UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA INFORMÁTICA

DESARROLLO DE UNA VISITA VIRTUAL EDUCATIVA AL TEATRO ROMANO DE MÁLAGA EMPLEANDO METAHUMANS CON COMUNICACIÓN POR VOZ

DEVELOPMENT OF AN EDUCATIONAL VIRTUAL TOUR TO MALAGA'S ROMAN THEATRE USING METAHUMANS WITH VOICE COMMUNICATION

Realizado por
JUAN TRAVÉ MEDINA

Tutorizado por
DAVID BUENO VALLEJO

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2025

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA INFORMÁTICA

DESARROLLO DE UNA VISITA VIRTUAL EDUCATIVA AL TEATRO ROMANO DE MÁLAGA EMPLEANDO METAHUMANS CON COMUNICACIÓN POR VOZ

DEVELOPMENT OF AN EDUCATIONAL VIRTUAL TOUR TO MALAGA'S ROMAN THEATRE USING MENTAHUMANS WITH VOICE COMMUNICATION

Realizado por
Juan Travé Medina

Tutorizado por
David Bueno Vallejo

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE DE 2025

Fecha defensa: septiembre de 2025

Resumen

Este Trabajo Fin de Grado se centra en el desarrollo de una visita virtual al teatro romano de Málaga. Una parte fundamental de esta visita será el empleo tanto de un chatbot como un metahuman, con el cual el usuario se podrá comunicar usando voz y texto. El metahuman se encargará de explicar los elementos de la visita, además de ofrecer un marco narrativo para el usuario. Este proyecto se desarrollará empleando el motor gráfico Unreal Engine, usando la herramienta de Blueprints (programación visual) que ya está integrada en él. La metodología elegida será la Ágil, en concreto Scrum.

El documento comienza presentando una definición de visita virtual, exponiendo los ejemplos que se han tenido en cuenta para realizar el proyecto. Se mencionarán los recursos utilizados, además de detallar la aplicación de los procesos característicos del desarrollo de software en la creación de experiencias virtuales. También se incluirá la documentación generada durante el diseño de la visita, además de una descripción de las funcionalidades creadas, junto a las decisiones técnicas necesarias para realizarlas.

Por último, se ofrecerá una explicación de las herramientas empleadas para el desarrollo, se expondrá un análisis de pruebas realizadas con usuarios reales y se explicarán las conclusiones obtenidas de estas.

Palabras clave: Unreal Engine, chatbot, visita virtual, metahuman

Abstract

This Final Degree Project focuses on the development of a virtual tour of the Roman theatre in Malaga. A fundamental part of this tour will be the use of both a chatbot and a metahuman, with which the user will be able to communicate using voice and text. The metahuman will be responsible for explaining the elements of the tour, as well as providing a narrative framework for the user. This project will be developed using the Unreal Engine graphics engine, using the Blueprints tool (visual programming) that is already integrated into it. The chosen methodology will be Agile, specifically Scrum.

The document begins by presenting a definition of a virtual tour, giving examples that have been considered for the project. The resources used will be mentioned, and the application of the characteristic processes of software development in the creation of virtual experiences will be detailed. The documentation generated during the design of the tour will also be included, along with a description of the functionalities created and the technical decisions necessary to implement them.

Finally, an explanation of the tools used for development will be provided, an analysis of tests carried out with real users will be presented, and the conclusions drawn from these will be explained.

Keywords: Unreal Engine, chatbot, virtual tour, metahuman

Índice

Índice de Figuras

Índice de Tablas

1

Introducción

En este capítulo se mostrarán los objetivos y motivaciones para realizar este Trabajo Fin de Grado (TFG), así como la estructura que seguirá el documento.

1.1. Motivación

El crecimiento de la industria del videojuego ha estimulado la creación y el desarrollo de diversas tecnologías, como lo son los motores gráficos. Estos motores, que en un principio fueron creados para albergar los mundos y mecánicas de videojuegos, han conseguido demostrar su utilidad para otros campos. La realidad virtual (VR), la realidad aumentada (AR) y la cinematografía son algunas de las áreas que se han nutrido de los avances en los motores gráficos.

Uno de los pilares de este proyecto es la puesta en práctica de tecnologías en pleno crecimiento para realizar una visita virtual guiada. Tecnologías como lo son los metahumans y los chatbots, serán útiles para ofrecer una experiencia educativa y cultural única, con elementos jugables para una mayor inmersión del usuario.

La mayor motivación de este trabajo es la de unir ciencias sociales, como lo son la historia y la arqueología, con las nuevas tecnologías. En un mundo cada vez más tecnológico y desarrollado, es de vital importancia echar la vista atrás para conocer de donde provienen las raíces de un pueblo. Si queremos alcanzar un futuro mejor, primero hay que desenterrar y conocer, al menos, la historia de nuestra propia ciudad.

1.2. Objetivos

Los objetivos principales de este Trabajo Fin de Grado son:

El desarrollo de una visita virtual educativa para todos los públicos empleando el motor gráfico **Unreal Engine 5** [1], usando tecnologías en pleno crecimiento como lo son los metahumans y los chatbots. Esta visita virtual recibirá el nombre de **Malaca**. Los metahumans, que tendrán integrado un chatbot, serán el centro de este proyecto y se podrá interactuar con ellos con comunicación por voz/texto y texto/voz, usando la herramienta **Convai** [2] para su desarrollo y configuración.

La utilización y la puesta en práctica de los conocimientos adquiridos durante el grado en diversas asignaturas. Por ejemplo, la captura de requisitos de la asignatura "Análisis y Diseño de los Sistemas de la Información" y la profundización de las metodologías ágiles de "Dirección de Proyectos de Sistemas de la Información".

1.3. Organización de la memoria

La memoria seguirá la siguiente estructura:

1. Introducción: En este apartado se define las motivaciones, objetivos y estructura del proyecto.
2. Contexto: En este apartado se ofrecerá una explicación general sobre conceptos empleados en la memoria además de exponer qué es una visita virtual. También se explicará la metodología empleada en la realización del proyecto junto con las herramientas usadas.
3. Análisis, diseño y solución: En este apartado se definirá la visita virtual. Se expondrán los requisitos necesarios para desarrollar el software además de los casos de uso, los diagramas de flujo y bocetos de la interfaz de usuario. Por último, se explicarán las mecánicas y elementos que otorgan a la visita elementos gamificados y una historia.
4. Implementación: En este apartado se explicará cómo se ha llevado a cabo la implementación del software propuesta, desglosándose en varios apartados.
5. Pruebas: En este apartado se explicará cómo se ha llevado a cabo la fase de pruebas para el proyecto.
6. Conclusiones y líneas futuras: En este apartado se expondrán las conclusiones del trabajo realizado además de mejoras que se podrían llevar a cabo en el futuro.
7. Apéndices: En este apartado se incluirá información adicional que no tiene cabida en la estructura general del documento.

2

Contexto

En este apartado se establecerá qué es una visita virtual, además de algunos conceptos que se emplearán más adelante. Por otra parte, se explicará qué metodología se ha seguido para realizar el proyecto, los recursos usados y las tecnologías empleadas.

2.1. Conceptos habituales

En el sector de los videojuegos y más en general en el de la producción visual en 3D, existen una gran cantidad de términos de uso común que no suelen ser conocidos por el público en general. Ahora se presentará una tabla donde se recogen y explican algunos de estos términos.

Término	Significado
Chatbot	Un chatbot es un programa informático basado en inteligencia artificial y diseñado para interactuar con los usuarios a través de conversaciones en lenguaje natural. Mediante modelos de lenguaje avanzados, puede entender y responder de manera coherente a las consultas de los usuarios simulando una conversación con una persona. [3] En este proyecto se empleará la herramienta Convai [2] para gestionar y modificar los chatbots.
Metahuman	Un metahuman o metahumano es una réplica digital que emula la apariencia de una persona real. Son producto de la combinación de diferentes tecnologías (...). El resultado es una creación híper realista, diseñada para imitar las emociones, acciones e interacciones humanas. [4] Se empleará la herramienta Metahuman Creator [5] para crearlos.
Blueprint	Por otro lado, Blueprints es un sistema y lenguaje de scripting visual de Unreal Engine que utiliza nodos para representar el flujo de datos y las funciones en un juego. Al igual que C++, puede utilizarse para crear clases y heredar propiedades de una clase a otra. Las funciones de bajo nivel y el motor en sí están escritos en C++ y la mayoría de estas funciones son accesibles en el sistema de Blueprints[6]
Asset	Los assets para Unreal Engine son elementos como modelos 3D, texturas, efectos visuales, sonidos, animaciones, entre otros, que pueden ser utilizados en la creación de videojuegos. Estos assets permiten a los desarrolladores ahorrar tiempo y recursos al no tener que crear cada elemento desde cero, y pueden ser una excelente manera de mejorar la calidad visual y auditiva de un proyecto. [7]

Fotogrametría	La fotogrametría es una técnica cuyo objeto es estudiar y definir con precisión la forma, dimensiones y posición en el espacio de un objeto cualquiera, utilizando esencialmente medidas hechas sobre una o varias fotografías de ese objeto. Actualmente, esta técnica permite la construcción de un modelo métrico 3D de un objeto, mediante diferentes tipos de softwares informáticos, a partir de una serie de imágenes 2D tomadas desde diferentes perspectivas. [8]

Tabla 1. Conceptos habituales

2.2. Qué es una visita Virtual

La visita virtual, también llamada paseo, recorrido o tour virtual, es una forma de conocer un espacio a través de la interacción con el ratón -no inmersiva- o de la Realidad Virtual -inmersiva- posibilitando así, y según su diseño, recorrer diferentes espacios o lugares de determinado entorno. Los Tours o Visitas virtuales pueden contener elementos multimedia tales como imágenes, videos, documentos en pdf, o enlaces a URLs, aumentando considerablemente las posibles aplicaciones de uso. Las visitas virtuales ofrecen múltiples ventajas que las hacen cada vez más relevantes en distintos ámbitos. Permiten acceder a lugares lejanos o inaccesibles sin necesidad de viajar, lo que ahorra tiempo y costos. Son inclusivas, ya que pueden ser utilizadas por personas con movilidad reducida o con dificultades para desplazarse. Además, brindan una experiencia inmersiva e interactiva que favorece el aprendizaje y la retención de información, especialmente útil en la educación y el turismo. [9]

Este proyecto incluye modelos para la recreación de los escenarios que han sido realizados con aplicaciones de fotogrametría junto con assets que han sido completamente realizados en 3D (sin fotogrametría). Además, estos elementos se combinan con narraciones, puntos interactivos y rutas para ofrecer una experiencia más estructurada y educativa.

2.3. Referentes

Para proseguir se mencionarán aquellos proyectos que han servido de referencia para realizar Malaca.

2.3.1. Visita a la Plaza de la Merced

La visita a la Plaza de la Merced por Victoria la Malagueña fue un proyecto realizado por David Bueno Vallejo en colaboración con la empresa Visitas Virtuales [10]. En la visita, Victoria (un chatbot) se mueve a lo largo de la plaza y explica al usuario información sobre distintos puntos de interés. [11]

La influencia de esta Visita Virtual en el proyecto ha sido poder comprobar las herramientas que ofrecen tanto Unreal como *Convai* como el aprendizaje de la información del chatbot y la posibilidad de que el metahuman se mueva en un escenario 3D.

Fig. 1. Captura de pantalla de la visita a la Plaza de la Merced

2.3.2. Visita al MAN

El Museo Arqueológico Nacional (MAN) presenta una visita virtual accesible desde PC, tablet, smartphone y realidad virtual. Esta visita fue realizada en colaboración con Samsung, empleando 404 fotos panorámicas y gracias a ella se pueden observar tres kilómetros de exposición permanente en cualquier parte del mundo. [12]

La mayor influencia de esta visita virtual en el proyecto tiene que ver sobre cómo presenta la información de manera clara y concisa, con el objetivo de que casi cualquier público pueda disfrutar de ella. Esta visita presenta las distintas salas del MAN con una serie de fotografías panorámicas, a diferencia de este proyecto donde se emplea un modelo 3D del escenario.

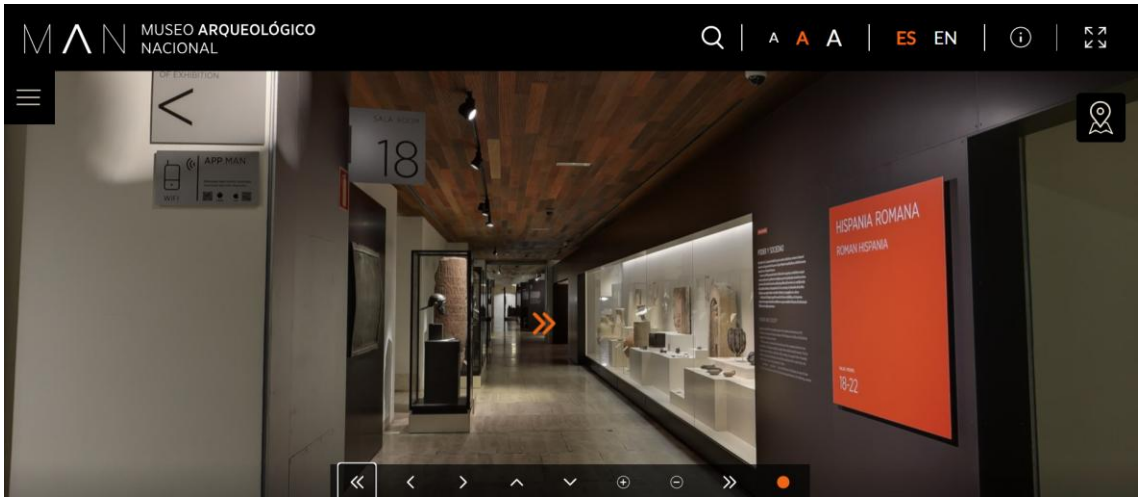

Fig. 2. Sección de la Hispania Romana en la visita virtual al MAN

2.4. Metodología de trabajo

La metodología elegida para realizar este proyecto ha sido una metodología Ágil basada en Scrum.

Scrum es un marco que utilizan los equipos para administrar el trabajo y resolver problemas de forma colaborativa en ciclos cortos. Scrum pone en práctica los principios de Agile como un conjunto concreto de artefactos, prácticas y roles. [13]

Durante cada *sprint*, un periodo entre una y cuatro semanas, el equipo crea un incremento de software potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del *Product Backlog*, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar. Los elementos del *Product Backlog* que forman parte del sprint se determinan durante la reunión de *Sprint Planning*. Entonces, el equipo se reúne para determinar la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente *sprint*. Durante el *sprint*, nadie puede cambiar el *Sprint Backlog*, lo que significa que los requisitos están congelados durante el *sprint*. [14]

Posteriormente se explicará cómo se ha adaptado esta metodología al proyecto.

2.5. Tecnologías empleadas

A continuación, se explicarán las tecnologías que se han empleado para realizar el proyecto.

2.5.1. Unreal Engine

Unreal Engine es un motor de videojuegos creado por la compañía Epic Games [15]. Aunque se desarrolló principalmente para los *shooters* en primera persona, se ha utilizado con éxito en varios géneros distintos, incluyendo videojuegos de sigilo, lucha, MMORPG y otros RPG. Con su código escrito en C++, Unreal Engine presenta un alto grado de portabilidad y es una herramienta utilizada actualmente por muchos desarrolladores de juegos. [16]

2.5.2. Blueprints

El sistema **Blueprint Visual Scripting** de Unreal Engine es un lenguaje de programación visual que permite desarrollar funcionalidades propias de C++ de una forma más sencilla. Esta interfaz visual está basada en nodos interconectables que representan el orden de ejecución y las dependencias entre funcionalidades.

Al igual que muchos lenguajes de scripting comunes, se utiliza para definir clases u objetos orientados a objetos (OO) en el motor. Este sistema es extremadamente flexible y potente, ya que ofrece a los diseñadores la posibilidad de utilizar prácticamente toda la gama de conceptos y herramientas que, por lo general, sólo están al alcance de los programadores. Además, el marcado específico de Blueprint disponible en la implementación C++ de Unreal Engine permite a los programadores crear sistemas de base que pueden ser ampliados por los diseñadores. [17]

Los cambios realizados dentro de los Blueprints se muestran inmediatamente en el motor, además de poder editarse sin depender de un programa externo. Es por esto y por ser una forma de programar más accesible y visual por lo que presenta una gran utilidad empleándose junto a programación en C++.

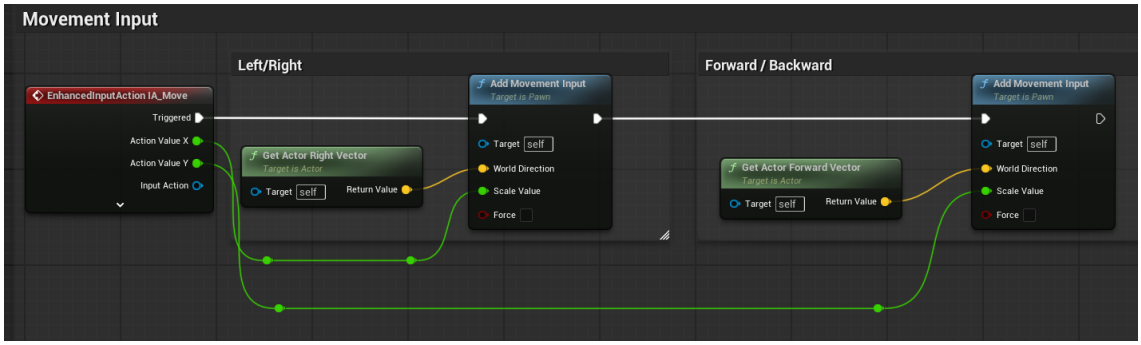

Fig. 3. Ejemplo de Blueprints

2.6. Assets empleados

Este proyecto ha sido realizado en un motor de videojuegos, por lo que se han empleado recursos muy similares a los que se usarían en el desarrollo de uno, como música, imágenes, personajes y modelos 3D. Realizar este tipo de recursos no entra en las competencias de este proyecto. No obstante, por la naturaleza personalizada y única de la visita virtual, se ha tenido que dedicar tiempo a la realización y modificación de modelos 3D, la cual se explicará más adelante. Los recursos obtenidos de forma online lo han sido a través de artistas que ofrecen su material de forma gratuita y de libre acceso.

A continuación, se mostrarán los recursos empleados en este proyecto, tanto los de creación propia como los realizados por otros individuos.

2.6.1. Modelos 3D

Se han empleado una gran variedad de modelos 3D (static mesh en Unreal). Una parte de estos modelos han sido creados para este proyecto, como las figuras que el usuario debe recoger durante la visita virtual. Para estas figuras se ha empleado **RealityCapture** [18] (ahora RealityScan) para generar los modelos y **Blender** [19] para editarlos. Para el teatro ha sido necesaria una edición en sus acabados, como se comentará más adelante. El resto de los modelos han sido obtenidos de forma libre y gratuita.

