware objetivo de este proyecto, las Meta Quest 2, usan un sistema operativo propietario pero basado en el código fuente de Android, por lo que para probar una aplicación externa en el mismo es necesario empaquetarlo como un APK para Android. Android SDK Manager fue necesario para instalar en el sistema operativo local, el SDK de la versión de Android concreta a la que apunta nuestro proyecto para la correcta construcción del mismo (versión del SDK 26).

2.1.4. Perforce Helix Core y Perforce P4V

La compañía de desarrollo Perforce creó a finales de los años 90 el sistema de control de versiones Helix y que actualmente se conoce como Helix Core[32]. Este software está enfocado al control de versiones para proyectos de grandes dimensiones.

Figura 8. Logo de Helix Core.
(Fuente: https://www.perforce.com/products/helix-core)

Este sistema de control de versiones, o VCS según sus siglas en inglés, utiliza una base de datos central, en la que se almacenan los proyectos y a la que se conectan los clientes. Los clientes podrán hacer 'checkout' sobre un fichero y editarlo si no está bloqueado para su uso por otros usuarios, impidiendo una doble edición del mismo fichero. Esto puede causar problemas de organización si un usuario se olvida de liberar un fichero tras su uso, pero es una manera segura de evitar la corrupción de un fichero derivada de dos personas editándolo a la vez, y los fallos de bloqueo se pueden evitar haciendo un uso correcto del software y estableciendo unos protocolos.

Se eligió este software para alojar el proyecto debido al gran tamaño del mismo, que hacía imposible su alojamiento en otras herramientas VCS de uso más extendido. Además, la posesión de una licencia de uso por parte de la Universidad hace su acceso más idóneo.

Como programa cliente para el uso de Perforce Helix Core se usa Perforce P4V[33], un programa cliente visual que permite gestionar todo lo relacionado con checkouts y subidas de ficheros al repositorio principal de manera fácil y visual.

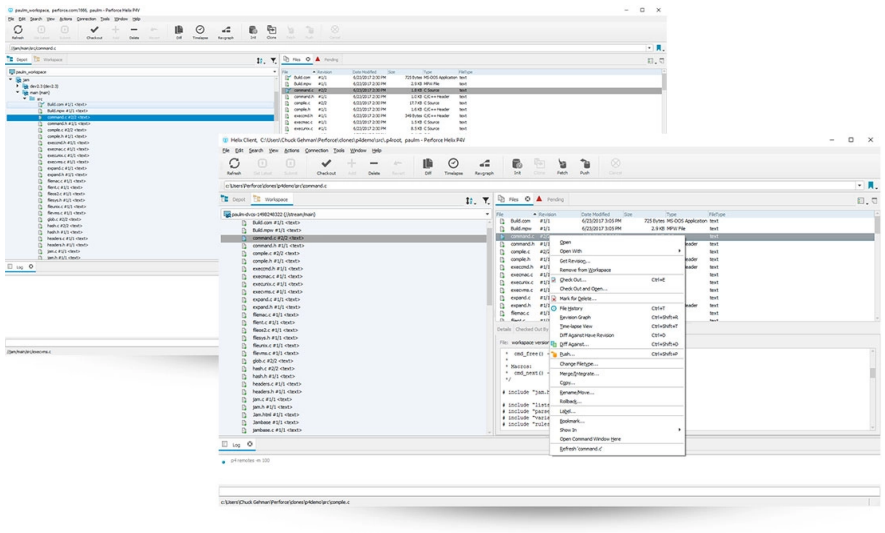

Figura 9. Interfaz de usuario del cliente P4V para Perforce.
(Fuente: https://www.perforce.com/products/helix-core-apps/helix-visual-client-p4v)

2.1.5. Audacity

Audacity[34] es el software open source y gratuíto de referencia para edición y grabación de audio digital. Permite desde ediciones básicas como recortar partes de un audio o amplificar el volumen hasta operaciones más complejas como análisis de espectro de audio o reducción de ruido, así como la exportación final a un fichero mp3[35].

Este software ha sido usado para la creación de ficheros de audio para las paradas concretas del guía. El propio software soporta la grabación de la voz para las explicaciones del guía y posteriormente, usando algoritmos de reducción de ruido y amplificación del volumen, se han creado ficheros de audio lo más profesionales posibles, a la espera de que locutores expertos graben los audios finales.

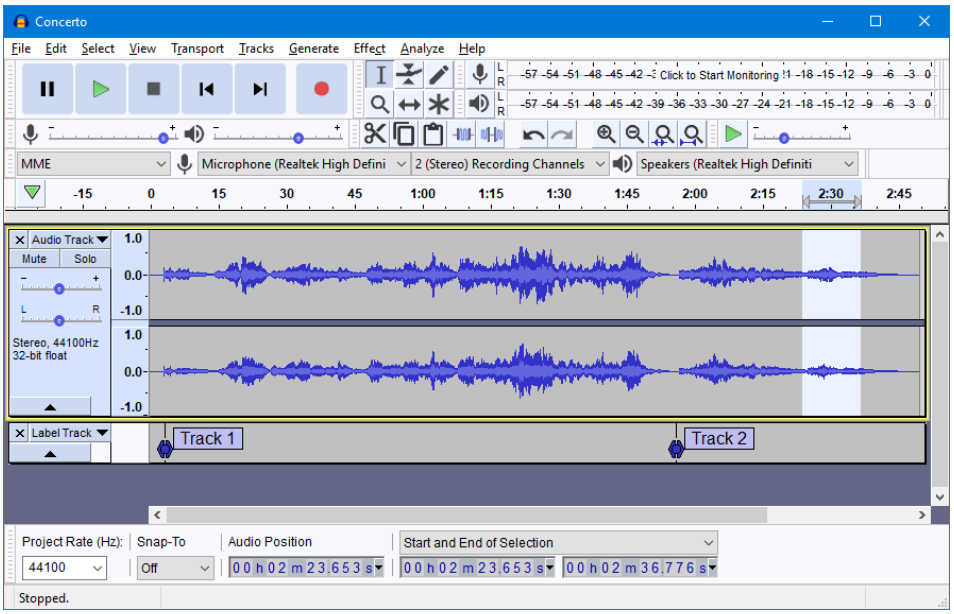

Figura 10. Interfaz de usuario de Audacity.
Fuente(https://www.hispasonic.com)

2.1.6. Blender

Existe una gran variedad actualmente de programas que permiten realizar modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales, pero el referente en cuanto a software libre y de código abierto es Blender[36].

En este programa creado por la "Blender Foundation" se han modelado todos los assets propios elaborados para el proyecto cómo casas, árboles o edificios emblemáticos de la ciudad de La Laguna, para su posterior importación desde Unity para crear las escenas y entornos de la ciudad.

2.1.7. Oculus Developer Hub

Oculus Developer Hub[37] es el software propietario de Meta utilizado principalmente para enlazar nuestro dispositivo de realidad virtual de esta misma marca con un PC mediante conexión por cable o conexión WIFI. Permite gestionar las actualizaciones de software del dispositivo, facilitar las iteraciones de desarrollo, así como proveer datos de análisis para evaluar el rendimiento que ha tenido determinado software en la plataforma.

2.2. Herramientas de Hardware

El punto que concierne a las herramientas de hardware no será tan extenso como las herramientas de software, ya que solo se cubrirá el hardware objetivo de este proyecto, las gafas de realidad virtual Meta Quest 2.

2.2.1. Meta Quest 2

El visor de realidad virtual Meta Quest 2[38] fue lanzado al mercado como Oculus Quest 2 en octubre de 2020 y rebautizado un año más tarde. Su principal característica es que puede ser usado como un visor independiente sin necesidad de ser conectado a otro dispositivo que le proporcione potencia gráfica suficiente para desarrollar su función. Por esto, no es tan potente como otras versiones de visores conectados, pero su movilidad y uso sin cables compensan este hecho con creces. A pesar de esto, también existe la opción de conectar el visor a una computadora externa usando Wifi o un cable USB especial para aumentar su potencia en caso de que sea necesario. Su sistema operativo interno está basado en Android, algo que será importante en el caso de querer desarrollar un producto para las mismas.

Figura 11. Gafas de Realidad Virtual Meta Quest 2.
(Fuente: https://www.eurogamer.net)

El visor implementa muchas mejoras con respecto a su versión anterior, como un peso más ligero, pantallas con mejor resolución y mejor tasa de refresco de

pantalla y controladores mejorados tanto en especificaciones como en duración de la batería. Sin embargo recibió críticas en su lanzamiento relacionadas sobre todo con la obligatoriedad de hacer login con una cuenta de Facebook para su uso.

Las especificaciones en detalle de las Meta Quest 2 son las que se muestran en la tabla 2.

Meta Quest 2 - Especificaciones	
Chipset	Qualcomm Snapdragon XR2
CPU	Qualcomm Kryo 585 de 8 núcleos y 8 hilos
GPU	Qualcomm Adreno 650
RAM	6GB LDDR5 2750 MHz
Almacenamiento	Versiones de 128GB y 256GB UFS.
Pantalla	IPS LCD con resolución 1832x1920px por ojo y 120Hz
Sistema Operativo	Oculus Mobile basado en Android 10
Conectividad	Bluetooth 5.0 LE, WIFI6, Jack 3.5 y USB-C
Tracking	Oculus Insight con 4 cámaras y 6 grados de libertad
Entrada	Dos controladores Oculus Touch, uno para cada mano
FOV	110°

Tabla 2. Características técnicas de las gafas de realidad virtual Meta Quest 2.

Capítulo 3
Desarrollo del proyecto

En este capítulo se detallarán todos los cambios, mejoras, añadidos o correcciones de errores que se han realizado en el proyecto. Se ha dividido todo el trabajo realizado en tres secciones:

- Documentación y Producción
- Optimización
- Actualizaciones y mejoras de contenido

Se analizarán estos apartados en este orden además de repasar las correcciones de errores que se han realizado dentro de cada uno de ellos.

3.1. Gestión y Documentación

A pesar de empezar siendo un proyecto de optimización de software, tras la tarea de aprendizaje y adaptación a los elementos ya incorporados al proyecto, el objetivo derivó de manera orgánica a una optimización más centrada en la gestión, optimizaciones que permitan un trabajo más fácil, menores tiempos de adaptación para futuros colaboradores y la posibilidad de colaboración directa entre los mismos. Entre estas labores se encuadran la organización de la estructura de carpetas del proyecto para un mejor entendimiento de los ficheros usados, así como labores de gestión relacionadas con el despliegue del proyecto en un servidor de Perforce.

3.1.1. Aprendizaje del proyecto

Una parte importante del proyecto y la primera tarea que se realizó es asimilar todos los conocimientos que vienen incorporados en un proyecto que ya se encuentra en un estado intermedio de desarrollo.

El proyecto ya contaba con una gran cantidad de sistemas implementados, como distintas escenas para distintas partes de la ciudad para intentar reducir la carga o el funcionamiento del plugin VAT que permite reducir la carga de las animaciones de los trajes.

Este aprendizaje también sirvió para detectar errores y establecer propuestas de posibles mejoras que se pudieran implementar.

3.1.2. Organización en carpetas del proyecto

Durante el proceso de adaptación al proyecto, una de las labores más costosas fue probablemente acostumbrarse a la ubicación de los assets en la estructura de carpetas del mismo. La estructura del proyecto era muy poco adecuada. Las carpetas de plugins se mezclaban con carpetas de mallas o materiales de mallas similares ubicados en lugares distintos.

Para ello se decidió investigar y buscar una organización de carpetas estándar que dejar reflejada en un documento del proyecto y que futuros colaboradores y los propios profesores encargados del mismo puedan usar para mantenerlo organizado.

Se consultaron varios foros especializados de Unity para investigar sobre una organización de carpetas que fuera mejor para este tipo de proyectos[39]. La mayor parte de la discusión estaba entre una organización por tipos, basada en separar los assets entre materiales, texturas, mallas, scripts, etc., usada en mayor medida para proyectos pequeños sin muchos assets. Y por otro lado una organización por relevancia, donde los distintos tipos de assets pertenecientes a cada objeto se encuentren en su propia estructura de carpetas.

La estructura que se decidió aplicar fue una mezcla de ambos, usando una organización por tipos para assets creados por y para el proyecto, mientras que los assets adquiridos u obtenidos de terceros se ubicaron en su propia carpeta y organizados por relevancia.

A continuación se aplicó dicha organización, siempre comprobando que los assets reubicados siguieran funcionando adecuadamente. A su vez, los assets que se encontraron que estaban inactivos o en desuso se eliminaron, ya que el proyecto pecaba también de un peso muy elevado que era clave disminuir.

La organización básica de la carpeta Assets quedó dividida en las siguientes carpetas (figura 12):

- **Audio**: Ubicación de todos los ficheros de audio que se usarán en el proyecto, desde la voz del guía para las distintas paradas, como posible música ambiente o cualquier audio que se requiera en el futuro.
- **Materials**: Carpeta en la que se ubicarán todos los materiales del proyecto. Dentro de la carpeta Materials se encuentra la carpeta Textures que tendrá todas las texturas de los correspondientes materiales.
- **OldAssets**: Aquí se colocarán assets descartados que no quieran ser eliminados aún por si es necesario hacer backtrack debido a un funcionamiento incorrecto de sus actualizaciones.

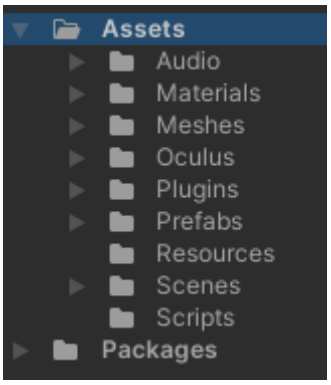

Figura 12. Estructura de carpetas final del proyecto en la raíz de la carpeta 'Assets'.
(Fuente: Elaboración propia)

- **Meshes**: Ubicación de todos los ficheros de mallas en formatos como .FBX y similares.
- **Prefabs**: Carpeta en la que se ubicarán los prefabs creados para el proyecto.
- **Scenes**: Aquí se colocarán los ficheros de las distintas escenas en las que se quiera dividir el proyecto.
- **Scripts**: Carpeta básica que contendrá todos los scripts con código que se usarán en el proyecto.