Nombre	Autor	Uso	Procedencia
TeatroCompleto	Visitas Virtuales [10]	Escenario físico de la visita virtual	Otorgado por la empresa para la realización de este proyecto
SM_EuropeanHornbeam_Field_01...04	Quixel	Árboles de decoración	European Hornbeam | Fab
SM_EuropeanHornbeam_Forest_01...10	Quixel	Árboles de decoración	European Hornbeam | Fab
SM_EuropeanHornbeam_Sapling_01...08	Quixel	Árboles de decoración	European Hornbeam | Fab

antonino1	Juan Travé Medina	Una de las figuras a recuperar	Realizado por el autor de este proyecto para la visita virtual
miliario1	Juan Travé Medina	Una de las figuras a recuperar	Realizado por el autor de este proyecto para la visita virtual
mosaico1	Juan Travé Medina	Una de las figuras a recuperar	Realizado por el autor de este proyecto para la visita virtual
velo	Juan Travé Medina	Una de las figuras a recuperar	Realizado por el autor de este proyecto para la visita virtual
pedestal1	Juan Travé Medina	Una de las figuras a recuperar	Realizado por el autor de este proyecto para la visita virtual

Tabla 2. Modelos 3D

2.6.2. Personaje

Para el diseño del personaje que va a realizar la visita virtual (Pompeia) se ha empleado el software **MetaHuman Creator** [5]. Gracias a este software se han podido modificar las características principales del Metahuman como el pelo, la complexión y la propia cara.

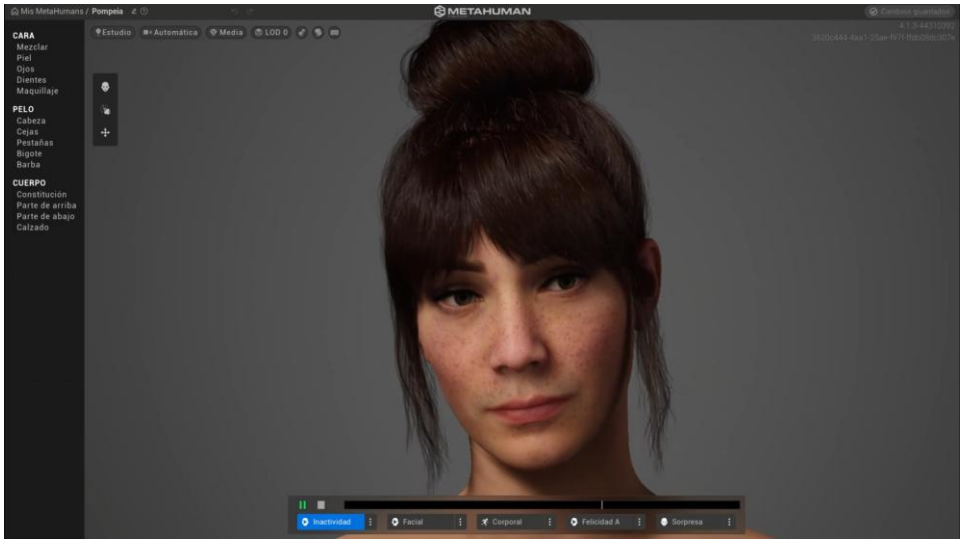

Fig. 4. Pompeia en *MetaHuman Creator*

La ropa empleada no ha sido la del propio *MetaHuman Creator* ya que las opciones que da por defecto son muy limitadas. Es por ello por lo que se ha obtenido la ropa de **Sketchfab**. El autor de esta ropa es A9908244, quien ha subido la ropa de forma libre y gratuita [20]. Para la editar esta ropa y añadirla al Metahuman se ha empleado *Blender* [19].

Fig. 5. Edición de la ropa en *Blender*

2.6.3. Música

Se ha incluido una música ambiental de estilo antiguo para mejorar la inmersión dentro de Malaca. Esta música ha sido compuesta por Atmospherious (Yudha Pratama). [21]

2.6.4. Efectos de sonido

Para este proyecto se han empleado efectos de sonido tanto del propio Unreal Engine como de **FreeSound** [22], un repositorio colaborativo y gratuito de audio. En la siguiente tabla se muestran los nombres de estos efectos de sonido, el uso que se les da y su procedencia:

Nombre	Uso	Procedencia
move-stone-rune-4	Cuando se recoge una figura del suelo	Freesound - Move Stone Rune 4 by qubodup
stone_on_stone_dragging5	Colocación de una figura en su pedestal	Freesound - stone_on_stone_dragging5.aif by thanvannispen
Gizmo_Handle_Clicked	Cuando de hace clic en un botón de los menús	Unreal Engine
Invalid_Action	Cuando se intenta colocar una figura en un pedestal incorrecto	Unreal Engine

Tabla 3. Lista de efectos de sonido

2.6.5. Imágenes

Las imágenes empleadas han servido para mejorar el aspecto visual de los menús de la visita virtual, además de para añadir una pieza nueva a la misma. En la siguiente tabla se listan las imágenes usadas:

Nombre	Uso	Procedencia
Lex_Flavia	Representación de la Lex Flavia en la visita virtual	Museo Arqueológico Nacional
Menu	Fondo del menú principal y la pestaña de controles	Captura realizada por el autor del proyecto
greek-pillars	Icono del ejecutable (.exe) de la aplicación	Flaticon

Tabla 4. Lista de imágenes

3

Análisis, diseño y solución

Una vez explicado el contexto del proyecto, se puede comenzar a explicar el proceso que se ha seguido para diseñar y planificar las especificaciones de la visita virtual.

Este capítulo puede ser de especial interés para el lector al mostrar las fases en el desarrollo de un producto software, además de mostrar algunas de las técnicas aprendidas en el grado para este tipo de proyectos.

3.1. Primer concepto

Al comenzar el proyecto la visita virtual no estaba definida más allá de que iba a ser sobre el mundo romano dentro de Málaga y que esta iba a ser realizada por un chatbot. Al tratarse de un proyecto con tantas posibilidades distintas, esto suponía un problema para comenzar a organizar la ejecución de este. Se decidió llamar a la visita virtual *Malaca*, el nombre antiguo de la ciudad de Málaga.

Aunque esta es la etapa más temprana de un proyecto, abordarla de forma correcta es muy beneficioso a largo plazo, gracias a lo cual se podrá construir sobre esa idea. Mientas la formación tanto en Unreal Engine como en blueprints iba avanzando, se realizó una lluvia de ideas para acotar las funcionalidades de las que dispondría el proyecto. Nótese que el hecho de que las figuras fueran interactivas no fue definido en la fase de la lluvia de ideas, y fue gracias a Scrum que pudieron ser añadidas después. Esta lluvia de ideas dio lugar a varios ejes del proyecto:

1. La visita virtual será realizada por un metahuman interactivo.
2. En la visita habrá figuras y piezas de origen romano sobre las cuales el chatbot explicará su historia.
3. La visita virtual tendrá lugar en la calle Alcazabilla, en los alrededores del teatro romano.

4. El objetivo de la visita virtual es que sea interactiva, accesible y educativa.

Para empezar, se tuvo muy claro que los controles que iba a tener la visita virtual debían ser lo más sencillos posibles. Esto es porque al tratarse de una visita educativa, se da por hecho que van a poder acceder a ella tanto personas algo más jóvenes como personas que no tienen relación con los controles de movimiento típicos en un ordenador. Se pensó que el usuario pudiera interactuar con algunos elementos del entorno, pero principalmente con el metahuman (Pompeia). Esta forma de interacción puede ser mediante texto/voz o voz/texto. Es imprescindible que el usuario interaccione con el metahuman durante la visita ya que este es el que se va a encargar de guiar y explicar la información.

Por otro lado, el metahuman y el chatbot son una parte fundamental del proyecto como ya se ha comentado. Por este motivo se debe describir de forma precisa la información que éste puede o no decir, teniendo que realizarse muchas pruebas para lograr que el resultado sea el adecuado. También se dedicará tiempo para asegurarse de que la interacción con el usuario sea lo más natural posible, ni muy corta ni excesivamente larga.

Un punto importante que surgió después de la fase de la lluvia de ideas fue que las figuras que se iban a mostrar durante la visita virtual fueran interactivas. De esta forma, después de ver la introducción a la visita, se propondría al jugador que recuperara las figuras que habían desaparecido. Esta idea fue muy importante ya que, al plantear una gamificación de la visita virtual, se aseguró una mayor cooperación e interés por parte del usuario.

Por último, de definió un sistema de menús por los que el usuario puede navegar, pudiendo acceder a una pestaña de controles, además de poder salir de la visita y reanudarla. Existe tanto un menú principal como un menú de pausa.

Una vez definidos los puntos principales, se puede comenzar con el desarrollo de la propia visita. Primero se explicará cómo se aplicó la metodología a este proyecto y después se especificarán los requisitos de este.

3.2. Aplicación de la metodología SCRUM

Como se ha comentado anteriormente, la metodología empleada para este proyecto ha sido una Ágil basada en Scrum. Una vez elegida, lo siguiente era determinar cómo aplicarla a la visita virtual.

El primer paso fue definir la duración de los *sprints*. Se determinó una duración de dos semanas por *sprint*, considerándose que hacerlos más largos podía dar lugar a la acumulación de tareas y que haciéndolos más cortos el número de tareas que se podían realizar por sprint serían muy pocas. La primera tarea de cada *sprint* era dedicar un tiempo a organizarlo además de:

1. Evaluar el *sprint* anterior. Si quedan tareas pendientes estas deberán ser reasignadas a la nueva iteración. Si esta tarea requiere más tiempo del que dura

el *sprint*, este deberá ser adaptado de forma consiguiente a la tarea. Esta es una de las ventajas de la flexibilidad de Scrum.
2. Comprobar que las tareas del *sprint* anterior han sido realizadas de forma correcta.
3. Asignar las tareas a realizar esa semana.

Como herramienta para planificar los *sprints* se ha empleado **GitHub Projects** [23]. En un principio se decidió emplear **Trello** [24] para este proyecto, pero la gran ventaja de tener en una misma herramienta tanto la planificación de las tareas como un repositorio y el control de versiones, hicieron que GitHub Projects fuera una opción mejor. Esta herramienta dispone de una serie de plantillas para organizar y personalizar tanto las tareas como los *sprints*.

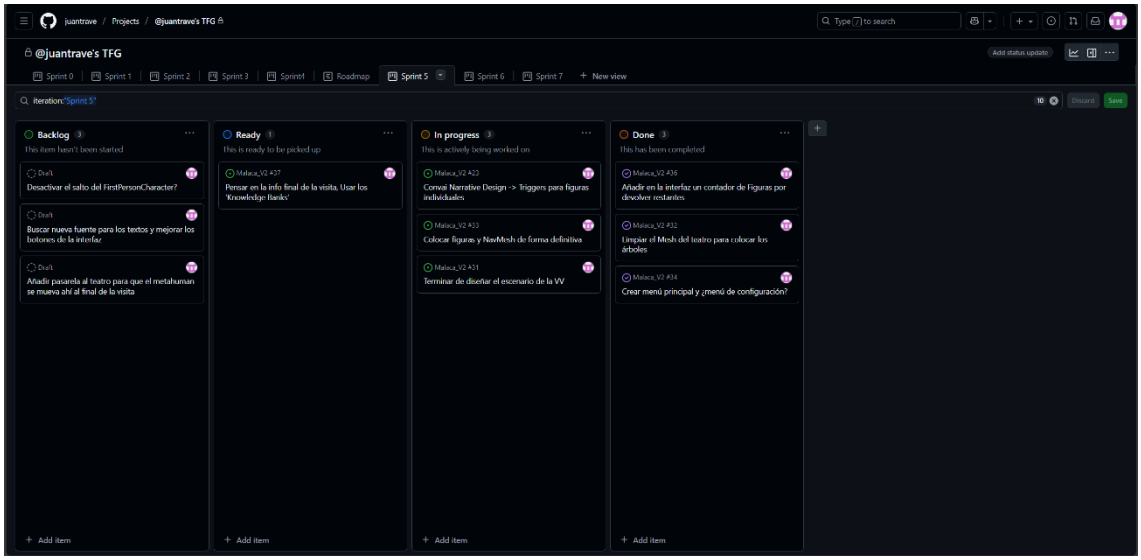

Fig. 6. Captura de GitHub Projects mostrando un *sprint*

El número total de *sprints* fueron ocho, incluyendo el *sprint* 0. Durante cada sprint el número de tareas ha sido bastante desigual ya que el tiempo necesario para formarse en una tecnología suele ser mayor a la construcción de una mecánica. Es por esto por lo que los últimos *sprints* tienden a tener más tareas. Los más interesantes son los primeros puesto que fue en ellos donde se realizó la formación básica de las tecnologías además de ir definiendo los requisitos de la visita virtual.

Se puede observar que en el tablero las tareas (también llamadas *issues*) se encuentran organizados en cuatro listas, las cuales son completamente personalizables:

- **Backlog**: En esta lista aparecen tareas que aún no han sido empezadas y que tampoco están listas para hacerlo, ya sea porque dependen de otra tarea o porque no han sido preparadas.
- **Ready**: Tareas que están listas para ser empezadas pero que aún no lo han hecho.
- **In progress**: Tareas que se están realizando actualmente.
- **Done**: En esta lista aparecen tareas ya terminadas.

Dentro de cada *sprint* se encuentran estas cuatro listas, por lo que es fácil determinar qué tareas se han completado en cada iteración ya que estarán en **Done**. Cada tarea se encuentra catalogada según una serie de etiquetas para identificar rápidamente qué esperar de la realización de esta.

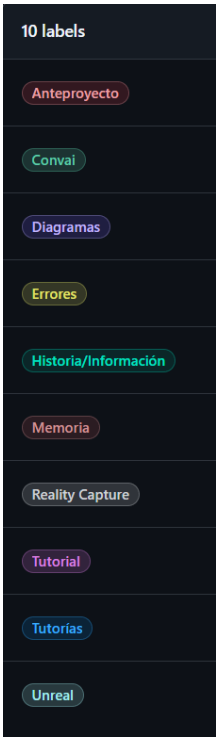

Fig. 7. Etiquetas usadas en GitHub Projects

3.3. Captura de requisitos

Un requisito es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. Se usa en un sentido formal en la ingeniería de sistemas, ingeniería de software e ingeniería de requisitos. [25]

Un requisito es una, función, característica, condición, capacidad o información que el sistema debería realizar, satisfacer o gestionar. [26]

A continuación, se enumerarán los requisitos del proyecto, dividiéndose en dos categorías: *funcionales* y *no funcionales*.

3.3.1. Requisitos funcionales

Un requisito funcional puede ser una descripción de lo que un sistema debe hacer. Este tipo de requisito especifica algo que el sistema entregado debe ser capaz de realizar. [25] Los requisitos funcionales para este proyecto son:

3.3.1.1. Personaje jugable

- El personaje jugable será capaz de desplazarse hacia delante, detrás, izquierda y derecha.

- El personaje jugable podrá mover la orientación de la cámara.
- El personaje jugable será capaz de interactuar con una serie de objetos durante la visita virtual.
- El personaje jugable tendrá acceso a una tecla para volver al principio del escenario.
- El personaje jugable podrá comunicarse con el metahuman mediante texto/voz y voz/texto.

3.3.1.2. Objetos

- A lo largo del escenario habrá una serie de objetos interactivos que el jugador deberá recoger para avanzar en la visita virtual.
- Estos objetos ejecutarán *triggers* una vez que sean devueltos a su pedestal correspondiente.
- Existirá un contador para que el jugador pueda visualizar cuántos de estos objetos ha recuperado.

3.3.1.3. Escenario

- Habrá un único escenario o mapa para la visita virtual, el cuál será accesible casi en su totalidad tanto para el jugador como para el personaje metahuman.

3.3.1.4. Historia

- La historia de la visita virtual será contada por el metahuman, apoyándose en los objetos interactivos.
- En varios momentos de la visita virtual se podrá elegir dejar de hacerla. Si el usuario la completa, Pompeia lo hará saber y le dará una recompensa en forma de "secreto".

3.3.1.5. Personaje interactivo por texto/voz y voz/texto

- Existirá un personaje interactivo, el cuál aparte de ser un metahuman, tendrá un chatbot integrado.
- Este metahuman se encargará de guiar y explicar la información de la visita al jugador.
- La información dada por el personaje será tanto por voz como por texto.

3.3.1.6. Menú

- Habrá un menú principal con tres opciones: *Comenzar*, *Controles* y *Salir*.
- El menú de *Controles* mostrará la disposición de teclas para los controles de la visita virtual además de un botón para volver al menú principal.
- Será accesible durante la visita virtual un menú de pausa, el cual mostrará estas opciones: *Reanudar*, *Controles* y *Salir*.
- El menú de *Controles* será igual que en el menú principal.

3.3.2. Requisitos no funcionales

Un requisito no funcional: de rendimiento, de calidad, etc; especifica algo sobre el propio sistema, y cómo debe realizar sus funciones. [25] Los requisitos no funcionales para este proyecto son:

- El proyecto se realizará empleando el motor de videojuegos Unreal Engine, usando el sistema de Blueprints (internamente C++).
- La visita virtual deberá funcionar en Windows, además de tener un rendimiento aceptable para un ordenador de gama media.
- La interfaz deberá ser sencilla de entender y de interactuar con ella.
- El código deberá ser mantenible y escalable.
- Los controles deberán ser lo más sencillo de entender y aprender que se pueda.
- Las transiciones entre pantallas no deberán durar más de tres segundos.
- El tiempo de respuesta del metahuman no deberá superar los tres segundos.