3.1.2.1 Elaboración de la Guía de Organización

Para mantener las ideas anteriores plasmadas en un documento, se realizó una Guía de Organización que se puede consultar en el Apéndice A de este documento.

3.1.3. Aplicación de un Sistema de Control de Versiones

Otro de los problemas que tenía el proyecto era la falta de una plataforma de trabajo colaborativo en la que varias personas pudieran poner su trabajo común a disposición del otro sin tener que transferir un fichero comprimido de un tamaño considerable.

3.1.3.1. Selección de la plataforma

Se consideraron varias plataformas para usar como sistema de control de versiones antes de decantarse por Helix Core, como Plastic SCM[40] o el propio Github[41]. Sin embargo, para incorporar proyectos de la envergadura del que

estamos tratando en estos sistemas, era necesario utilizar versiones de pago que aumentaran las capacidades que ofrecen de manera gratuíta.

La decisión fácil parecía ser usar Perforce. La universidad cuenta con licencia de uso de este sistema, además de un servidor propio ya instalado donde podríamos ser capaces de alojar todo el espacio que requiere el proyecto.

3.1.3.2 Despliegue del proyecto en Perforce Helix Core

El propio despliegue del proyecto fue bastante sencillo. Los profesores a cargo del proyecto comunicaron al administrador del servidor de Perforce de la universidad nuestra intención de alojarlo allí, y este creó un depot para ello, así como los permisos para las personas involucradas.

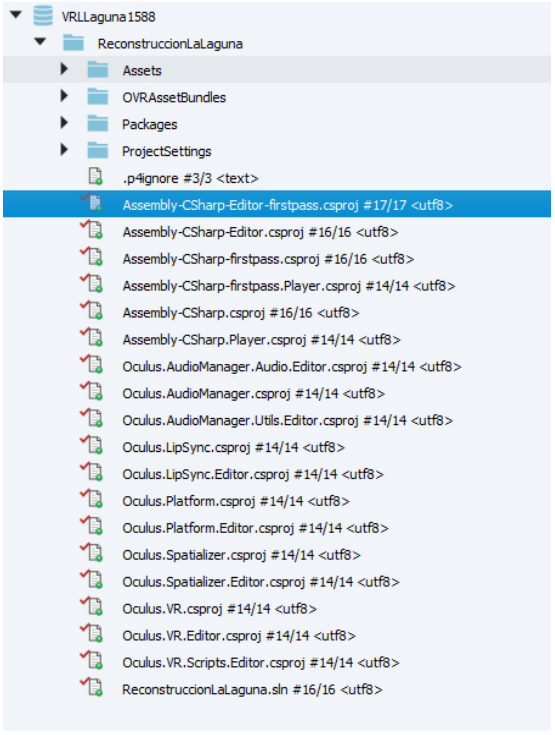

Figura 13. Estructura del proyecto en el depot de Perforce Helix Core.
(Fuente: Elaboración propia)

Con respecto a la preparación del proyecto para su despliegue, se buscaron las carpetas que fueran imprescindibles para compartir un proyecto de Unity en un sistema de control de versiones, permitiendo así alojar solo lo imprescindible y ahorrar espacio[42].

Las carpetas imprescindibles dentro de un proyecto de Unity son:

- Assets
- Packages
- Project Settings

El contenido completo del proyecto se puede ver en la figura 13. El resto de carpetas podrían ser ignoradas y cuando se descargara el proyecto desde Perforce en un nuevo PC, estas carpetas se generarían la primera vez que se abriera el proyecto de Unity. Para ignorar estas carpetas, Perforce tiene un sistema parecido al usado en Git u otros sistemas, un documento 'ignore' que incluya expresiones regulares con las carpetas, extensiones y otros tipos de archivos que se quieran ignorar al momento de la subida. En Perforce se conoce como 'P4ignore'.

3.1.3.3. Integración de Perforce con el motor Unity

Unity contiene ya por defecto una opción de configuración para su integración con varios sistemas de control de versiones. Debido al uso extendido de Perforce dentro del mundo del desarrollo de videojuegos y siendo Unity un motor de desarrollo de videojuegos, uno de los sistemas de control de versiones soportados es, obviamente, Perforce.

Para ello es necesario encontrar el menú de configuración correspondiente y rellenarlo con los datos de nuestro servidor de perforce y el workspace local concreto en el que se encuentra alojado nuestro proyecto, así como nuestro usuario, contraseña, etc. (figura 14).

Figura 14. Ejemplo de identificación en Perforce dentro de los menús de Unity.
(Fuente: Elaboración propia)

Tras esto, Unity se enlazará con dicho sistema y comprobará en tiempo real, por ejemplo, que una persona ya está editando un fichero y nos impedirá editarlo hasta que esta otra persona acabe. O nos solicitará extraer los ficheros del sistema

antes de editarlos, e incluso los extraerá automáticamente, por ejemplo en las escenas cuando hagamos ediciones en ellas.

3.1.3.4 Elaboración de la Guía de Instalación y Uso de Perforce

Para que futuros colaboradores pudieran tener una integración al repositorio de Perforce sencilla y lo más directa posible, se realizó un documento de guía de instalación y uso de la herramienta, que oriente al usuario durante la instalación de los programas necesarios, así como de la integración del motor Unity con Perforce y su posterior uso cotidiano. Se puede encontrar este documento en el Apéndice B.

3.1.4. Corrección de errores

En este apartado se revisarán todas las correcciones de errores que se llevaron a cabo dentro de las labores correspondientes a la gestión y documentación.

3.1.4.1 Error de Build en Unity

El único error considerable que se puede abarcar dentro de este apartado son los varios errores que se encontraron la primera vez que se intentó hacer un build del proyecto y generar un APK[43] para reproducir el juego en las Meta Quest 2.

Como ya se ha comentado, no era un solo fallo, sino varios, así que se irán repasando en el orden que se resolvieron y explicando cómo se resolvieron cada uno.

3.1.4.1.1. Error de Shader de Oculus

El primer error mostraba errores de shaders al no poder leer ciertos scripts que se encontraban dentro de la carpeta del plugin de Oculus.

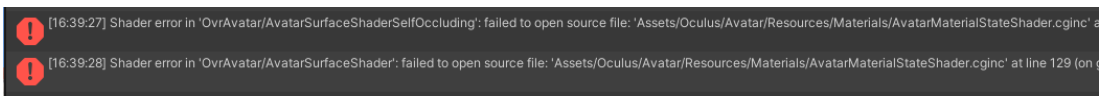

Figura 15. Errores de Shader surgidos en Unity.
(Fuente: Elaboración propia)

Dichos shaders tienen rutas escritas en código directamente para encontrar estos scripts, por lo que había dos soluciones posibles:

- Ubicar la carpeta Oculus en la raíz del directorio 'Assets' para que los shaders encuentren las rutas que ya tienen escritas.

- Modificar las rutas para que se adapten a la nueva organización de carpetas o hacerlas dinámicas.