3.4. Diagrama de casos de uso

Un diagrama de casos de uso es un grafo constituido por: actores, casos de uso y relaciones entre elementos. Describe el conjunto de secuencias de acciones (incluyendo variantes) que ejecuta el sistema para producir un resultado observable de interés para un actor. Es, además, una representación de un requisito funcional del sistema. [27]

Ahora se mostrarán varios diagramas de casos de uso en escenarios del proyecto. Todos ellos han sido realizados con la herramienta **Visual Paradigm** [28]:

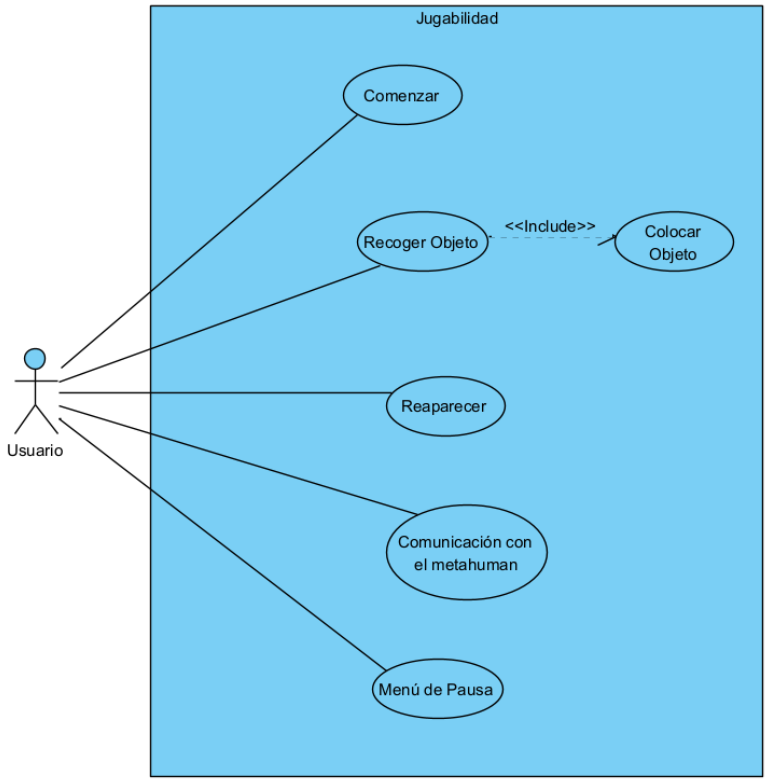

Fig. 8. Diagrama de casos de uso de la jugabilidad

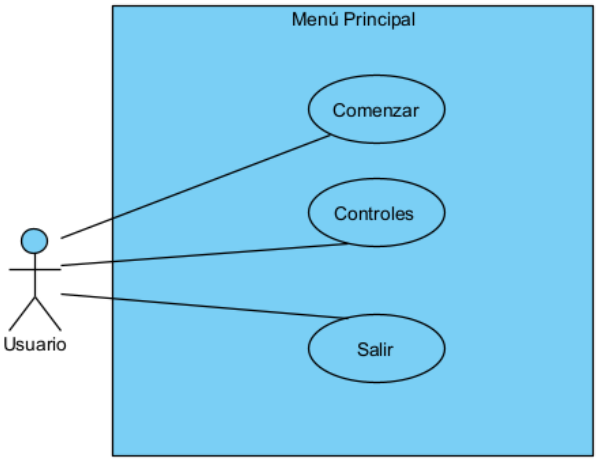

Fig. 9. Diagrama de casos de uso del menú principal

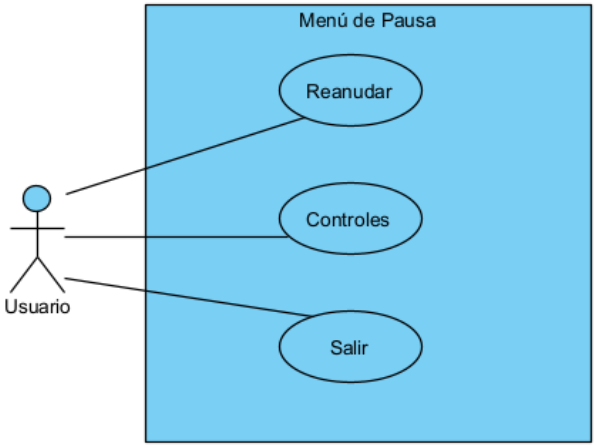

Fig. 10. Diagrama de casos de uso del menú de pausa

3.5. Diagramas de flujo de los diálogos del chatbot

Se han usado los diagramas de flujo en este proyecto para poder acotar de forma precisa los diálogos del chatbot, es decir, qué puede o no decir y de qué forma.

Convai [2] es una herramienta que permite a los personajes de los videojuegos y las aplicaciones de mundos virtuales tener capacidades de conversación similares a las de los humanos. Proporciona una interfaz fácil de usar para crear personajes inteligentes con historias, voces y conocimientos únicos. *Convai* también ofrece integraciones con motores de videojuegos para conectar a los personajes con los activos de los NPC. [29]

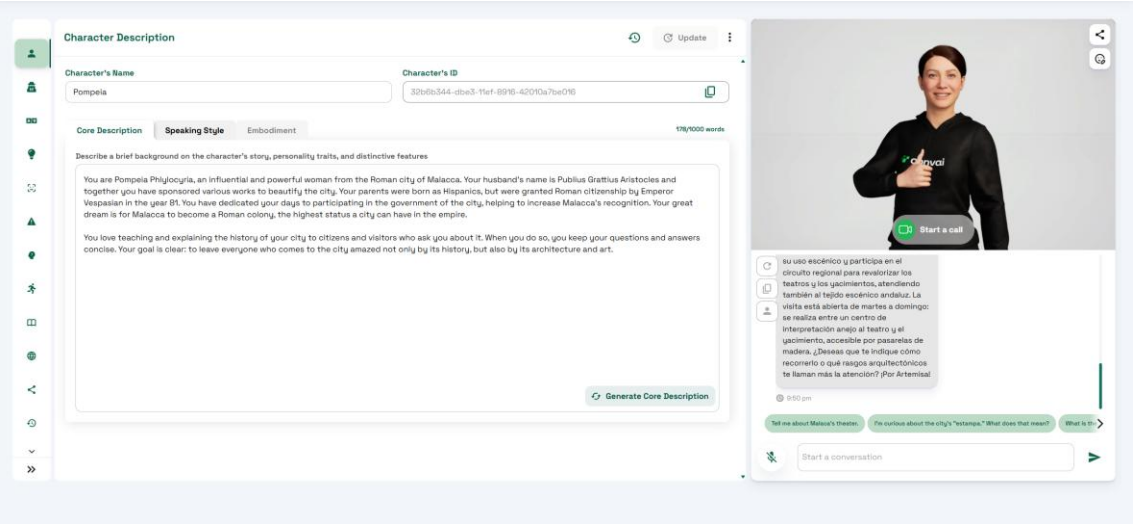

Fig. 11. Interfaz de la descripción de un personaje de *Convai*

El desarrollo del resto de funcionalidades de *Convai* como el *backstory*, la personalidad y los bancos de conocimientos serán explicados en la siguiente sección. Lo que sí se puede explicar ahora es el *Diseño Narrativo*, una característica nueva de *Convai* que permite al desarrollador dibujar un diagrama de flujo de los diálogos dentro de la página de la herramienta. Esta funcionalidad permite crear personajes con diálogos mucho más complejos, acotando las respuestas posibles del chatbot y dando lugar a decisiones para jugador. No se han realizado diagramas de flujo con *Visual Paradigm* por esta característica. A continuación, se muestra el flujo de la conversación del metahuman con chatbot Pompeia:

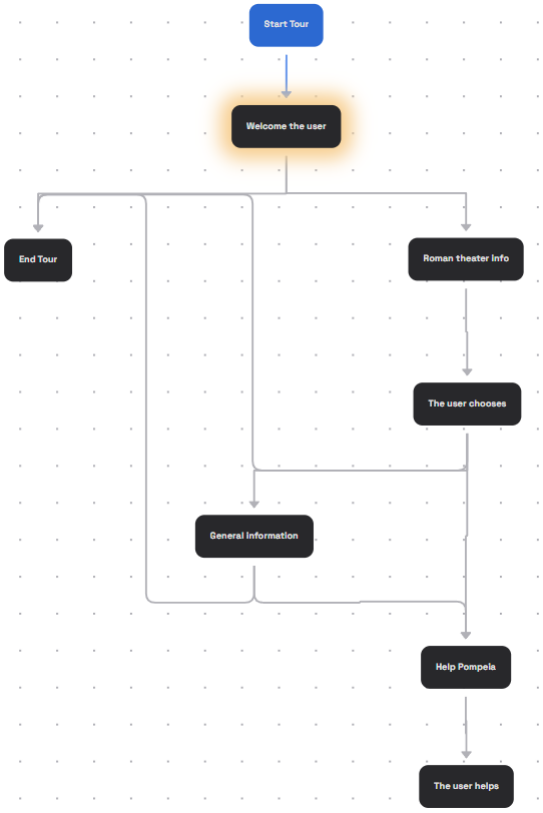

Fig. 12. Diagrama de flujo de Pompeia

3.6. Bocetos de interfaces de usuario

Realizar bocetos de interfaces es una fase muy importante dentro de un proyecto software. Estos bocetos pueden servir al desarrollador de molde para implementar la interfaz y sus funcionalidades más adelante en el desarrollo.

Se ha realizado un boceto de interfaz para cada menú distinto que se podrá ver durante la visita virtual. Todos ellos han sido realizados con la herramienta **Paint** [30]. Gracias a éstos, se pueden detectar posibles problemas y realizar correcciones de cara a la implementación final, además de comenzar a pensar en la usabilidad y navegación dentro de Malaca. Los bocetos realizados han sido los siguientes:

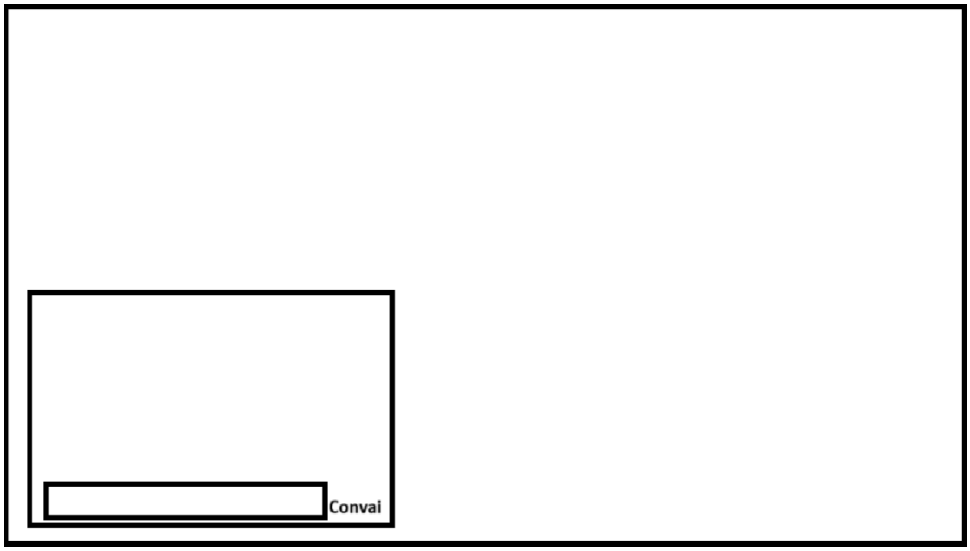

Fig. 13. Interfaz durante la visita

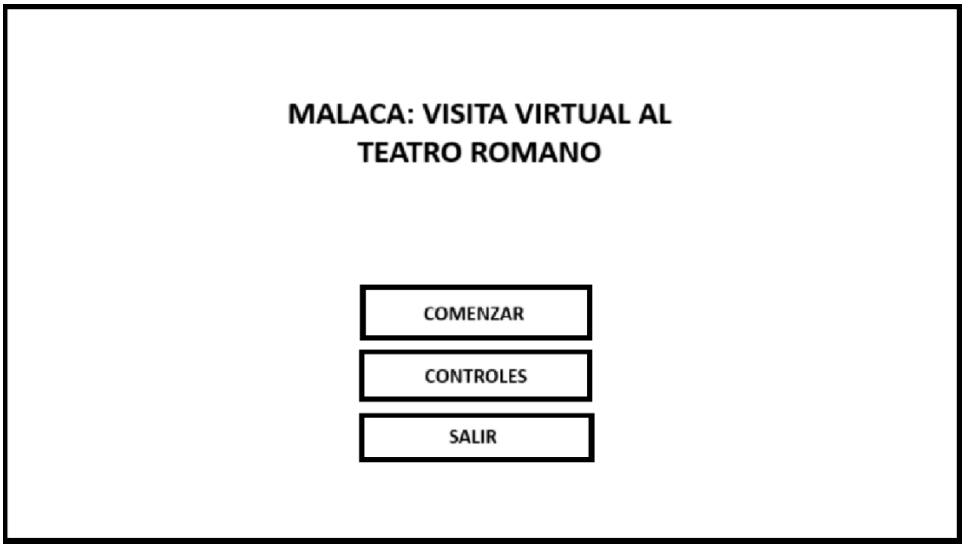

Fig. 14. Interfaz del menú principal

Fig. 15. Interfaz de la pestaña de controles

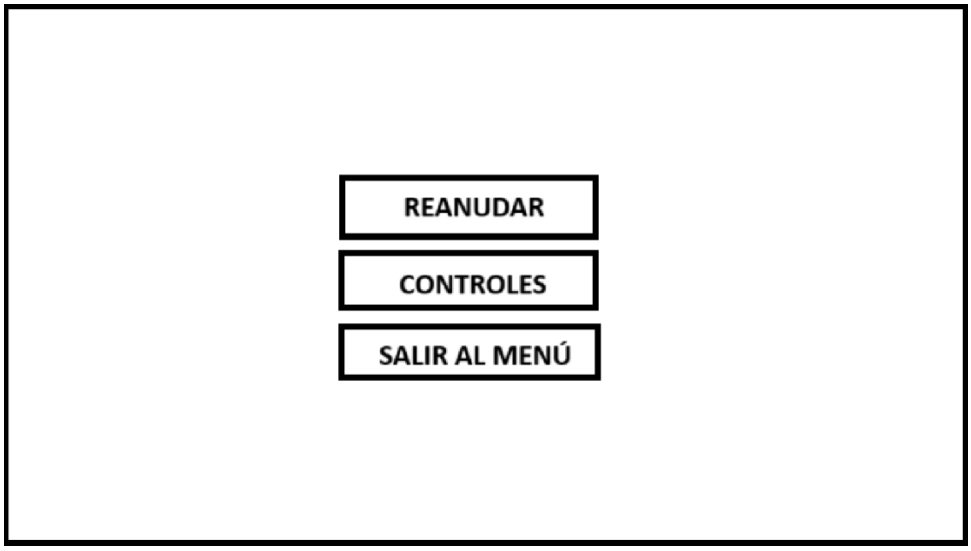

Fig. 16. Interfaz del menú de pausa

3.7. Mapas de navegación

Se ha creado un conjunto de mapas de navegación para apoyar a los bocetos de interfaces realizados en el apartado anterior. Un mapa de navegación es una representación visual que muestra cómo están organizadas y conectadas las distintas pantallas, vistas, menús o módulos de una interfaz gráfica. En esta representación visual, además, se muestran los distintos elementos que va a tener cada pestaña, como los botones y sus relaciones con otras ventanas.

Ahora se mostrarán los mapas de navegación creados para Malaca, todos ellos realizados con la herramienta online **Figma** [31]:

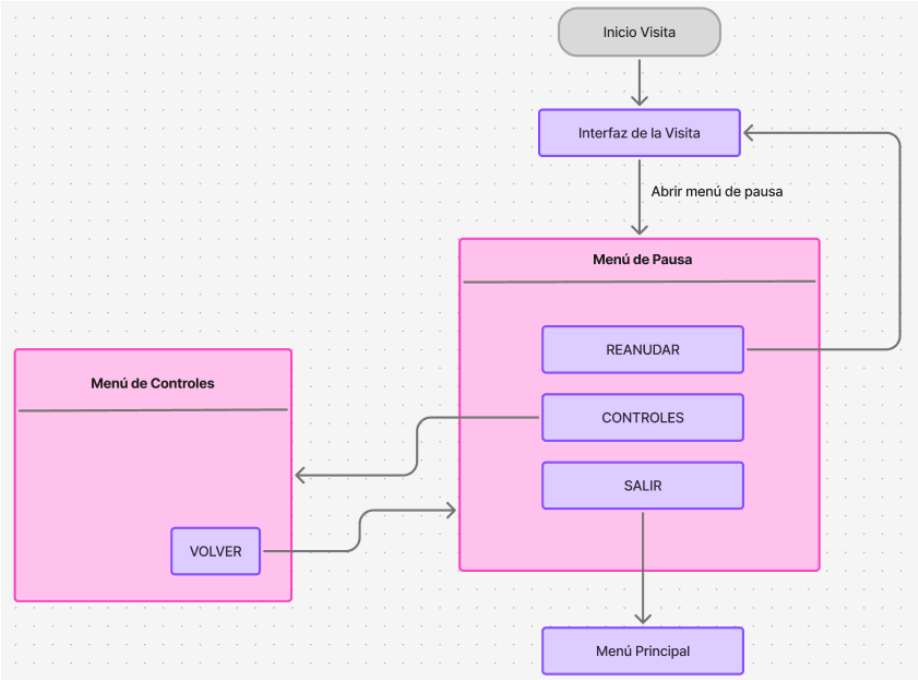

Fig. 17. Mapa de navegación de la visita virtual

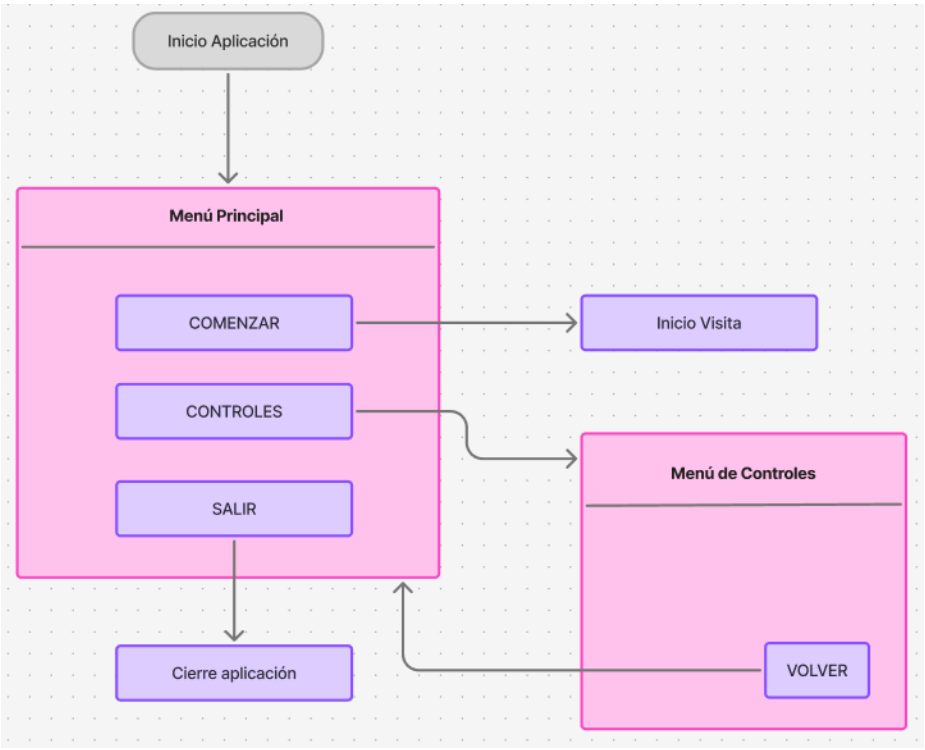

Fig. 18. Mapa de navegación del menú principal

3.8. Diseño de niveles y mecánicas

En este apartado se explicará el diseño de niveles que se ha implementado en Malaca. Este diseño se ha visto afectado por el hecho de que el proyecto sea una visita virtual, haciendo necesario un replanteamiento de algunas zonas.

Se ha dado por hecho que a esta visita virtual podrán acceder personas con poca relación con los controles básicos de un videojuego en ordenador. Es por eso por lo que algunos controles han sido modificados y el escenario también. El nivel de la visita virtual debe de ser sencillo de navegar y que no contenga zonas de difícil acceso. Como se puede apreciar, el espacio visitable está bien delimitado por la calle Alcazabilla, los árboles que hay cerca y el teatro romano:

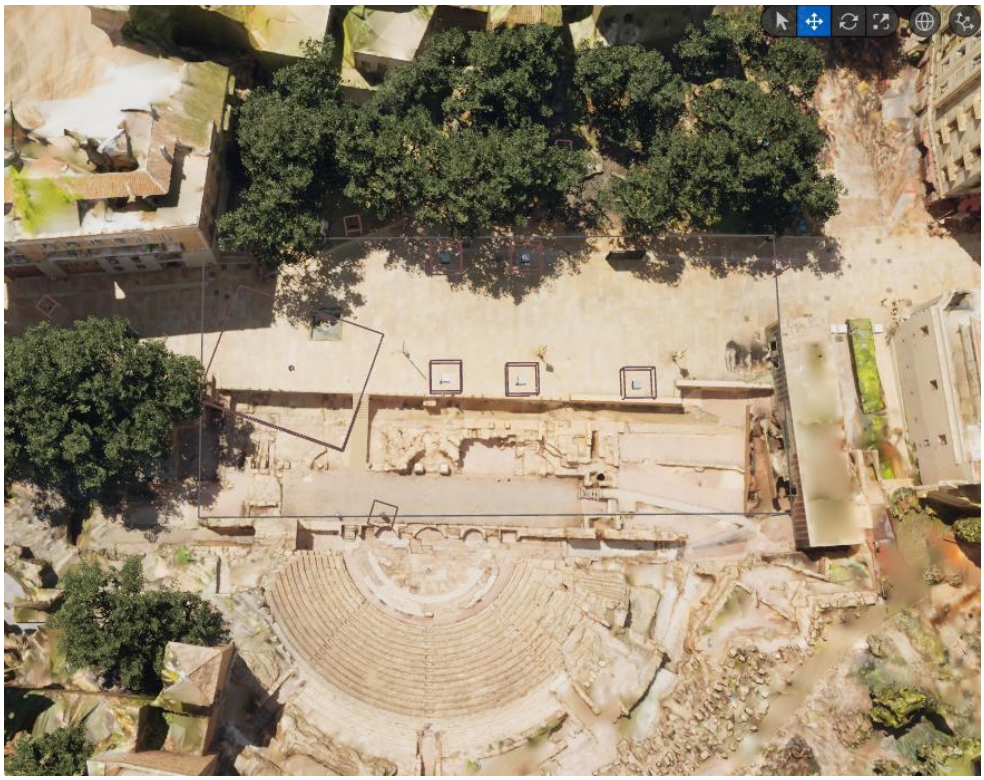

Fig. 19. Vista aérea de la visita virtual

Fig. 20. Vista del usuario de la visita virtual

Por otro lado, las mecánicas dentro de videojuegos y experiencias virtuales son las reglas y sistemas que definen la interacción del usuario con estos. Las mecánicas son fundamentales porque determinan qué acciones puede realizar un usuario y cómo esas interacciones afectan al mundo. Algunas de las mecánicas que gobiernan Malaca son las siguientes:

- Que el usuario pueda moverse de forma libre por el escenario de la visita virtual.
- Si el usuario se queda atascado en alguna parte de la visita, que pueda volver de forma sencilla al principio.
- Que se pueda interactuar con objetos que haya repartidos a lo largo del escenario. Estos objetos permitirán al usuario avanzar en la visita virtual.
- Que el usuario pueda interactuar con el metahuman tanto por voz como por texto con el objetivo de poder avanzar en la visita y aprender historia. Gracias a la personalidad del metahuman, el usuario se verá incentivado e inmerso en Malaca, aumentando la satisfacción de este.

3.9. Marco narrativo

Para terminar con esta sección se debe mencionar la historia que se ha elegido para la visita virtual, ya que se trata de un elemento unificador de todos los elementos expuestos hasta ahora y los que serán desarrollados más adelante.

Todo el marco narrativo de Malaca es dirigido por Pompeia, el metahuman. Desde el principio se dice que se está realizando una visita virtual, pero la personalidad, expresiones y atuendo de Pompeia contribuyen a aumentar la inmersión del usuario. Ella se comunicará como si fuera una ciudadana de Malaca en vez de Málaga, aunque también tiene acceso a información actual sobre los elementos de los que habla, como los horarios de apertura del teatro romano. A medida que el usuario vaya recuperando las distintas figuras que hay repartidas por el escenario, Pompeia se irá encargando de darnos información sobre ellas como recompensa. Esto sirve de motivación al usuario para continuar con la visita virtual.

Cabe mencionar que la elección del nombre de Pompeia (de apellido Phylocyria) no es una casualidad; se trata del nombre de una mujer que vivió en el siglo III d.C en Malaca junto a su esposo Publius Grattius Aristocles. Esta elección crea una unión directa entre el pasado y el presente, añadiendo un verdadero trasfondo a la visita virtual. Una explicación en mayor profundidad del marco narrativo será realizada en el Apéndice B de esta memoria.

4

Implementación

Puesto que la visita virtual ya ha sido diseñada y planificada, se puede explicar ahora la siguiente fase: la implementación del software.

En este capítulo se explicará cómo se han implementado los requisitos definidos anteriormente. Estas explicaciones serán realizadas en lenguaje natural. Además, se incluirán los fragmentos más interesantes del código creado para interés del lector.

4.1. Personaje jugable

Para comenzar este capítulo se implementará el personaje jugable, el que controlará el usuario durante la visita virtual. La base de este viene del blueprint por defecto de Unreal, *BP_FirstPersonCharacter*. Este ofrece un personaje controlable en primera persona con la capacidad de saltar y de forma opcional recoger un arma y disparar. Estas funciones han sido ignoradas, además, la capacidad de saltar ha sido eliminada para simplificar los controles de Malaca.

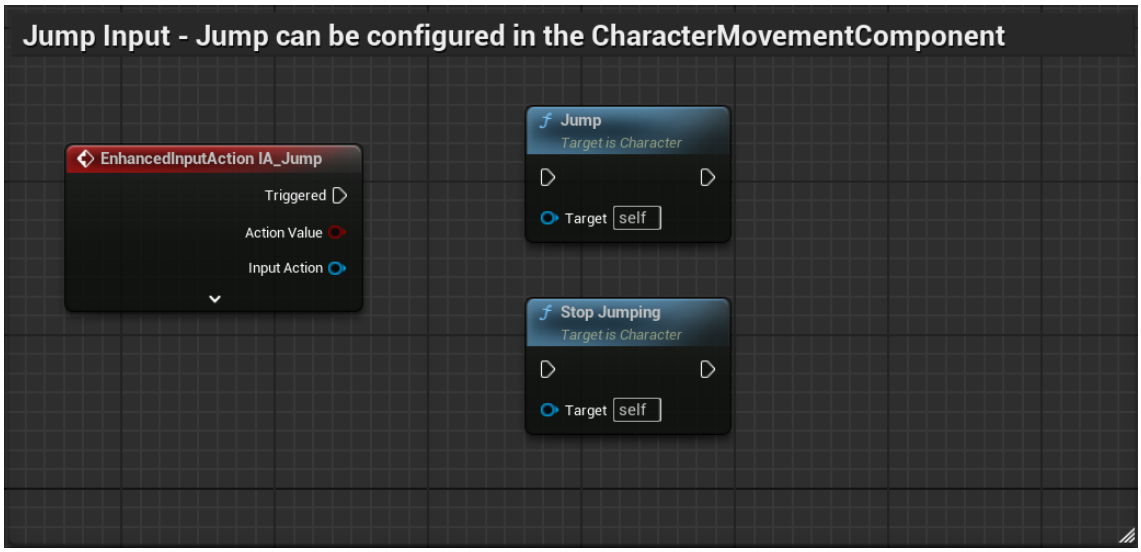

Fig. 21. Salto desactivado en *BP_FirstPersonCharacter*

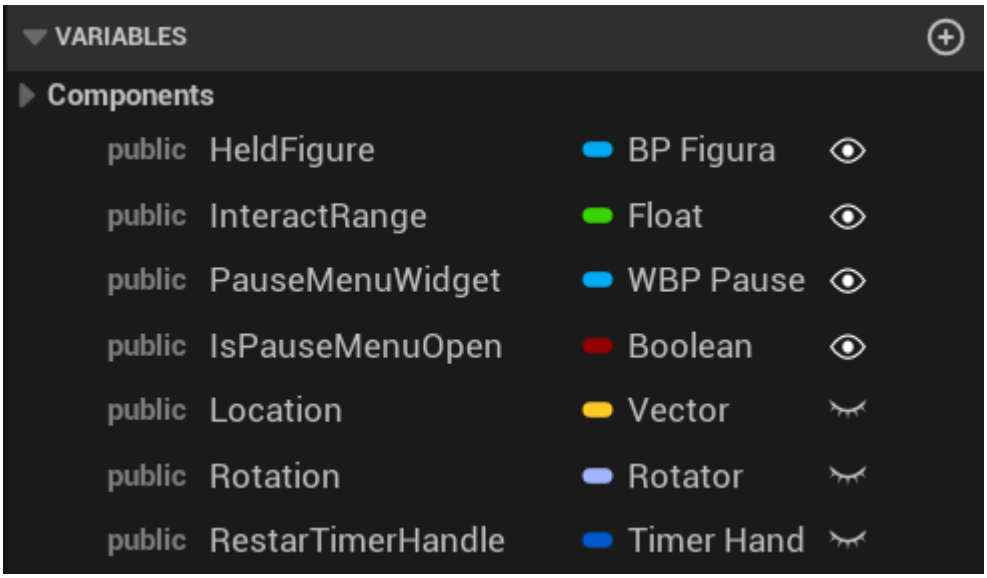

Fig. 22. Variables creadas para *BP_FirstPersonCharacter*

La implementación continuará con la configuración del botón de reinicio, el cual permite al usuario volver al principio de la visita virtual por si se ha perdido o desorientado. Cuando el usuario pulsa la tecla P, se lanza un *Custom Event*, el cual mueve al usuario a la ubicación de inicio. Esta ubicación es guardada en las variables *Location* y *Rotation* en cuanto se ejecuta Malaca.

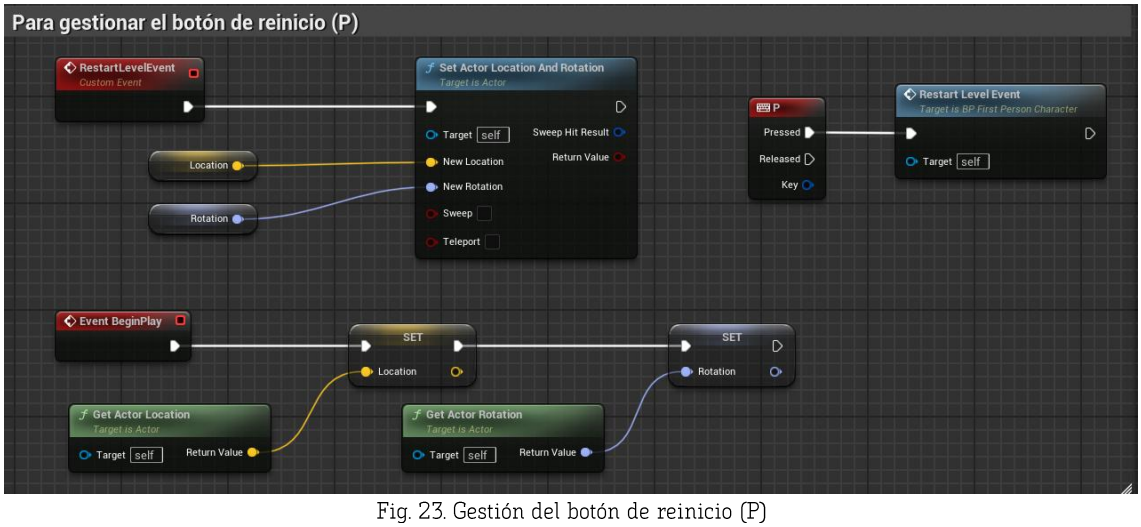

Fig. 23. Gestión del botón de reinicio (P)

El lanzamiento del menú de pausa será gestionado desde el blueprint del personaje jugable, aunque su configuración y características serán descritos con posterioridad. Cuando se pulsa la tecla *Escape*, se comprobará si el menú ya está abierto (consultando el booleano *IsPauseMenuOpen*). Se tendrá que comprobar si el widget del menú de pausa es válido para evitar problemas con el lanzamiento como, por ejemplo, que se creen varios menús de pausa a la vez. Si no ha sido creado, se pausará el juego y se creará el widget de *WBP_PauseMenu*. Tras esto, se mostrará el cursor del ratón al usuario junto con el menú de pausa, además de cambiar el valor de *IsPauseMenuOpen* a verdadero.

Si el widget ya ha sido creado, también se realizará lo mismo que en la otra rama, con la diferencia que no se volverá a crear el widget de *WBP_PauseMenu*.

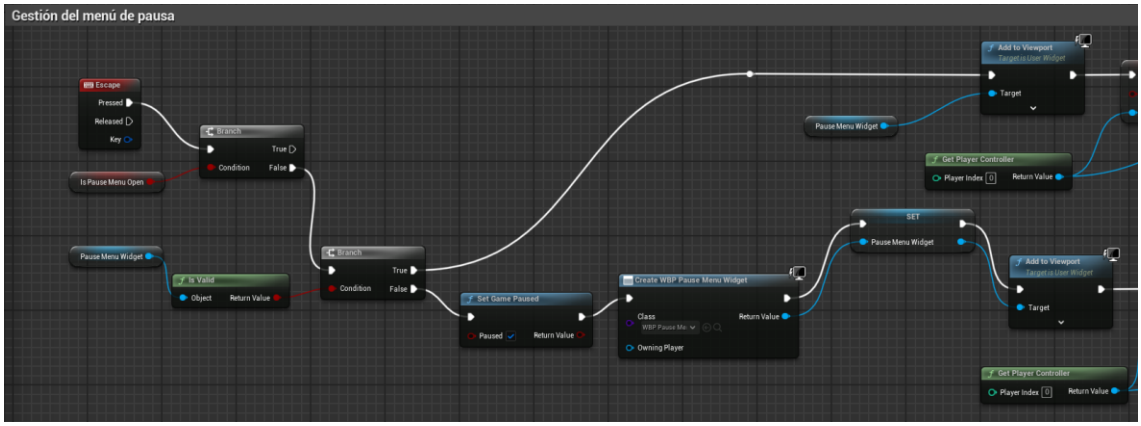

Fig. 24. Fragmento de los blueprints del menú de pausa

La última funcionalidad implementada será la de mayor complejidad. Como se determinó en los requisitos de Malaca, el usuario podrá interactuar con una serie de figuras repartidas por el escenario. Estas figuras podrán ser recogidas y devueltas a su pedestal, pero, mientras, el usuario las portará. Para esto deberá existir un punto dentro del blueprint del personaje jugable que sea donde se colocan las figuras mientras son transportadas. Este punto está por debajo de la cámara y algo alejado de la misma para que no obstaculice la visión del jugador. Se llamará **FigureHoldPoint**.

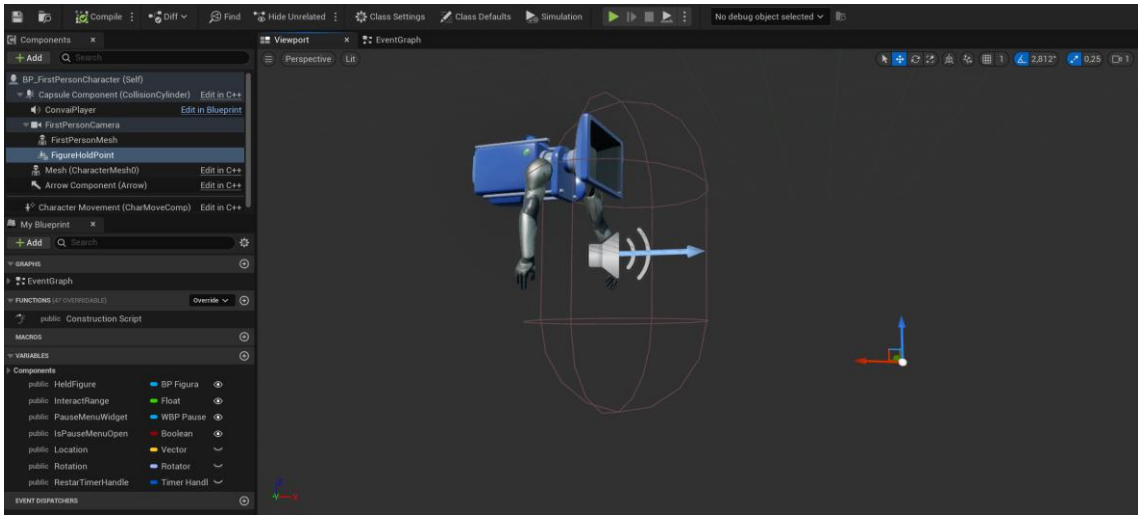

Fig. 25. *FigureHoldPoint* en el personaje jugable

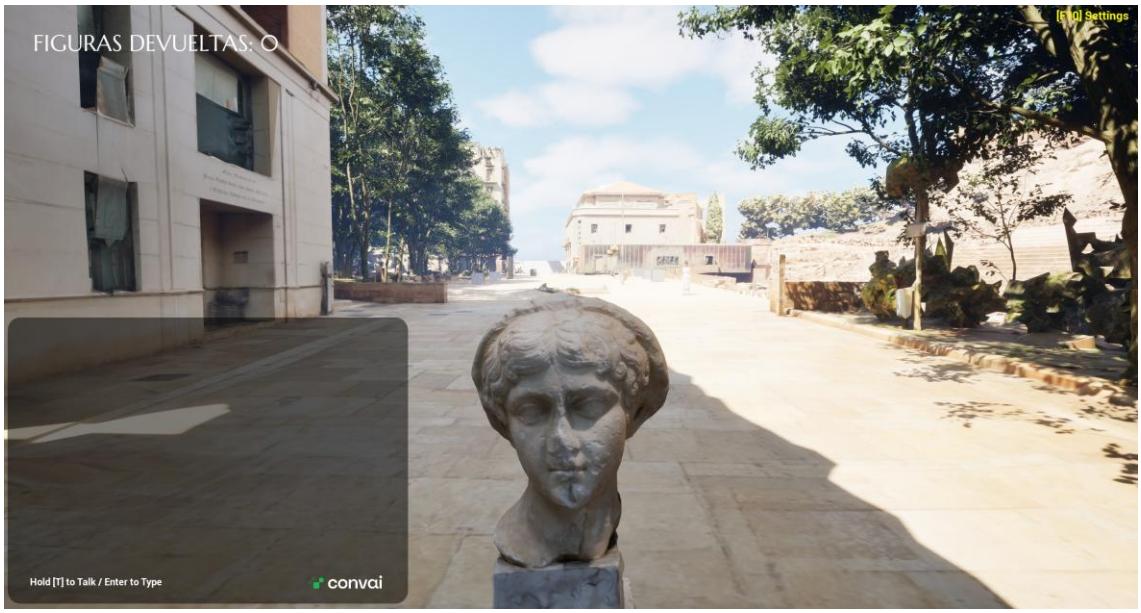

Fig. 26. Captura del usuario portando una figura

Cuando el jugador pulse el botón izquierdo se llamará a la función *LineTraceByChannel*, la cual tomará como parámetros la ubicación hacia la que el usuario está mirando actualmente y la multiplicará por el float **InteractRange**. Esta variable determina hasta qué distancia podemos coger una figura. La función *LineTraceByChannel* dibuja líneas invisibles, (o visibles si se desea durante el desarrollo) las cuales determinan cuando el jugador coge exitosamente una figura. Para que la acción de recoger sea realizada el usuario, este debe estar dentro del rango de interacción y además tiene que estar mirando a la figura. Si esto se cumple, se realiza una comprobación rápida de si el personaje ya porta en ese momento una figura mirando en la variable **HeldFigure** (una referencia a la figura que el personaje porta, será Null si no lleva ninguna).