Se terminó por tomar la primera opción y reubicar la carpeta Oculus para que las rutas la encontraran.

3.1.4.1.2. Error de versión del SDK de Android

Otro de los errores surgidos fue un error derivado de la versión del SDK de Android instalada. A pesar de elegir en Unity una versión objetivo para la compilación, la versión del SDK instalada por defecto no coincidía con la necesaria.

Para solucionar este error fue necesaria la descarga del IDE Android Studio y por consiguiente de su herramienta Android SDK Manager que lleva incorporada. Con ella se pudo descargar la versión correcta del SDK y posteriormente cambiar en Unity la configuración para que buscara el SDK en la carpeta correcta

3.1.4.1.3. Error Win32 IO result 23

El último error obtenido después de solucionar los mencionados en los epígrafes previos mostraba un código de error 'Win32 IO result 23'. Tras buscar por varios foros de dudas de internet se llegó a la conclusión de que estaba causado por la ubicación del proyecto en un disco que no fuera la unidad C del sistema operativo Windows. Al mover el proyecto entero a la unidad C del PC el problema se soluciona.

No se han encontrado motivos por los que sucede este error, pero se sospecha que puede estar relacionado a que el proyecto fue creado por primera vez en una Unidad C de un PC y necesitará estar en alguna unidad C para poder ser construído.

3.2. Optimización

A pesar de mejorar ámbitos como la colaboración y la gestión del proyecto, el trabajo no se iba a limitar a esto y la optimización en cuestiones más técnicas seguía siendo un objetivo. Por ello se investigaron ciertas técnicas, así como maneras inventivas de ahorrar procesamiento dentro del proyecto.

3.2.1. Occlusion Culling

El Occlusion Culling[44] es una técnica por la cual se desactiva la renderización de los objetos que la cámara no puede ver, ya sea porque se encuentran detrás de la misma, o porque otro objeto obstruye su visualización.

Al encontrarnos en un entorno con edificios que con regularidad obstruyen nuestra visión del resto del mapa, esta técnica es muy recomendable. Al contrario, si nos encontráramos en un entorno abierto, de campo, con pocos obstáculos a la vista, donde pudiéramos tener un gran rango de visión, la técnica más óptima sería el Frustum Culling, que solo deja de renderizar los objetos que se encuentran fuera del rango de visión de la cámara, de manera que aunque un objeto quede oculto por otro, mientras se encuentre dentro del rango de visión de la cámara seguirá renderizado pero consumiendo menos recursos que el occlusion culling.

Figura 16. Apariencia de la ciudad antes de la aplicación del Occlusion Culling.
(Fuente: Elaboración propia)

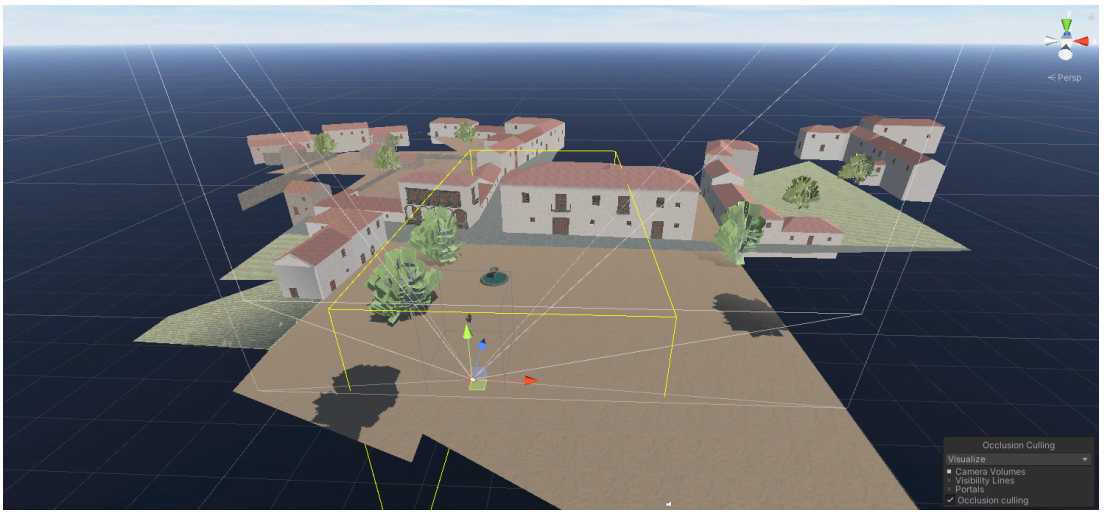

Figura 17. Apariencia de la ciudad tras la aplicación del Occlusion Culling.
(Fuente: Elaboración propia)

Unity tiene una herramienta con la que es posible aplicar el Occlusion Culling de manera rápida y directa. Es necesario seleccionar todos los objetos que queramos que se vean sometidos al Occlusion Culling y crear con dicha herramienta un perfil de oclusión que luego usará el algoritmo de Occlusion Culling para ocultar los objetos pertinentes. Cabe destacar que los objetos que se ocultan mediante esta técnica deben estar indicados como objetos estáticos para que el perfil de oclusión los tenga en cuenta. En las figuras 16 y 17 se puede observar el cambio que provoca la aplicación de dicha técnica.

3.2.2. Optimización del terreno

Relacionado con el tema anterior del Occlusion Culling, cuando se estaba realizando esta técnica, se percató que el objeto que contiene el terreno del mapa, nunca desaparecía en ninguna de sus partes, mientras que otros objetos como las casas o los árboles, si que detenían su renderización correctamente con el Occlusion Culling. El problema estaba causado porque la malla estática que compone el terreno era una sola gran malla de tamaño muy elevado que, obviamente, como la cámara siempre tenía alguna parte del terreno a su vista, nunca desaparecía.

Se solicitó al equipo encargado de la generación de las mallas y del trabajo con Blender 3D dentro del proyecto, si era posible la división de la malla del terreno en partes, ya fueran partes iguales a modo de sectores o dividir la malla en los triángulos simples de los que estaba compuesta. Al ser medianamente plana,

aunque con algún desnivel, no debía de causar una enorme cantidad de divisiones.

Finalmente se decidió a dividir la malla en rectángulos, llegando al resultado esperado, el algoritmo de Occlusion Culling ahora si ocultaba correctamente las partes del terreno que no entraban dentro de la visión de la cámara.

Esto supuso una reducción masiva del número de triángulos renderizados por el proyecto durante su ejecución, pasando de prácticamente 20 millones de triángulos a 3 o 4 millones dependiendo de la orientación de la cámara, una reducción de más del 75% de triángulos en tiempo de ejecución.

3.2.2.1. ¿Terreno por secciones o malla completa?

Cuando se recibió por primera vez el proyecto, el terreno era un objeto completo en la escena principal, pero además, el proyecto estaba dividido en 6 escenas, cada escena ocupando una de las secciones de la ciudad de La Laguna. Por ejemplo, el escenario 1 abarcaba los alrededores de la Plaza del Adelantado, desde la Ermita de San Cristóbal hasta el convento de Santa Clara. Para adecuarse a estos escenarios, el terreno estaba cortado para cada uno de estos escenarios, renderizando solo la zona necesaria.