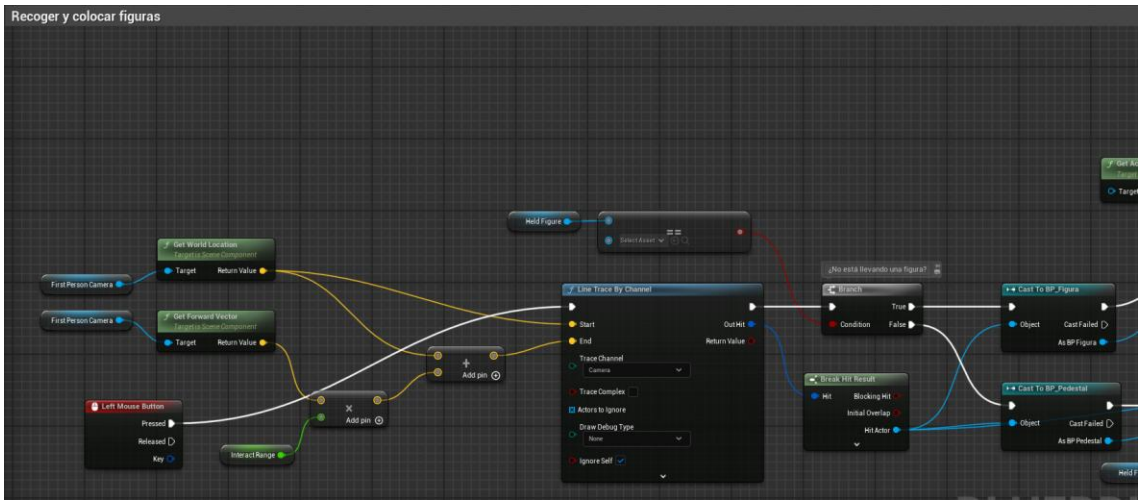

Fig. 27. Fragmento del manejo de figuras en *BP_FirstPersonCharacter*

Si el jugador no está llevando una figura, se cambia el valor de HeldFigure al de la pieza actual y se llama a la función **AttachActorToComponent**. Esta función coloca la figura en el *FigureHoldPoint*, además de hacer que esta se mueva con el usuario.

Después, se desactiva la colisión de la figura para evitar algunos errores que surgen cuando el pedestal y la figura colisionan y se oculta el texto del *WBP_IteractionFigure*. Este widget será explicado en los siguientes puntos.

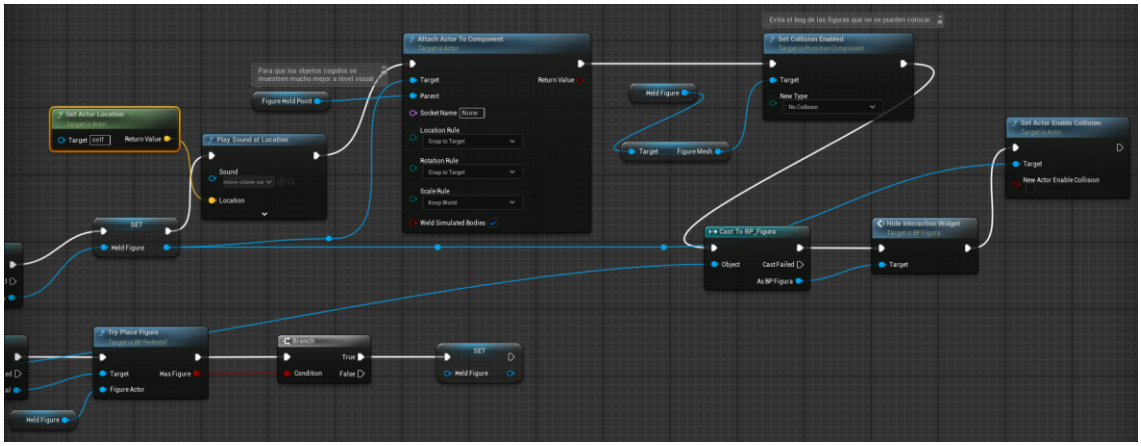

Fig. 28. Parte final del manejo de figuras en *BP_FirstPersonCharacter*

Si el personaje jugable porta ya una figura, esta intentará ser colocada en un pedestal (condición falsa del Branch) llamando a la función **TryPlaceFigure** de BP_Pedestal. El funcionamiento de ésta será explicado en los siguientes puntos.

4.2. Escenario

El lugar donde se va a realizar la visita virtual es la calle Alcazabilla, en el actual centro histórico de Málaga. En esta calle es donde se encuentra el teatro romano, a las faldas de la Alcazaba. Para la realización de este proyecto, la empresa Visitas Virtuales [10] ha aportado el modelo 3D de la calle Alcazabilla, el cual es la base sobre la que se ha construido Malaca. Este modelo fue realizado con técnicas de fotogrametría, empleando diversos dispositivos como drones para su obtención.

A continuación, se explicará el proceso de modificación y mejora del escenario. Estas actualizaciones no han sido sólo realizadas para mejorar el aspecto visual de la visita virtual, sino para el correcto funcionamiento del resto de elementos implementados.

4.2.1. Configuración y optimización

El principal problema encontrado al importar el modelo en Unreal Engine fueron las colisiones de este, ya que el motor gráfico reconocía todo el escenario como un bloque unido, apreciable en la Fig. 29. Esto hacía imposible que un jugador pudiera moverse por la visita virtual.

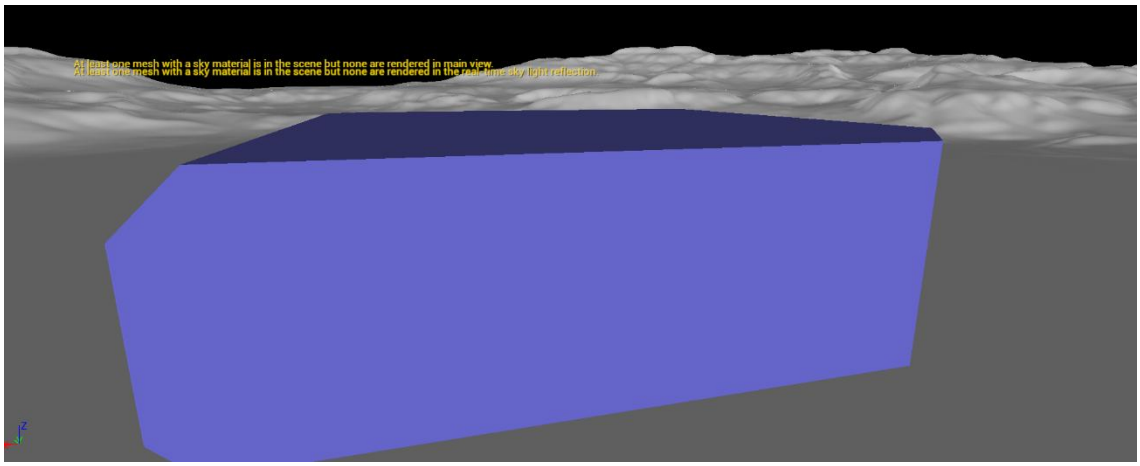

Fig. 29. Colisión por defecto del mesh del teatro

Para abordar este problema se decidió separar el modelo del teatro en dos, creando dos **SubMesh**, uno que sería el propio suelo físico de la visita y otro que fuera el resto, principalmente los edificios y el terreno de la Alcazaba. Para separar el modelo en dos se usó la herramienta **Triangle Select**, dentro de Unreal Engine, la cual se empleó para delimitar las zonas del mesh que iban a separarse. Tras realizarse esto, se desactivaron las colisiones del modelo que no era el suelo físico. Gracias a esto, un personaje jugable ya puede moverse sin problemas por el escenario.

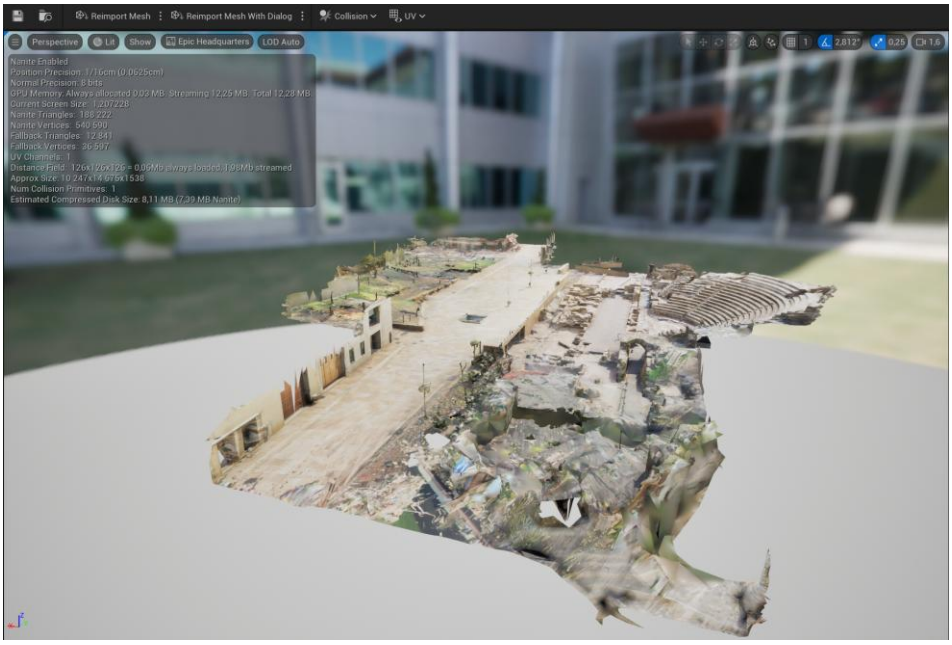

Fig. 30. Submesh del modelo del teatro

Otro punto tenido en cuenta en la modificación del modelo fue la optimización. El modelo entregado por Visitas Virtuales tenía una grandísima cantidad de triángulos y el triple de vértices. Los triángulos superaban el millón y medio, lo cual suponía una carga muy alta de rendimiento. Para paliar esto se redujo su número tanto en el *SubMesh* de la calle como en el de los alrededores. Como se puede ver en Fig. 31, el número de triángulos fue reducido en gran medida, buscando un balance entre calidad gráfica y rendimiento.

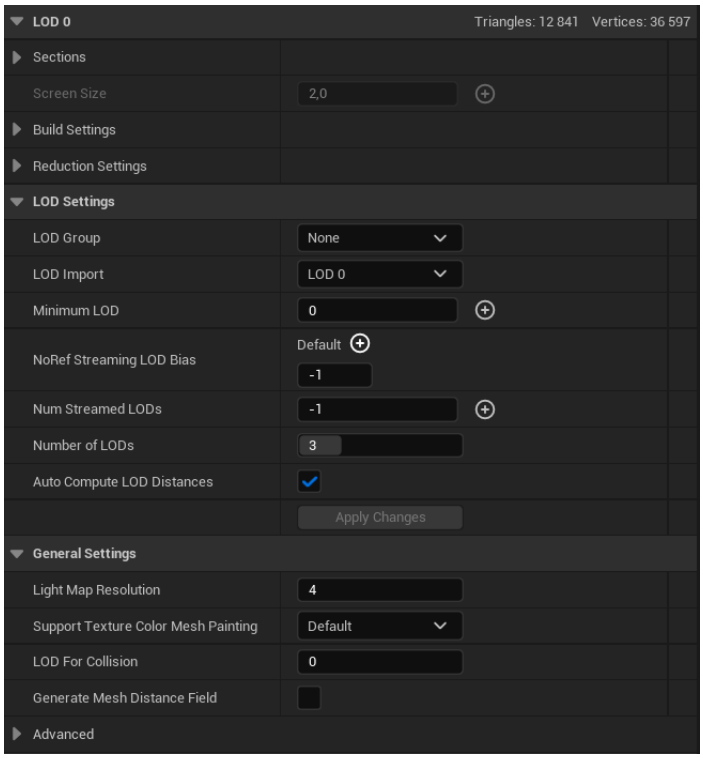

Fig. 31. Parámetros editados del *SubMesh* de la calle Alcazabilla

Otra optimización tenida en cuenta fueron los LOD (Level of Detail), una serie de técnicas empleadas para reducir la complejidad de un modelo 3D a medida que se aleja del espectador. El LOD0 se trata del modelo base y conforme el número aumenta (LOD1, LOD2, ...), la calidad gráfica desciende. Gracias a esta técnica se mejora el rendimiento de la visita virtual al liberar recursos del sistema para otras tareas que lo demanden. En el caso de Malaca, se ha empleado hasta un LOD3 en el modelo del escenario.

4.2.2. Mejoras visuales

Una vez perfeccionado el rendimiento del modelo otorgado, el siguiente paso fue la mejora visual del mismo. A esta tarea se dedicó más tiempo del planificado en las primeras fases del proyecto. Se tuvieron que realizar muchas modificaciones al mesh para llegar al aspecto actual, como se puede observar en la Fig. 33.


Fig. 32. Aspecto sin modificar del teatro

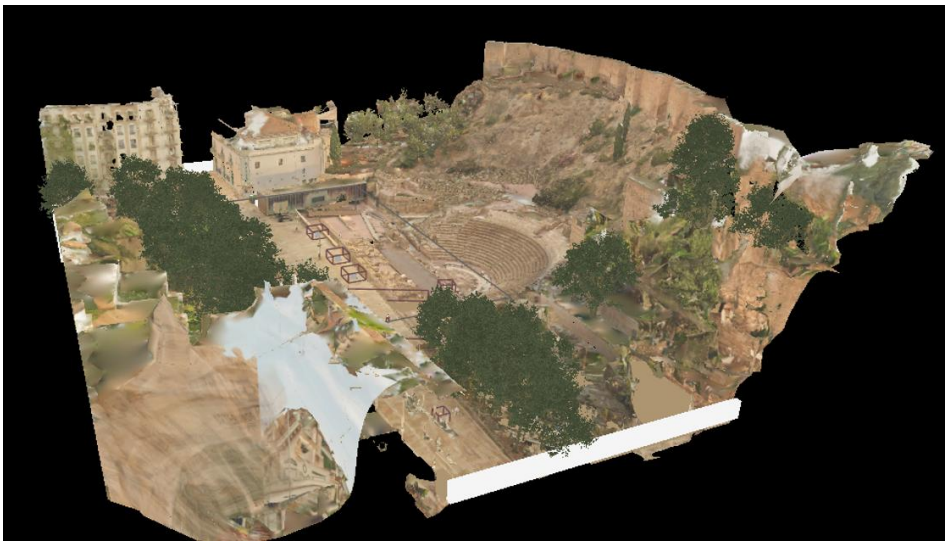

Fig. 33. Aspecto final del teatro

El modelo 3D usado para Malaca goza de una muy buena calidad, pero en sus primeras versiones contenía muchos elementos poco atractivos visualmente. Estos elementos podían ir desde agujeros en el suelo (los cuales provocaban problemas en la malla de navegación del metahuman) hasta extrañas formas geométricas de color blanco. Todo este ruido tuvo que ser eliminado manualmente. Para comenzar, se usó el **Modeling Mode** de Unreal Engine, el cual se usó también para separar el mesh en dos. En este caso se usó el *Triangle Select* para eliminar aquellas partes poco atractivas.

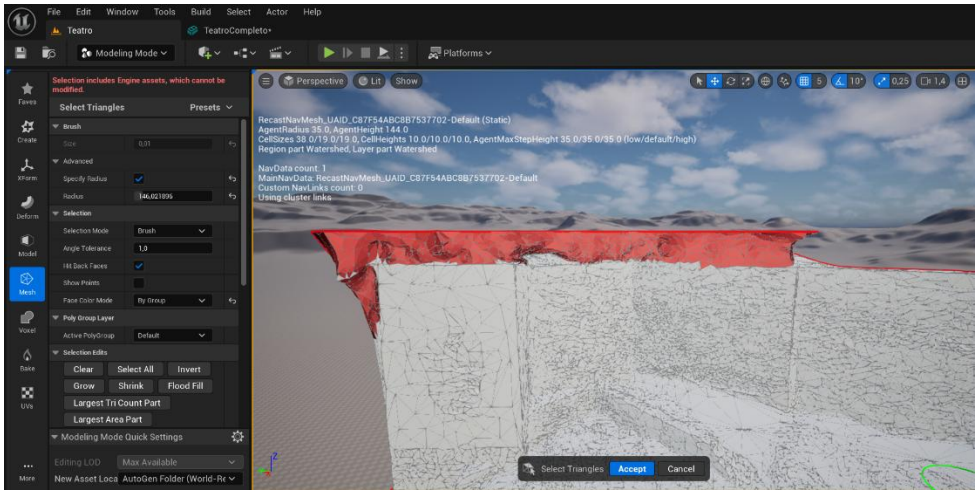

Fig. 34. Herramienta *TriangleSelect* en Unreal

Existían ciertos elementos del modelo para los cuales esta herramienta no era lo suficientemente útil, es por eso por lo que se volvió a emplear *Blender* para tapar agujeros y aplanar el terreno. La combinación de estas dos herramientas permitió una gran mejora visual de la visita virtual.