Todo esto era redundante, ya que el mapa estaba dos veces renderizado, una de manera completa en la escena principal y otra de manera parcial en el escenario correspondiente.

Una vez dividido el terreno completo en porciones que pudieran ser gestionadas por el algoritmo de Occlusion Culling, se vuelve innecesario dividir este terreno en secciones para cada uno de los escenarios. A pesar de ello se pidió al equipo del proyecto realizar la división del terreno en escenarios aplicando los recortes que ya se habían realizado para el terreno completo, reduciendo las divisiones a sus rectángulos para que los gestione mejor el Occlusion Culling.

Tras varias pruebas con ambos terrenos, los resultados fueron prácticamente idénticos, por lo que se decidió mantener un solo terreno completo dividido en rectángulos, debido a la facilidad de gestión de este frente a seis terrenos distintos, ya que el Occlusion Culling permite que a pesar de tener un solo terreno, gran parte de este no esté renderizándose.

3.2.3. Pooling de personajes

Debido a la alta carga que suponen los modelos de los personajes y sus animaciones, otra de las optimizaciones principales que se llevaron a cabo fue

intentar reducir esta carga, y un pooling de dichos modelos era un buen sitio por donde empezar.

Un 'Object Pooling'[45] en desarrollo de videojuegos es un patrón de diseño por el cual una serie de modelos son instanciados desde el comienzo del juego, pero puestos en modo inactivo hasta que se requiera su uso y siempre de manera cíclica, para que no sea necesaria la instanciación de más modelos en tiempo de ejecución, simplemente se activan y desactivan los ya instanciados.

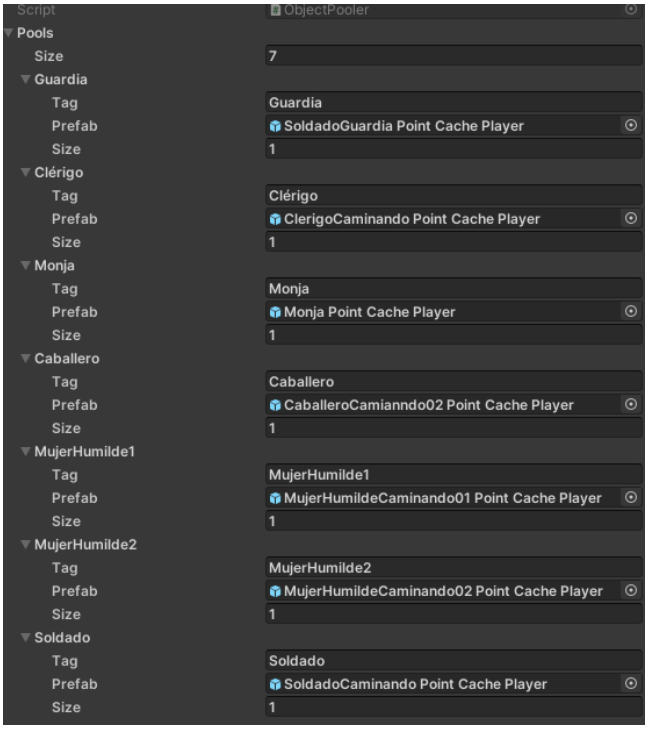

Figura 18. Distintos pools creados para cada uno de los siete modelos de personaje disponibles.
(Fuente: Elaboración propia)

En nuestro caso se dispone de siete modelos de personajes distintos para los cuales se ha creado un prefab para cada uno y cada prefab forma parte de su propio pool (en este caso de tamaño 1), terminando con siete pools de 1 personaje distinto cada uno (figura 18).

La condición por la cual se rige la desaparición de un personaje y la aparición del siguiente es la distancia a la cámara que representa la ubicación del jugador actual, de manera que siempre se encuentren a un rango determinado del jugador y el jugador sea consciente de su presencia y pueda ver sus animaciones. A partir de dicho rango, los personajes volverán a ser inactivos y se procederá a

realizar la aparición de un nuevo personaje según el pool. La aparición, de manera similar a la desaparición, se realiza en un rango alrededor del jugador, obviamente menor al rango de desaparición y de manera aleatoria.

Además, en este momento de la aparición, se comprueba que la ubicación aleatoria objetivo esté dentro de la malla de navegación (navmesh)[46] por el que se mueve el agente del personaje. Esto se consigue gracias a la función 'SamplePosition' de la librería NavMesh, que busca el punto más cercano dentro del NavMesh a un punto proporcionado a la función dentro de un rango. Si lo encuentra, lo devuelve en forma de un 'NavMeshHit' que podrá ser usado para obtener la ubicación en la que debe colocarse el modelo del personaje en cuestión.

Al arrancar el juego, el script encargado de la aparición y la gestión del pooling detallado anteriormente generará un número de personajes concreto de los pools dependiente de una variable pública modificable de dicho script. Posteriormente en su función de actualización 'update' comprueba el número de personajes presentes y si uno ha desaparecido genera al siguiente de nuevo en el rango comentado anteriormente. Por otro lado, la responsabilidad de que el personaje desaparezca y vuelva al pool es del script que gestiona la IA del propio personaje, que monitorea en cada 'update' la distancia al jugador.

3.2.4. Corrección de errores

En este apartado se revisarán todas las correcciones de errores que se llevaron a cabo dentro de las labores correspondientes a la Optimización.

3.2.4.1. Errores relacionados con el pooling y aparición de personajes

Cuando comenzaron las labores de prueba del Pooling de los personajes, era común que aparecieran errores debido a la complejidad de la labor.

3.2.4.1.1. Spawn de los personajes en el origen de coordenadas.

El primer error detectado en los procesos de pruebas de este sistema fue la aparición de los personajes en el origen de coordenadas.

Esto fue debido a que el GameObject[47] en el que se encuentra el script de aparición, es hijo de otro objeto y estaba usando la distancia relativa como si fuera la global. La posición del GameObject con respecto a su padre era (0, 0, 0), por lo que el lugar de aparición lo detectaba como el origen de coordenadas.

Una vez detectado el motivo del error la solución fue sencilla cambiando a usar la posición global de dicho GameObject.

3.2.4.2. Distancia de clipping de las sombras

Un error visual del proyecto en tiempo de ejecución era una aparición extraña de las sombras a medida que el jugador se acercaba a cualquier árbol o casa, de manera que, por ejemplo, cuando el jugador se acercaba a un árbol, la sombra del mismo aparecía ante este, prácticamente a varios metros.

Este error estaba causado por la distancia de clipping de las sombras, que solo aparecían a cierta distancia de la cámara para ahorrar recursos, pero el efecto que causaba era confuso a pesar del ahorro.

Figura 19. Sombras de objetos tras la modificación de la distancia de clipping.
(Fuente: Elaboración propia)

Se procedió a modificar esta distancia para mejorar este comportamiento y tras realizar varias pruebas con varios valores, se determinó que la distancia de clipping óptima eran 100 unidades (figura 19).

3.2.4.3. Altura de personajes

La altura excesivamente alta de los personajes quedaba en evidencia cuando en tiempo de ejecución pasaban delante de puertas y otros assets con los que comparar su tamaño.