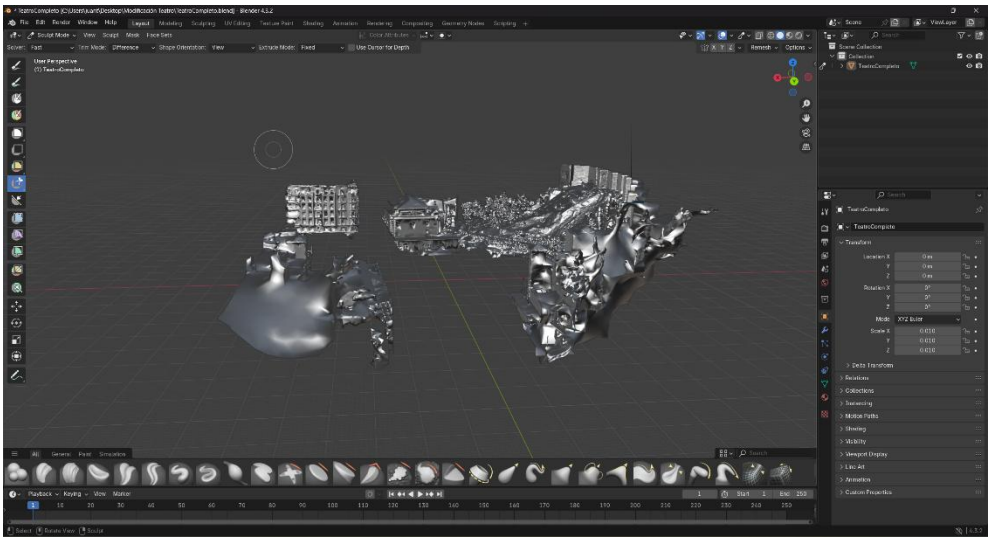

Fig. 35. Modificaciones del modelo en *Blender*

Para finalizar, se añadieron assets de árboles gratuitos para compensar la eliminación de los árboles escaneados.

4.3. Figuras interactivas

En este apartado se va a explicar cómo se han implementado las figuras y los pedestales con los que el usuario puede interactuar en Malaca. Se separará la implementación en dos partes, una que tratará sobre el modelado y creación de las figuras y otra que explicará el código realizado para estas.

4.3.1. Creación de los modelos

La particularidad personalizada de este proyecto (la Málaga romana) ha hecho que se deba dedicar tiempo a la realización de las figuras que estarán repartidas por el escenario y serán el tema central de la visita virtual. Se optó por crear estos modelos en vez de obtenerlos de forma libre en internet con el objetivo de dar una mayor calidad a la experiencia. Todas las figuras realizadas para Malaca (excepto la Lex Flavia que es tan sólo una imagen sobre un mesh) pueden encontrarse actualmente en el museo de Málaga [32], en la sección de la Málaga romana.

Para obtener modelos 3D de estos objetos se realizaron varios vídeos en el museo de Málaga. El autor del proyecto realizó vídeos de una gran variedad de figuras, mosaicos e inscripciones con un teléfono móvil, para luego seleccionar los mejores para Malaca. Estos vídeos intentaron capturar la mayor parte de los ángulos de las figuras para que al generar los modelos, estos tuvieran una mayor calidad. Los vídeos realizados luego fueron separados en imágenes con la herramienta **RealityCapture** [18] (ahora RealityScan). Tras limpiar las imágenes, se crearon los modelos. *RealityCapture* usa técnicas de fotogrametría, empleando las fotografías para crear el modelo desde distintos ángulos, cosa apreciable en la Fig. 37 en los puntos blancos alrededor del busto.

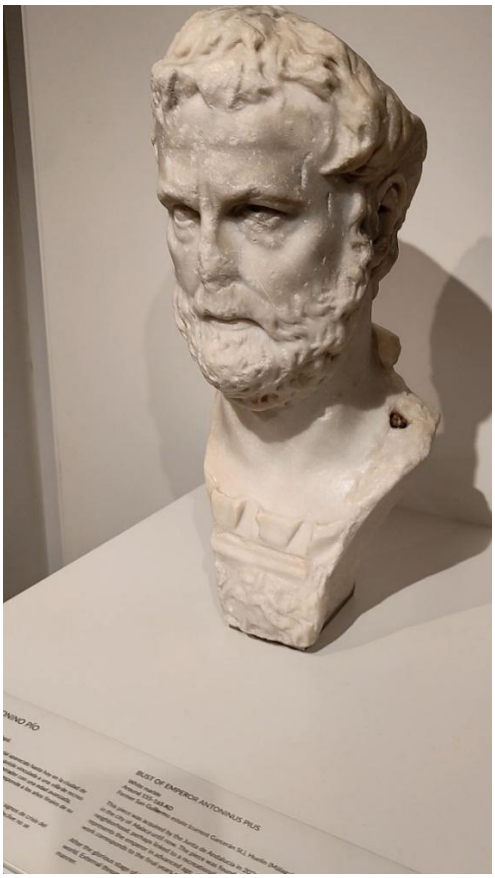

Fig. 36. Fotograma de uno de los vídeos realizados

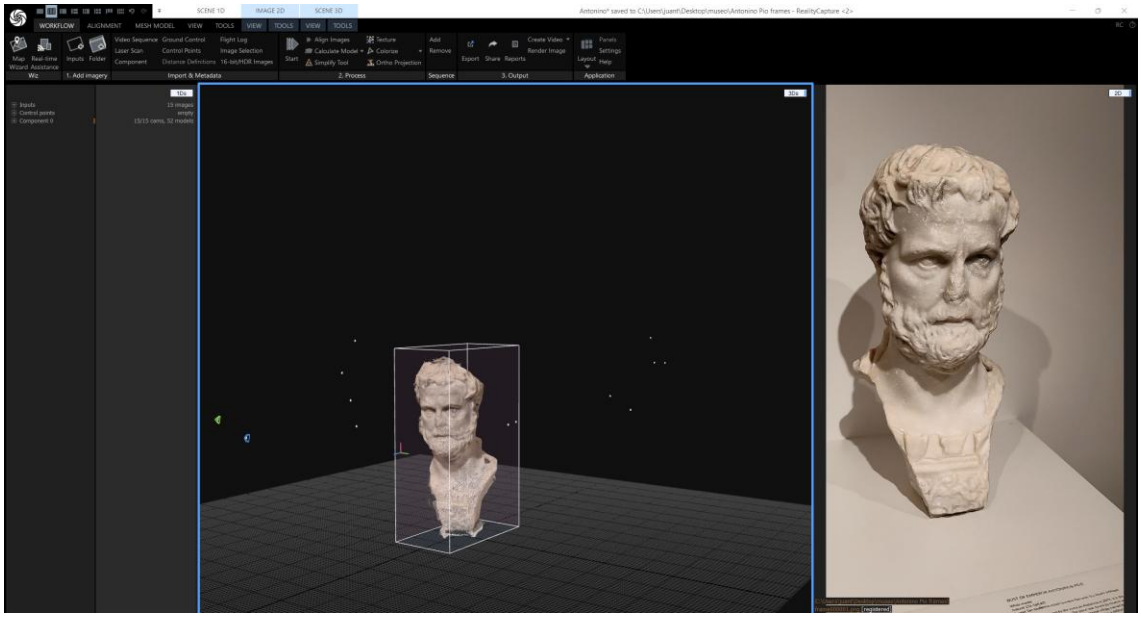

Fig. 37. Creación del modelo en *Reality Capture*

Una vez generada la figura necesaria, es importante eliminar los defectos o ruido alrededor de esta. Para ello se empleará la herramienta *lasso* mientras se van tapando los agujeros generados con la opción *Close Holes*. Una vez que el modelo esté listo, se seleccionará la opción **Textures** para que se guarden las texturas necesarias. Finalmente, se exportará el modelo en formato *.obj*.

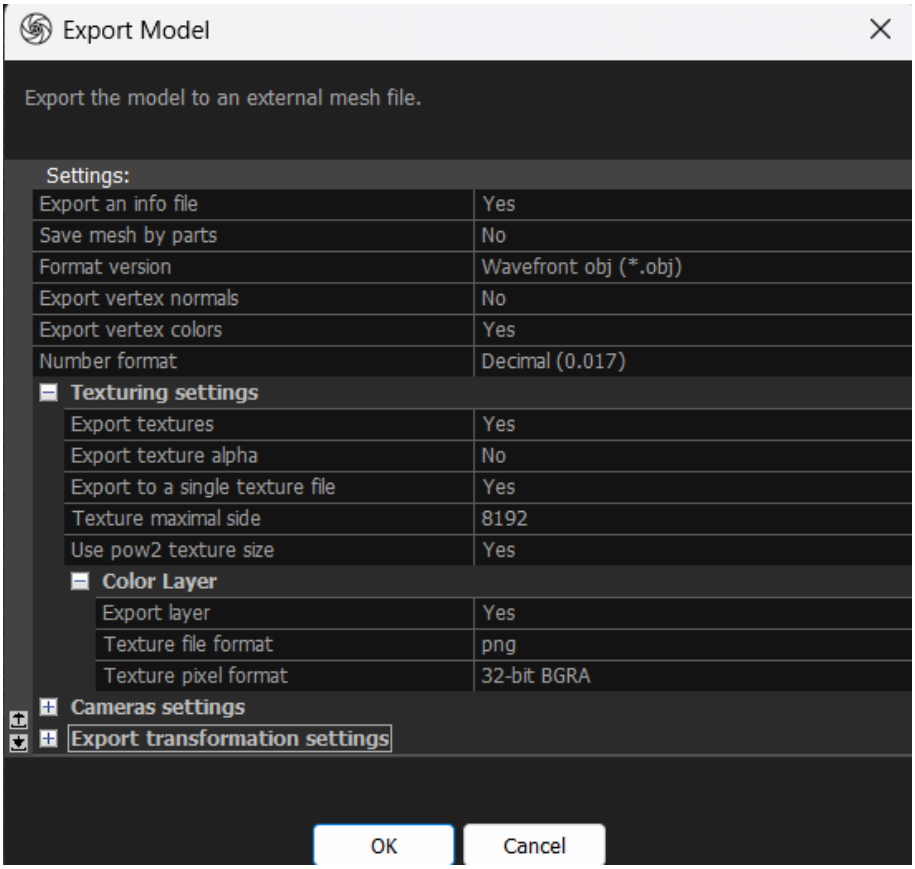

Fig. 38. Ajustes de exportación en *Reality Capture*

El siguiente paso consistirá en modificar el modelo en *Blender* para mejorar su aspecto visual. Al igual que ocurre en Unreal Engine, *RealityCapture* tiene sus limitaciones a la hora de modificar modelos 3D, por lo que se usará *Blender* para tapar los agujeros que queden, añadir material donde falte y mover vértices individualmente. Todo esto se realizará para que las figuras terminen con un acabado más profesional.

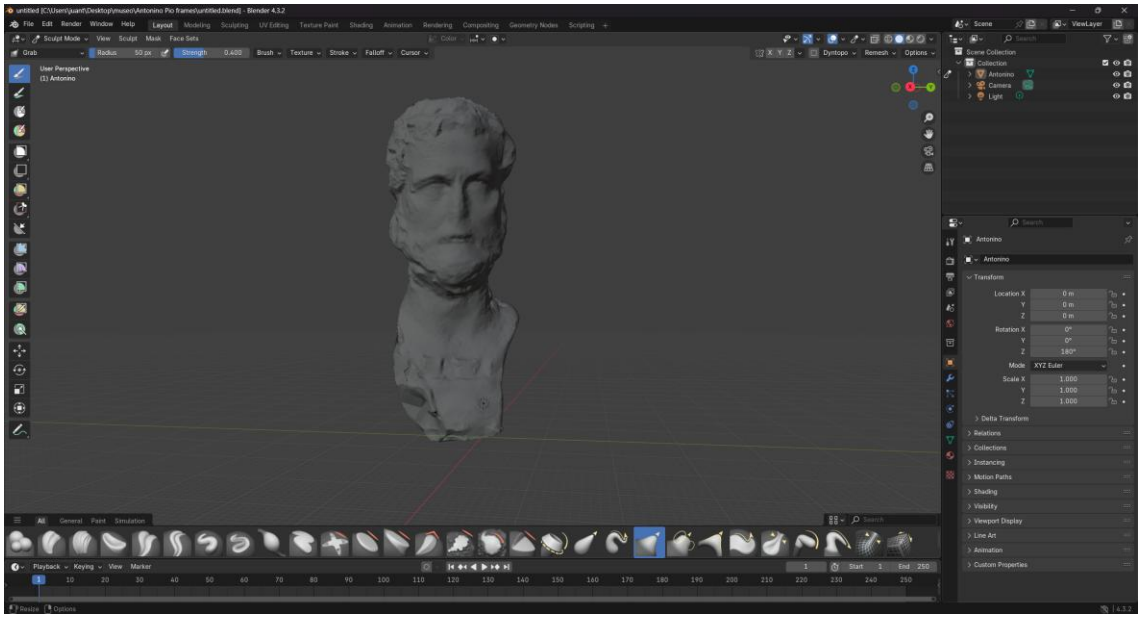

Fig. 39. Edición del modelo en *Blender*

Se aplicarán estos pasos en todas las figuras realizadas, obteniendo como resultado la Fig. 40, unos modelos 3D importados en Unreal Engine con una gran calidad.

Fig. 40. Resultado final de las figuras

Por último, cabe mencionar que la información de cada figura ha sido obtenida de una mezcla de las descripciones del museo de Málaga, como por ejemplo la del busto de Antonino Pío [33], con los conocimientos del autor de este proyecto. Se ha llevado a cabo una simplificación en muchos casos los datos para un mayor disfrute del usuario.

4.3.2. Implementación de las funcionalidades

Ahora, se explicará qué código se ha creado en blueprints para las figuras realizadas. También se expondrá el funcionamiento del código para los pedestales donde las figuras irán colocadas y los *Widget Blueprints* que mostrarán al usuario información importante.

4.3.2.1. BP_Figura

Se creará un blueprint llamado **BP_Figura**, el cual almacenará las funcionalidades que tendrán todas las figuras interactivas de Malaca. Esta clase será la clase padre del resto, por lo que es de especial importancia su buena configuración.

Para comenzar, tras crear el blueprint crearemos la variable booleana *bIsHeld*, la cual será por defecto falsa. Esta variable comunicará cuándo está siendo transportada la figura para realizar una serie de acciones. Se añadirá un **Static Mesh Component** para probar el blueprint y que tenga un aspecto visual, pero cuando se termine la implementación se pondrá a *None*. También será necesario añadir un **Box Collision** alrededor de la figura, el cual determinará el rango al que aparece este texto: Clic izquierdo para recoger. Por último, se añadirá un **Widget Component** parar gestionar este texto más adelante.

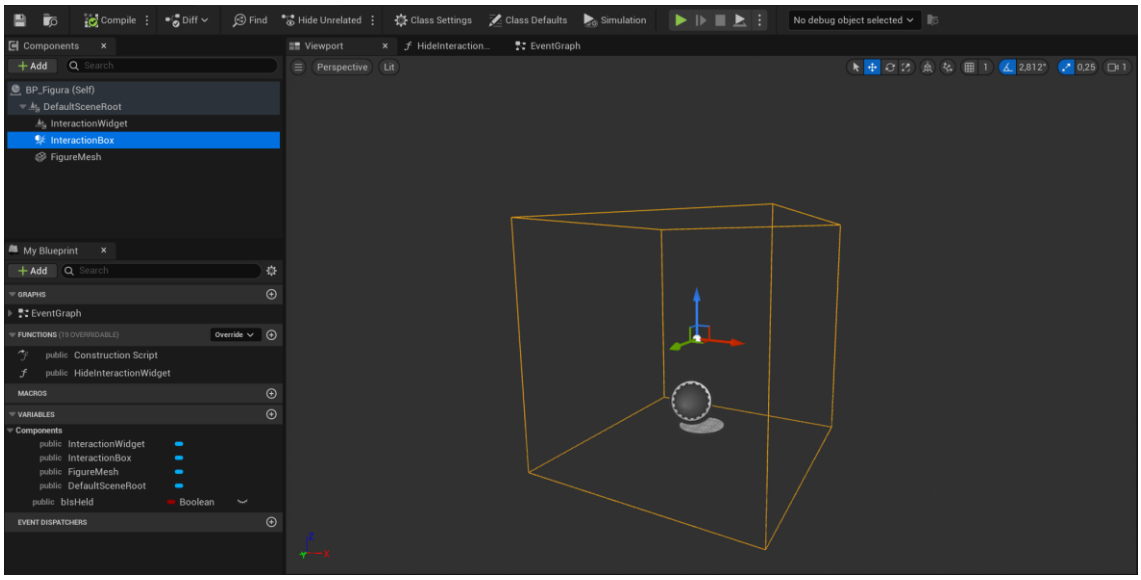

Fig. 41. Viewport de *BP_Figura*

Antes de continuar, se creará el widget **WBP_InteractionFigure**, el cual sólo tendrá el texto a mostrar cuando el usuario se acerque a una figura lo suficiente. Todas sus funcionalidades serán manejadas dentro del blueprint de la figura. Cuando se ejecute la visita virtual, el texto se ocultará automáticamente, como se muestra en la Fig. 43.

Fig. 42. Pestaña *Designer* de *WBP_InteractionFigure*

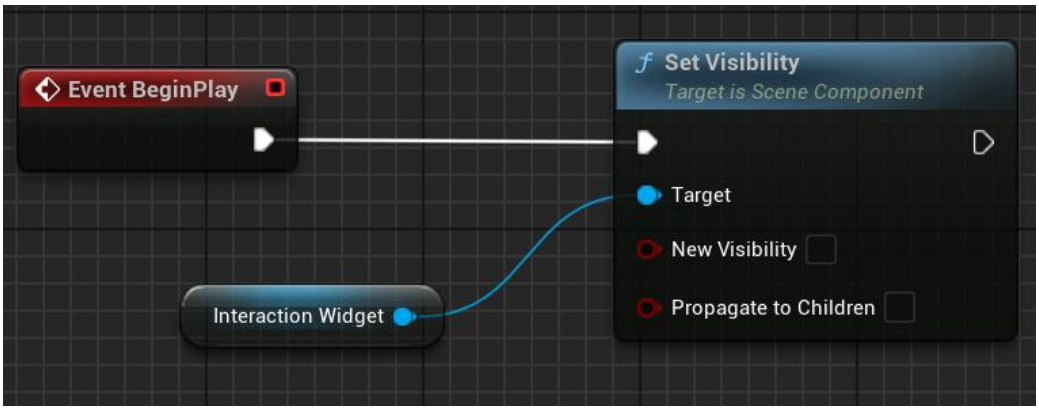

Fig. 43. *Event BeginPlay* de *BP_Figura*

Se deberá tener también en cuenta el manejo de la ocultación y la muestra del texto, esta última tendrá lugar cuando el usuario entre en el rango de la *InteractionBox*. Esto disparará un evento que llamará a la función *SetVisibility* para mostrar el texto de nuevo. La gestión de la ocultación del texto será realizada de igual forma, pero, al contrario. Será también necesario realizar un *cast* a *BP_FirstPersonCharacter* para determinar quién solapa la colisión (aunque siempre será el usuario como resulta evidente).

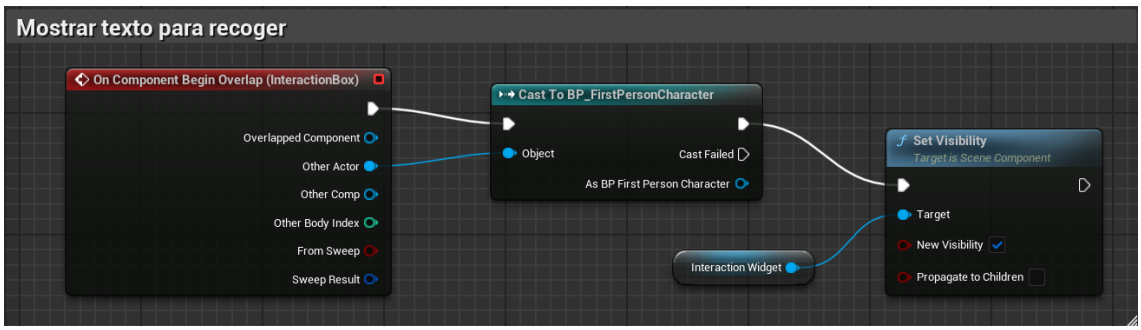

Fig. 44. Manejo del texto en *BP_Figura*

El siguiente paso será crear una clase hijo de *BP_Figura* que, como se ha comentado anteriormente, heredará las funcionalidades de su clase padre. Dentro de cada uno de estos nuevos blueprints se introducirá un modelo 3D como se puede apreciar en la Fig. 46. El *FigureMesh* se mantendrá a *None* como una forma de estandarizar las colisiones de todas las figuras interactivas.

Fig. 45. Creación de una clase hijo

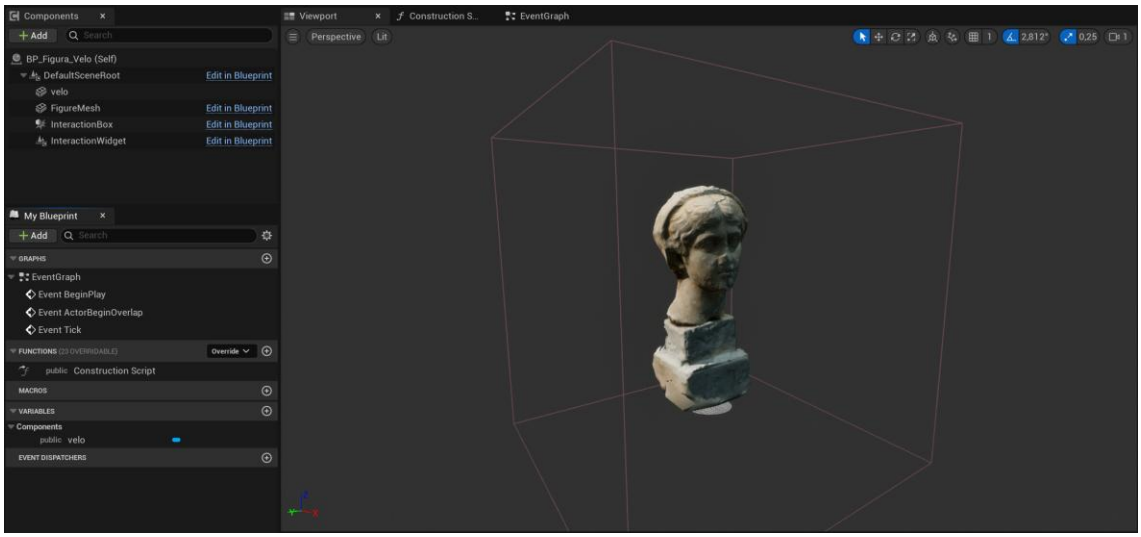

Fig. 46. Vista de *BP_Figura_Velo*

4.3.2.2. BP_Pedestal

La implementación de este blueprint será mucho más compleja que la de *BP_Figura* ya que el pedestal debe controlar una gran variedad de cosas. Las variables que tendrá este blueprint serán el booleano *bHasFigure*, el cual controlará que haya o no una figura colocada en él y una referencia a *BP_Figura* llamada **ExpectedFigureClass**, que servirá para saber qué figura debe ir en cada pedestal. También se guardará una referencia en texto de esto último para un manejo más rápido de algunas funcionalidades. Por último, habrá una referencia guardada al metahuman para el lanzamiento de los *triggers*. Esta funcionalidad será explicada en el punto 4.5 y no ahora.

El aspecto visual del pedestal será el de un cubo de color gris por lo que se cambiará el *mesh* debidamente. Se continuará introduciendo en el blueprint un *Box Collision* como en las figuras, pero en este caso para determinar a qué distancia aparecerá un texto que contiene el nombre de la figura de ese pedestal. Se introducirá un *Widget Component* para esto último además de un *Scene Component* llamado **PlacementPoint**. Este componente guarda el lugar exacto donde se deben colocar las figuras cuando el usuario intenta ponerlas en los pedestales.

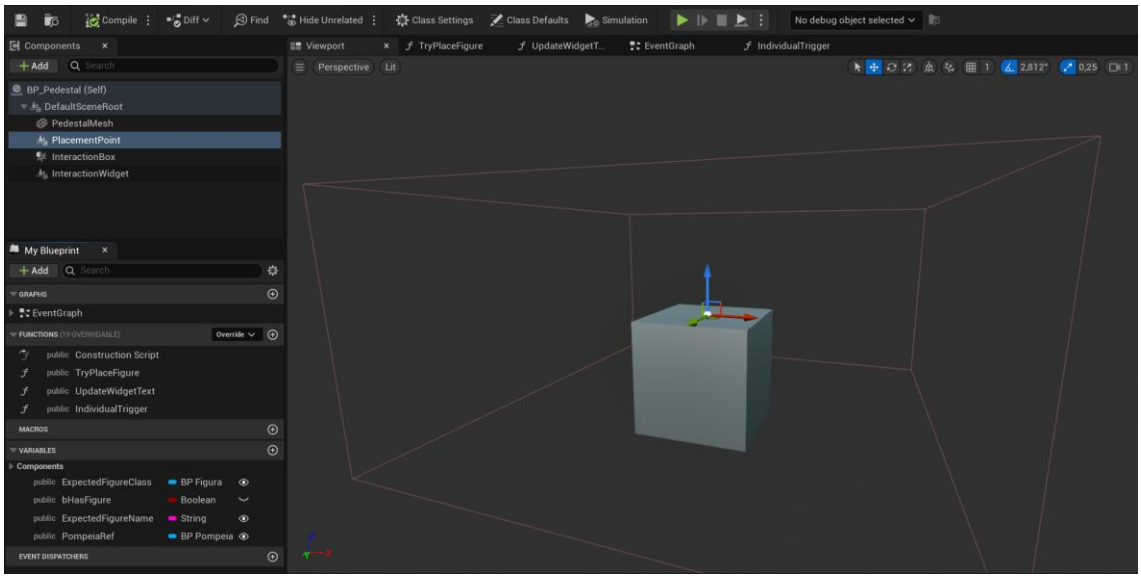

Fig. 47. Viewport de *BP_Pedestal*

Como en *BP_Figura*, el siguiente paso será la creación de un *Widget Blueprint*, en este caso uno que mostrará en cada pedestal qué figura se está esperando. Tras diseñarlo, se accederá a la pestaña *Graph* y se crearán dos funciones.

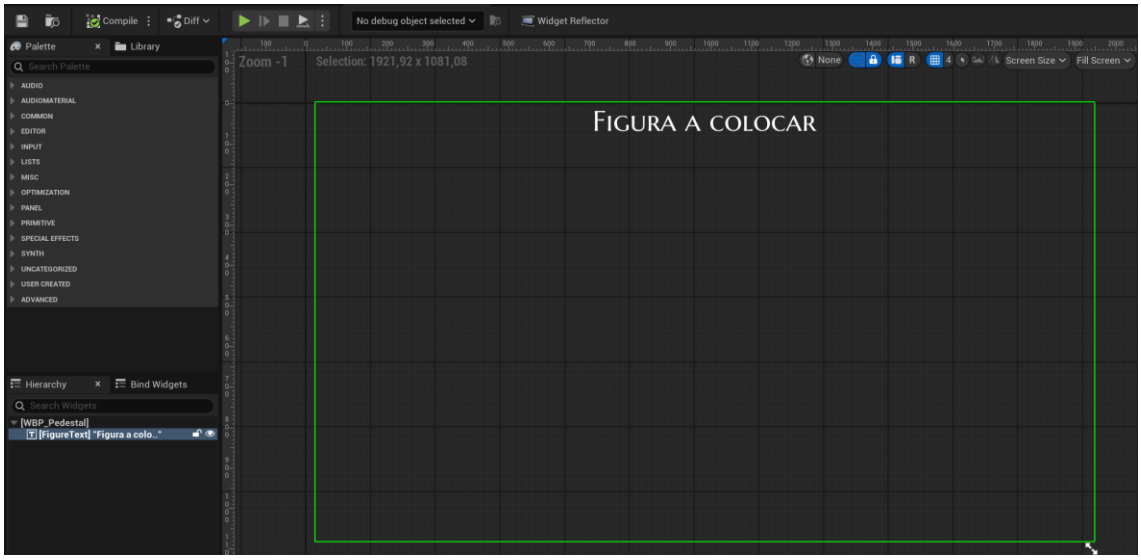

Fig. 48. Pestaña *Designer* de *WBP_Pedestal*

Por un lado, estará la función *SetExpectedFigureName*, la cual tomará como parámetro un String que contiene el nombre de la figura que se espera en ese pedestal. Esta función servirá de forma auxiliar para que cada texto, según el pedestal, sea distinto.

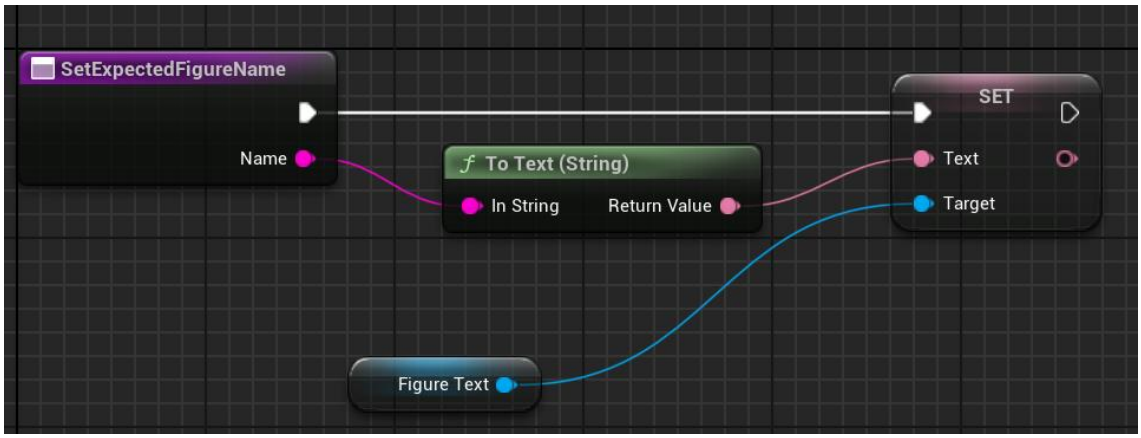

Fig. 49. Función *SetExpectedFigureName*

Fig. 50. Captura de un texto en un pedestal

Fig. 51. Un pedestal que espera la figura del mosaico

Una vez que el *WBP_Pedestal* esté listo, el siguiente paso continuará con la implementación de *BP_Pedestal*, como se puede apreciar en la Fig. 52. En el Event BeginPlay se guardará una referencia a Pompeia para más adelante. Luego, se pasará a ocultar el *Widget Blueprint* y se llamará a la función *SetExpectedFigureName* para saber qué figura espera ese pedestal. El manejo de la visibilidad del texto será igual que en blueprint de las figuras.

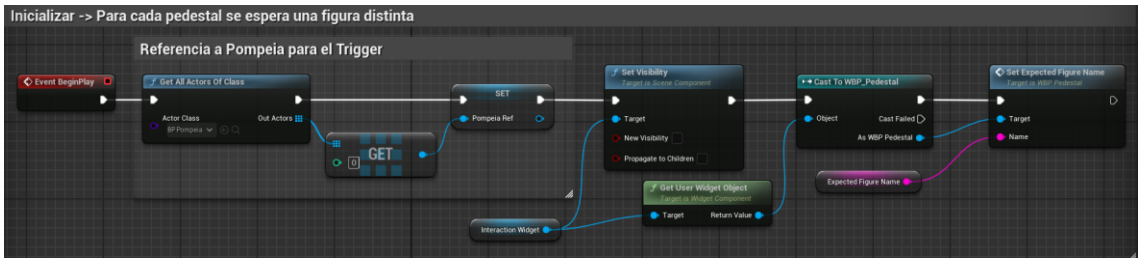

Fig. 52. Event BeginPlay de *BP_Pedestal*

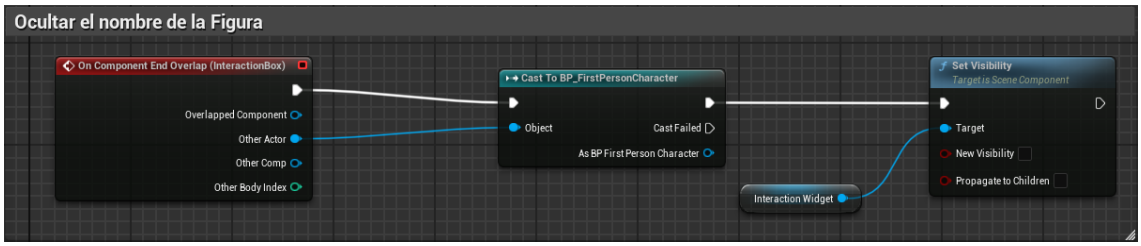

Fig. 53. Manejo de la visibilidad del texto del pedestal

El siguiente paso de la implementación será la creación de la función **TryPlaceFigure**. Al comenzar, la función mirará el valor del booleano *bHasFigure* para comprobar si el propio pedestal ya tiene una figura. Si no la tiene, se comprobará en otro *branch* si la figura que el usuario está intentando colocar coincide con la que se espera. Para hacer esto se realizará un *cast* a *BP_Figura* y se comprobará si el valor devuelto coincide con la referencia que guarda el pedestal (la correcta).

Si la figura que se intenta colocar no coincide, se reproducirá un sonido de error para hacérselo saber al usuario. Si el la correcta, se llamará a **SetActorLocation** para cambiar la ubicación de la figura con la del *PlacementPoint* descrito anteriormente. Se reactivarán las colisiones de la figura, las cuales habían sido desactivadas en el momento de recogerla para evitar una serie de errores. Finalmente, se hará un **AttachActorToComponent** entre la figura y el *PlacementPoint*, de forma similar a cuando el usuario recoge la figura del suelo.

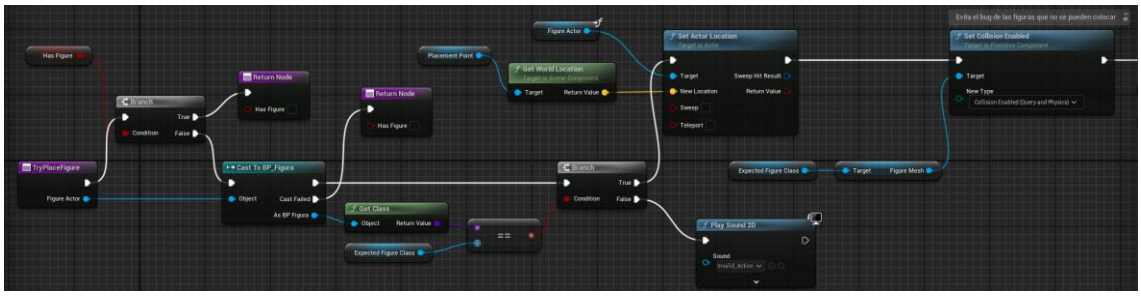

Fig. 54. Fragmento de *TryPlaceFigure*

La siguiente parte del código de la función se encargará de ejecutar un sonido cuando la figura es colocada. También se actualizará el contador de figuras devueltas llamando a la función **AddReturnedFigure** y se lanzará un *trigger* para el chatbot. La explicación de estas funcionalidades será realizada más adelante.

Fig. 55. Llamada a *AddReturnedFigure* en *BP_Pedestal*

4.4. Menús e interfaces

A continuación, se explicará la integración e implementación de los menús y elementos de la interfaz de Malaca. Se ha empleado como fuente para este proyecto *Marcellus SC* [34], descargada de forma gratuita para que la interfaz tenga un aspecto más "romano".

4.4.1. Menú Principal

Para los menús, no se ha empleado el mismo tipo de blueprints que hasta ahora; se han usado los denominados **Widget Blueprints**. Además, cabe mencionar que se ha creado un blueprint de GameMode llamado **GM_MainMenu**, el cual se encargará de gestionar el menú principal. Se explicará más sobre los modos de juego más adelante.

Este GameMode creará un widget de **WBP_MainMenu**, un *Widget Blueprint* que se explicará a continuación y le dará al usuario el control sobre el cursor del ratón. Para mostrarle al usuario el cursor del ratón, se llamará a la función *getPlayerControllerFromInputDevide*, que devolverá el id del dispositivo de input que está usando. Luego, ese id es llevado a la función *bShowMouseCursor*, la cual activará la visibilidad del cursor.

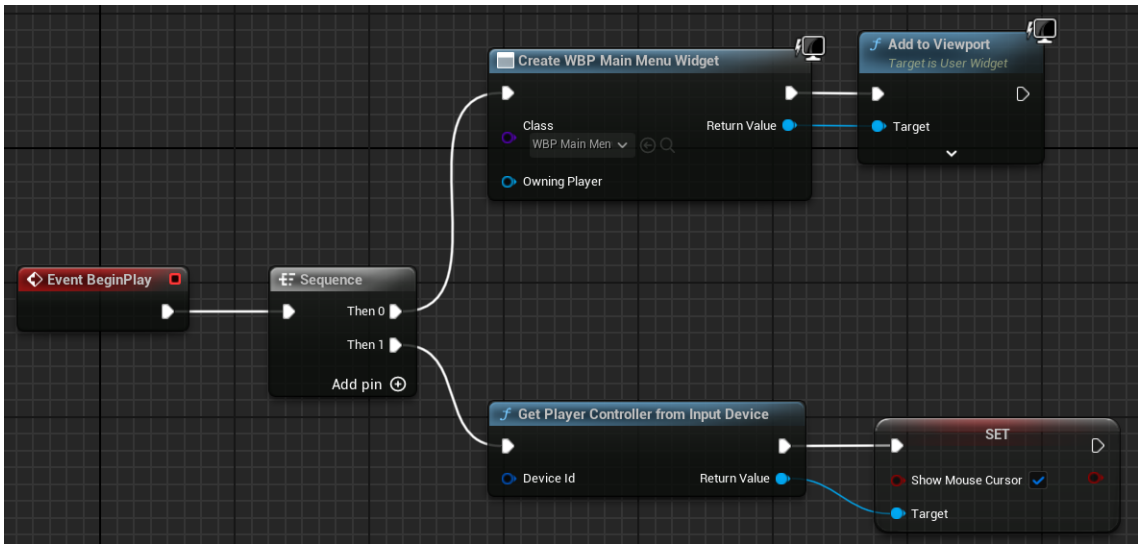

Fig. 56. Event Graph de *GM_MainMenu*

El primer paso para crear este blueprint será realizar el diseño dentro de su pestaña *Designer*, tanto el del propio menú principal como el de la pestaña de controles (que será por defecto invisible).