Las dos opciones para solucionar este problema pasaban por reducir el tamaño de dichos personajes o aumentar el tamaño de las casas y edificios. Se decidió

tomar la primera opción y reducir el tamaño de los personajes de cada uno de los escenarios realizando un scale del GameObject en el que se encuentran, ya que los modelos de las casas tienen las dimensiones correctas (figuras 20 y 21).

Figura 20. Altura de los personajes antes de la modificación de su escala.
(Fuente: Elaboración propia)

Figura 21. Altura de los personajes tras la modificación de su escala.
(Fuente: Elaboración propia)

3.2.4.4. Creación de Materiales para el terreno

Realizando las importaciones correspondientes para la optimización del terreno sucedía que la malla no contenía los materiales de la misma correctamente. La

malla debería contener los materiales que usa la misma en el momento de la exportación desde Blender, pero no funcionaba correctamente.

Se pidió al equipo del proyecto la exportación de los materiales a parte para crear en el motor Unity instancias permanentes de dichos materiales que luego poder relacionar con las mallas que se importen y con esto el problema quedó resuelto.

3.2.4.4.1. Displacement Maps para la creación de Materiales 3D

Al crear los distintos materiales a partir de los proporcionados por el equipo del proyecto, se observó que una de las texturas de algunos materiales como los de las tejas era un 'displacement map'[48]. Tras buscar la utilidad de estas texturas, se comprobó que son usadas de manera similar a un mapa de normales o un mapa de alturas para simular elevaciones en los materiales, pero en lugar de simularlas, convierten estas elevaciones simuladas en elevaciones reales, convirtiendo los materiales en una especie de materiales 3D.

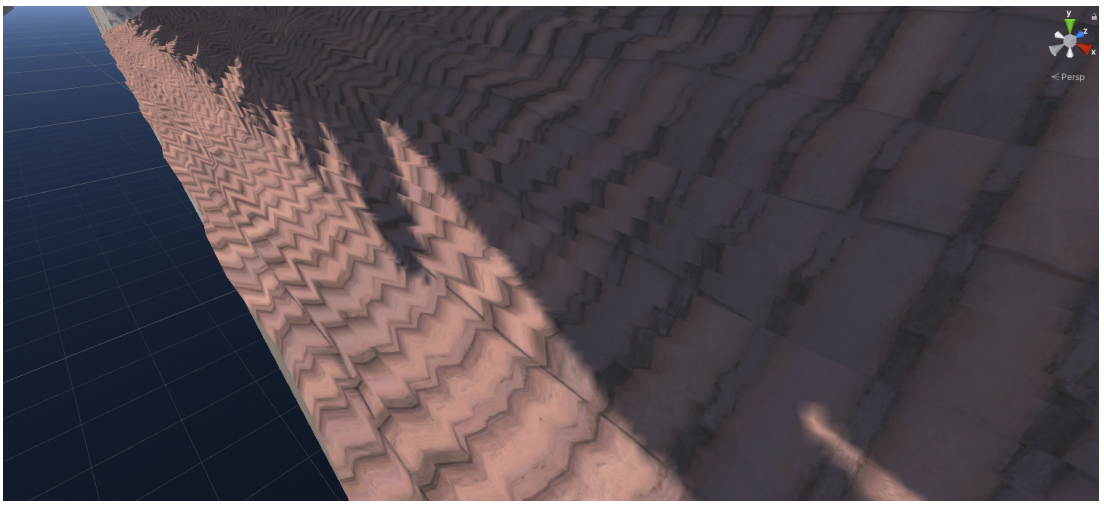

Figura 22. Error de renderizado de las texturas de las tejas debido al displacement map. (Fuente: Elaboración propia)

Se intentó sacar el máximo rendimiento de esta textura, por lo que se buscó el shader necesario en la store de Unity y se intentó implementar en dichos materiales, pero se detectaron muchos errores y bugs visuales.

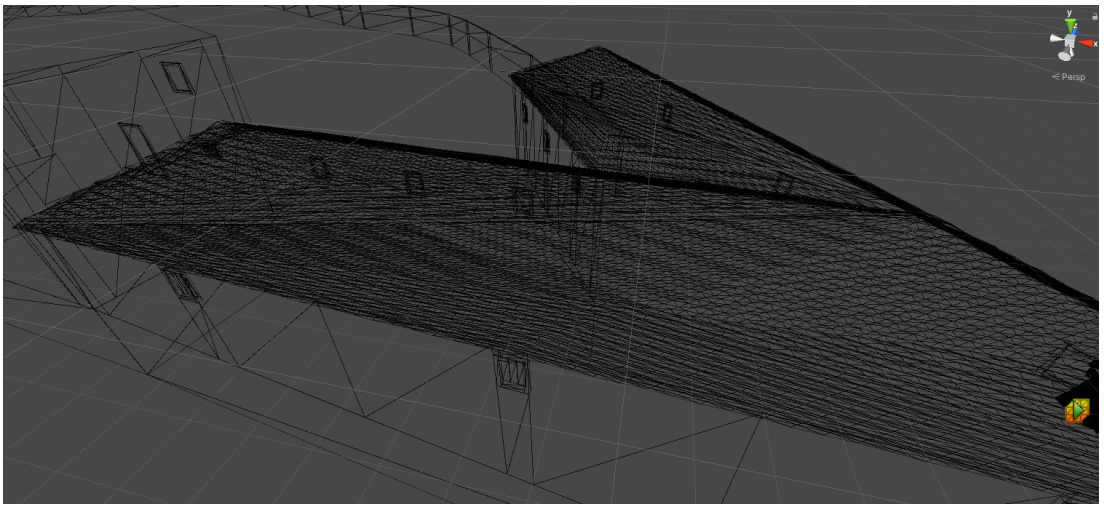

Figura 23. Densidad de triángulos en un tejado con el shader para Displacement maps aplicado.
(Fuente: Elaboración propia)

Además, al usar dicho shader la técnica de 'teselación'[49] para la generación de las alturas, el número de triángulos aumenta de manera indiscriminada, lo que choca de manera frontal con el planteamiento de este proyecto de optimización del rendimiento. Así que debido tanto al empeoramiento del rendimiento y el funcionamiento incorrecto, se decidió seguir utilizando estas texturas con el shader básico de Unity para mapas de altura.

3.3. Actualizaciones y mejoras de contenido

Finalmente, se han revisado sistemas que ya estaban creados en el proyecto, por decisión propia o por sugerencia de los profesores encargados, para intentar mejorar su funcionamiento o terminar de implementar todas sus funcionalidades.

3.3.1. Sistema de Guía y audios

Dentro del proyecto, uno de los pilares que proporcionará al jugador una experiencia de progreso y avance dentro del juego es el sistema de guía. Alguno de los personajes se diferenciará del resto actuando como guía y recorriendo ciertos puntos clave de la ciudad de La Laguna, reproduciendo audios explicando la historia y curiosidades de dichas ubicaciones.

En el punto en el que se recibió el proyecto, el guía necesita ser seleccionado por el jugador dentro del entorno VR para comenzar su ruta. Sigue los puntos de interés del escenario actual siguiendo un algoritmo que recorre los puntos

siempre yendo al siguiente más cercano reproduciendo el audio correspondiente al llegar a cada uno de ellos. Actualmente, los audios específicos no están desarrollados y se reproduce el mismo audio por defecto en todos los puntos, de manera que no se puede comprobar el correcto funcionamiento de todos los puntos de interés reproduciendo audios distintos.

3.3.1.1. Optimizar el algoritmo de recorrido de los puntos de interés

Como se ha comentado, el guía usa un algoritmo de recorrido al más cercano para recorrer los puntos de interés, de manera que desde un punto, siempre irá hacia el más cercano que no se haya recorrido ya. Esto puede no ser lo más óptimo en alguno de los escenarios por lo que se decide intentar cambiar este algoritmo y optimizar el recorrido que sigue el guía.

Lo que se necesitaba era un algoritmo que optimice el recorrido entre todos los puntos en un grafo conectado en el que existen mayor cantidad de aristas que de nodos, o sea, un algoritmo que realice un árbol de expansión mínimo (Minimum Spanning Tree)[50].

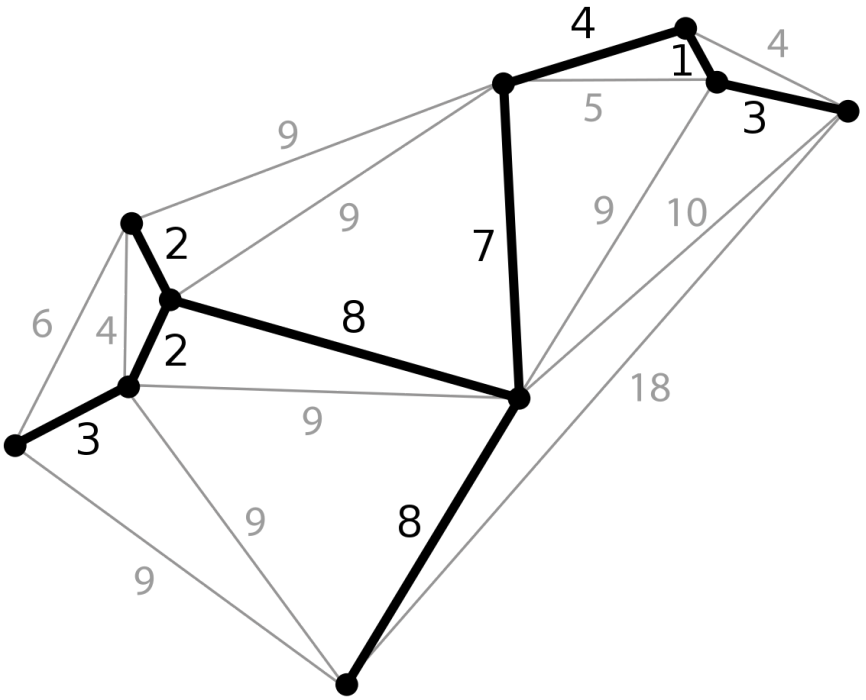

Figura 22. Ejemplo de MST en un grafo conectado..
(Fuente: https://en.wikipedia.org/wiki/Minimum_spanning_tree)

Los algoritmos más populares para realizar MSTs son el algoritmo de Prim y el algoritmo de Kruskal. Tras analizar ambos algoritmos para dilucidar cuál es más óptimo de usar, se decidió que la mejor opción es usar el algoritmo de Prim ya que funciona mejor que Kruskal para grafos conectados con mayor cantidad de aristas que de nodos.

Tras esto, se implementó el algoritmo, así como varias funciones de ayuda, que permiten, por ejemplo, generar una matriz de distancias del grafo a partir de los puntos de interés, para ser usado por el algoritmo o generar una ruta óptima a partir de los resultados del algoritmo.

3.3.1.2. Creación de audios concretos para objetivos del guía

Para sustituir los audios en inglés que reproduce el guía cada vez que llega a una parada, se hicieron audios personalizados para cada una de las paradas del guía del 'escenario 1'.

Cada uno de estos nuevos audios consta de la frase: "Este es el audio de (punto de interés correspondiente)", de manera que sea posible diferenciar que en cada parada el guía reproduce el audio correcto.

Las grabaciones fueron realizadas con el software Audacity y con mi propia voz. Tras esto, en el mismo software se realizaron modificaciones para aumentar el volumen y eliminar ruido de fondo de manera que los audios fueran lo más profesionales posibles.

Al comprobar el funcionamiento se corroboró que los audios se reproducen correctamente en cada parada en la que corresponden.

3.3.2. Corrección de errores

En este apartado se revisarán todas las correcciones de errores que se llevaron a cabo dentro de las labores correspondientes a la actualización y mejora de contenido.

3.3.2.1. Audios de comienzo y fin de la guía no se reproducen

Durante el proceso de testeo del correcto funcionamiento del guía se comprobó que los audios de inicio y fin no se reproducian. El guía pasaba directamente a moverse a los puntos de interés en lugar de reproducir el audio de inicio y al terminar, no reproducía el audio de fin.

En primer lugar, se identificaron ciertos errores en el código del guía que procedieron a corregirse. Aun así, los audios de inicio y fin seguían sin

reproducirse correctamente. Estos audios están incorporados como un componente AudioSource en el propio GameObject del guía, sin embargo, por algún motivo, no se reproducen correctamente de la misma manera que los componentes AudioSource de los hitos de la ruta.

Por ello se decidió añadir estos audios como hitos del guía para que pudiera reproducirlos. El audio de inicio se ubica en un hito justo en la localización inicial del guía. El audio de fin, en cambio, no es tan fácil, ya que dependiendo del escenario terminará en distintos sitios según el algoritmo de Prim, por ello se ubicará en un lugar cualquiera oculto del mapa, el script del guía lo localizará buscando su etiqueta personalizada, y reproducirá su audio al finalizar la ruta.

Conclusiones y Líneas futuras

Tras todo el trabajo realizado ha mejorado considerablemente el rendimiento del programa en el hardware objetivo. Se encontró un proyecto bien elaborado, pero que a duras penas podía ser ejecutado en las Meta Quest 2, con un entorno con gran número de triángulos que dificulta su renderización y sin un sistema de control de versiones. Después del desarrollo, el programa, aunque aún mejorable en tasa de frames, es jugable en las Meta Quest 2 y también mejoró la organización del proyecto.

Sin embargo, aún quedan ciertas mejoras que aplicar y trabajo que realizar, como por ejemplo la mejora de la jugabilidad del mismo, implementando algún sistema más que 'gamifique' de alguna manera las rutas y la exploración del entorno. Otra cosa que se queda en una especie de tintero personal, es realizar un intento de portabilidad del programa al motor Unreal Engine de Epic Games. La quinta y última versión de este motor implementa una gran cantidad de mejoras en el ámbito de la gestión óptima de grandes mapas y mundos abiertos, que podrían haber sido de gran ayuda para optimizar el rendimiento de este proyecto.

Summary and Conclusions

After all the work done, the performance of the program on the target platform has improved considerably. A well developed project was found in the beginning, but a project that could not run on Meta Quest 2 Headset, with an environment with lots of triangles making the renderization imposible and without a virtual control system to work appropriately with other colleagues. In the end, the project management is a lot better and the game can run on the target platform, although the framerate is yet improvable.

However, there are lots of improvements to apply and lots of work to be done in the game, like adding some other systems to improve playability, gamifying routes and exploration of the environment. Another thing that is left in a kind of personal way is trying to port the game to Unreal Engine. The fifth and last version of this game engine implements lots of improvements to manage large maps and open worlds that could have been very helpful to optimize the performance of the project at hand.

Bibliografía

[1] «Entrada de Wikipedia de Leonardo Torriani». https://es.wikipedia.org/wiki/Leonardo_Torriani (consultado 6 de junio de 2022)

[2] «San Cristóbal de La Laguna en la UNESCO». https://whc.unesco.org/es/list/929#top (consultado 6 de junio de 2022)

[3] «San Cristóbal de La Laguna en Ciudades Patrimonio de España». https://www.ciudadespatrimonio.org/ciudades/index.php?cd=9 (consultado 6 de junio de 2022)

[4] «Meta's Dominance in the VR Market will be Challenged in the Coming Years». https://www.idc.com/promo/arvr (consultado 6 de junio de 2022)

[5] Sega. https://www.sega.com (consultado 20 de junio de 2022)

[6] Nintendo. https://www.nintendo.com (consultado 20 de junio de 2022)

[7] Virtual Boy. https://es.wikipedia.org/wiki/Virtual_Boy (consultado 20 de junio de 2022)

[8] Oculus Rift. https://es.wikipedia.org/wiki/Oculus_Rift (consultado 20 de junio de 2022)

[9] Playstation VR. https://www.playstation.com/en-us/ps-vr/ (consultado 20 de junio de 2022)

[10] HTC Vive. https://www.vive.com/us/ (consultado 20 de junio de 2022)

[11] Beat Saber. https://beatsaber.com (consultado 20 de junio de 2022)

[12] Tetris Effect VR. https://www.oculus.com/experiences/quest/3386618894743567/ (consultado 20 de junio de 2022)

[13] «Arts and Culture VR/AR: Using Augmented & Virtual Reality to Increase Cultural Awareness». https://skywell.software/blog/augmented-virtual-reality-to-increase-cultural-awareness/ (consultado 20 de junio de 2022)

[14] Visitas virtuales del Museo Thyssen-Bornemisza. https://www.museothyssen.org/thyssenmultimedia/visitas-virtuales (consultado 20 de junio de 2022)

[15] Oculus Quest 2. https://store.facebook.com/es/es/quest/ (consultado 20 de junio de 2022)

[16] Unity3D. https://unity.com (consultado 31 de julio de 2022)

[17] Licencia PUE. https://www.pue.es/pue-academy/unity-certified-user/recursos-docentes (consultado 31 de julio de 2022)

[18] Vertex Animation Tools (VAT). https://assetstore.unity.com/packages/tools/animation/vertex-animation-tools-128190 (consultado 31 de julio de 2022)

[19] XR Plugin Management y Oculus XR Plugin. https://docs.unity3d.com/Manual/XRPluginArchitecture.html (consultado 31 de julio de 2022)

[20] Microsoft Visual Studio Code. https://code.visualstudio.com (consultado 31 de julio de 2022)

[21] Microsoft Visual Studio. https://visualstudio.microsoft.com/es/ (consultado 2 de agosto de 2022)

[22] .NET https://dotnet.microsoft.com/en-us/ (consultado 2 de agosto de 2022)

[23] C#. https://docs.microsoft.com/en-us/dotnet/csharp/ (consultado 2 de agosto de 2022)

[24] C. https://es.wikipedia.org/wiki/C_(lenguaje_de_programación) (consultado 2 de agosto de 2022)

[25] C++. https://cplusplus.com (consultado 2 de agosto de 2022)

[26] Java. https://www.java.com/es/ (consultado 2 de agosto de 2022)

[27] MonoBehaviour. https://docs.unity3d.com/ScriptReference/MonoBehaviour.html (consultado 2 de agosto de 2022)

[28] Android Studio. https://developer.android.com/studio (consultado 15 de agosto de 2022)

[29] Android. https://www.android.com (consultado 15 de agosto de 2022)

[30] IntelliJ JetBrains. https://www.jetbrains.com/idea/ (consultado 15 de agosto de 2022)

[31] Android SDK Manager. https://developer.android.com/studio/command-line/sdkmanager (consultado 15 de agosto de 2022)

[32] Helix Core. https://www.perforce.com/products/helix-core (consultado 15 de agosto de 2022)

[33] Helix Visual Client P4V. https://www.perforce.com/downloads/helix-visual-client-p4v (consultado 15 de agosto de 2022)

[34] Audacity. https://www.audacityteam.org (consultado 15 de agosto de 2022)

[35] MP3. https://es.wikipedia.org/wiki/MP3 (consultado 15 de agosto de 2022)

[36] Blender. https://www.blender.org (consultado 15 de agosto de 2022)

[37] Oculus Developer Hub. https://developer.oculus.com/documentation/unity/ts-odh/ (consultado 15 de agosto de 2022)

[38] Meta Quest 2. https://store.facebook.com/es/es/quest/ (consultado 25 de agosto de 2022)

[39] «Best Practices - Folder Structure». https://forum.unity.com/threads/best-practices-folder-structure.65381/ (consultado 25 de agosto de 2022)

[40] Plastic SCM. https://www.plasticscm.com (consultado 25 de agosto de 2022)

[41] Github. https://github.com (consultado 25 de agosto de 2022)

[42] «Which folders should I add to git». https://forum.unity.com/threads/which-folders-should-i-add-to-git.608149/ (consultado 4 de septiembre de 2022)

[43] APK. https://es.wikipedia.org/wiki/APK_(formato) (consultado 4 de septiembre de 2022)

[44] Occlusion Culling. https://docs.unity.cn/540/Documentation/Manual/OcclusionCulling.html (consultado 4 de septiembre de 2022)

[45] Object Pooling https://es.wikipedia.org/wiki/Object_pool_(patrón_de_diseño) (consultado 4 de septiembre de 2022)

[46] Navigation Mesh (NavMesh). https://en.wikipedia.org/wiki/Navigation_mesh (consultado 4 de septiembre de 2022)

[47] GameObject https://docs.unity3d.com/ScriptReference/GameObject.html (consultado 4 de septiembre de 2022)

[48] Displacement Mapping https://en.wikipedia.org/wiki/Displacement_mapping (consultado 4 de septiembre de 2022)

[49] Teselación. https://en.wikipedia.org/wiki/Tessellation_(computer_graphics) (consultado 4 de septiembre de 2022)

[50] Árbol de Expansión Mínimo o Minimum Spanning Tree (MST) https://en.wikipedia.org/wiki/Minimum_spanning_tree (consultado 4 de septiembre de 2022)

Apéndice A
Guía de Organización del proyecto

Guía de Organización

Apéndice B
Guía de Instalación y Uso de Perforce

Guía de Uso de Perforce