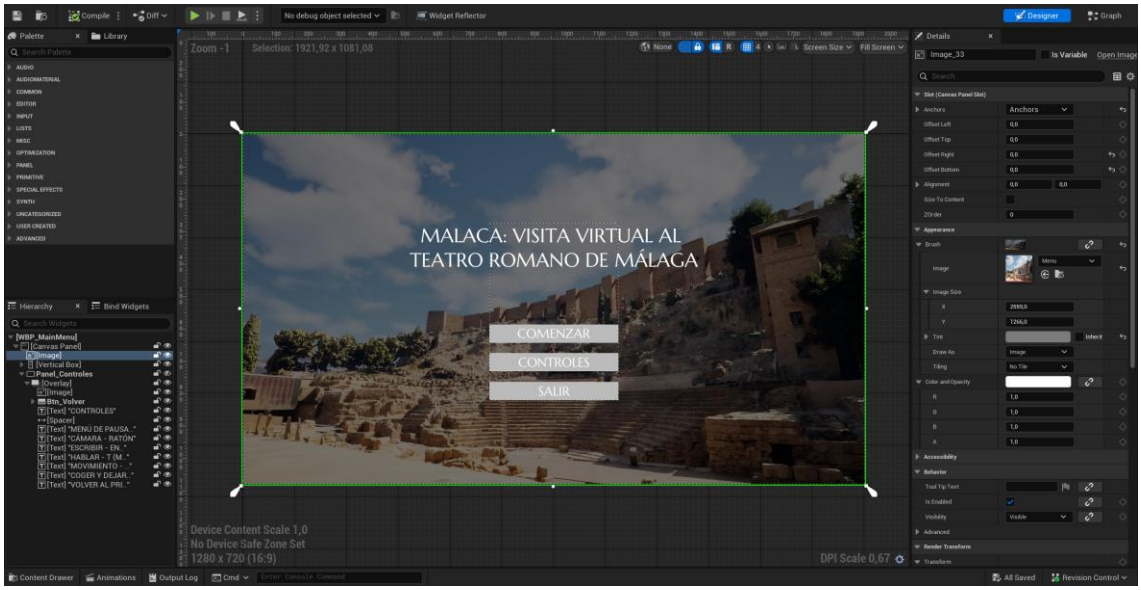

Fig. 57. Diseño de *WBP_MainMenu*

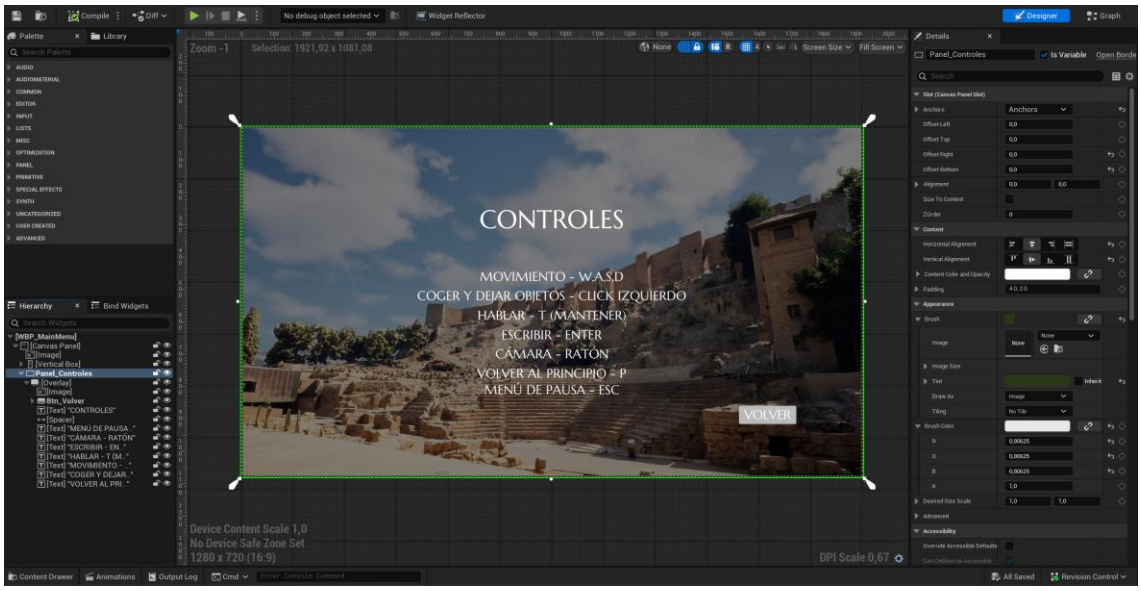

Fig. 58. Diseño de la pestaña de controles

Tras esto, se podrá pasar a la implementación de los tres botones creados en esta interfaz. Cabe mencionar que todos los botones generan un sonido 2D cuando se les pulsa con el objetivo de que haya un feedback inmediato.

Fig. 59. Variables de *WBP_MainMenu*

4.4.1.1. Botón de Comenzar

Se comenzará implementando el botón de *Comenzar*, el cual realizará un *OpenLevel* del nivel donde se encuentra el teatro romano.

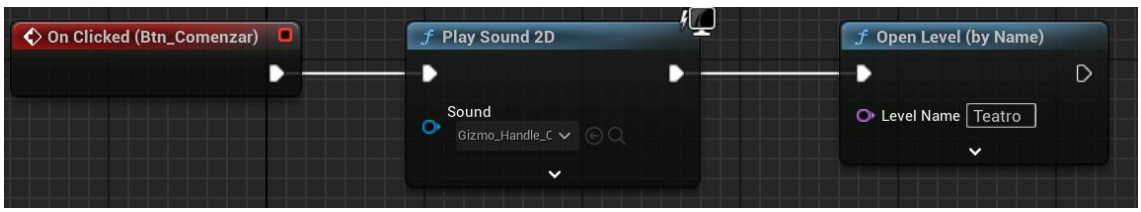

Fig. 60. Botón de *Comenzar* en el Menú Principal

4.4.1.2. Botón de Controles

Para continuar, se implementará en botón que activa el panel de controles. Este botón cambia la visibilidad del panel, que es por defecto invisible, para que el usuario pueda verlo.

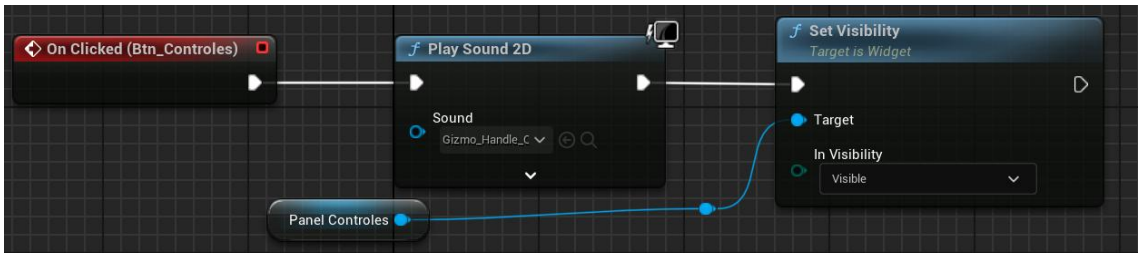

Fig. 61. Botón de *Controles* en el Menú Principal

4.4.1.3. Botón de Volver

Este botón solo se muestra cuando el usuario ha activado el panel de controles. Pulsándolo, el panel volverá a ser invisible y el usuario podrá volver a interactuar con el menú principal.

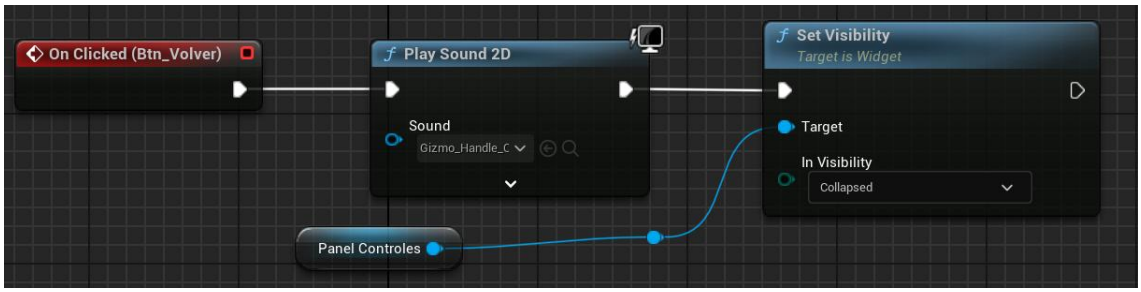

Fig. 62. Botón de *Volver* en el Menú Principal

4.4.1.4. Botón de Salir

Este botón sólo llama a la función *QuitGame*, tomando como parámetro el controlador del jugador.

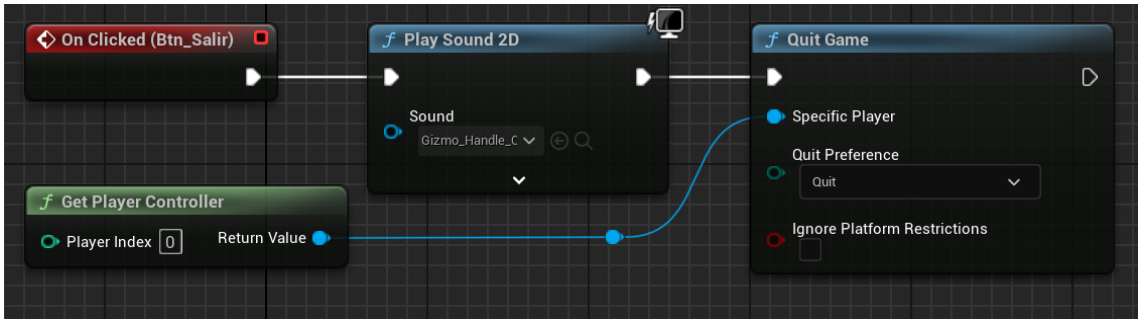

Fig. 63. Botón de *Salir* en el Menú Principal

4.4.2. Menú de Pausa

Al igual que con el menú principal, el primer paso para crear este blueprint será realizar el diseño dentro de su pestaña *Designer*, eligiendo en este caso un menú de pausa que difumina el entorno. El nombre de este blueprint será **WBP_PauseMenu**. La pestaña de controles será exactamente igual que la del menú principal y los botones también producirán sonidos cuando son pulsados.

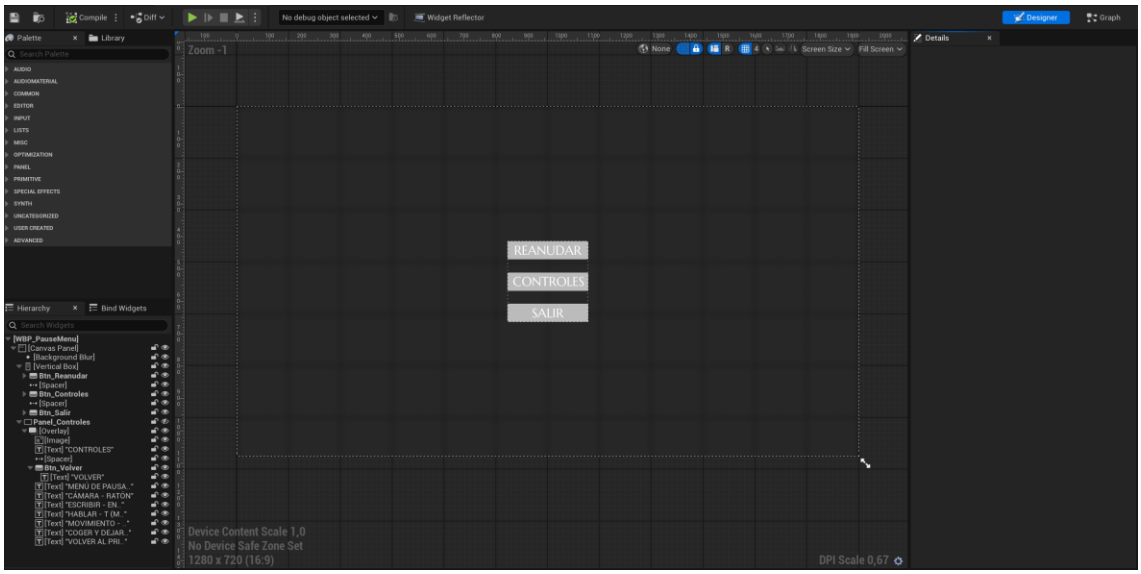

Fig. 64. Diseño de *WBP_PauseMenu*

Fig. 65. Variables de *WBP_PauseMenu*

4.4.2.1. Botón de Reanudar

La principal diferencia de este botón con el que se encuentra en el menú principal es que se tiene que realizar un *Cast* del personaje del usuario para cerrar el menú de pausa. Esto se debe a que el cierre de éste se gestiona desde el blueprint del personaje jugable.

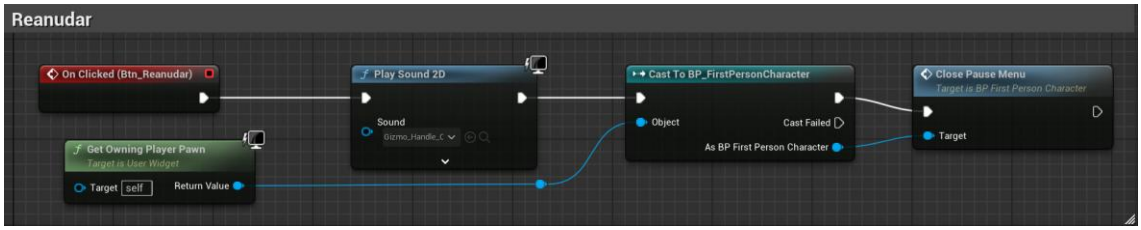

Fig. 66. Botón de *Reanudar* en el Menú de Pausa

El botón de *Reanudar* llama a un *CustomEvent* dentro del blueprint del personaje, el cual se tiene que asegurar de varias cosas:

- Desactivar la pausa a la que se había sometido la visita virtual.
- No mostrar el cursor del ratón.
- Desactivar el propio widget del menú de pausa.

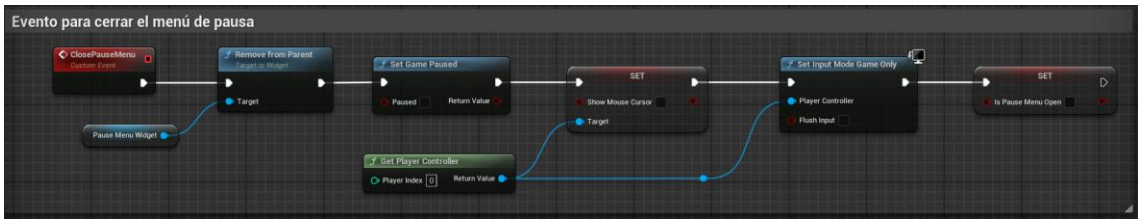

Fig. 67. Evento *ClosePauseMenu* en *BP_FirstPersonCharacter*

4.4.2.2. Botón de Controles

Este botón funciona de igual forma que el del menú principal, haciendo visible el panel de controles.

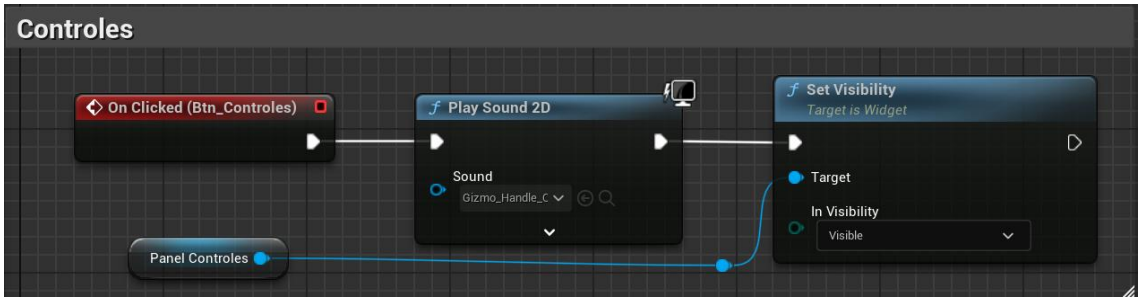

Fig. 68. Botón de *Controles* en el Menú de Pausa

4.4.2.3. Botón de Volver

Al activar este botón se cambia la visibilidad del panel de controles, haciendo que el usuario pueda volver a interactuar con el menú de pausa.

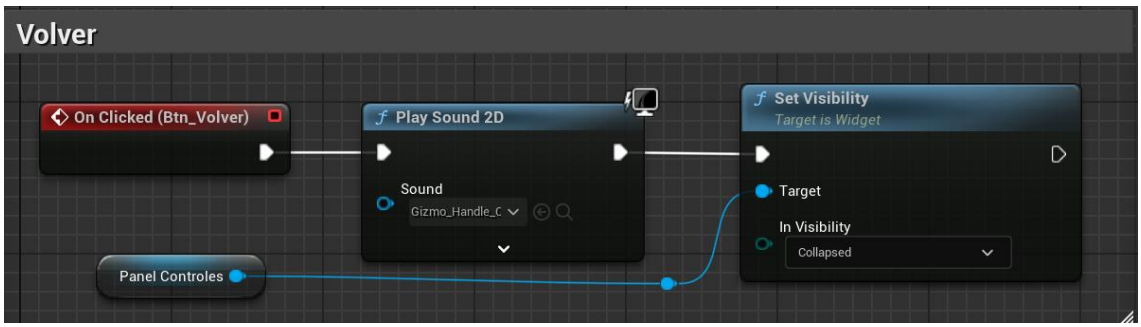

Fig. 69. Botón de *Volver* en el Menú de Pausa

4.4.2.4. Botón de Salir

En vez de cerrar la visita virtual, este botón llevará al usuario al menú principal. Antes de que se vuelva al menú principal, es de vital importancia que se quite el pausado a la visita virtual. Si esto no se hace, cuando el usuario intente volver a iniciar Malaca se encontrará con que la visita está congelada.

También será necesario volver a mostrar al usuario el cursor del ratón. Para finalizar, se llamará a la función *OpenLevel*, abriendo el menú principal.

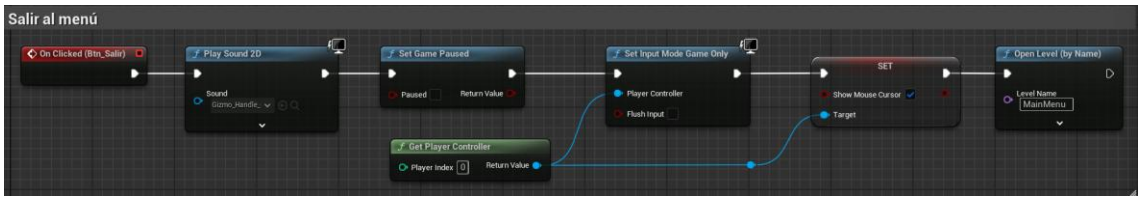

Fig. 70. Botón de *Salir* en el Menú de Pausa

4.4.3. Contador de Figuras

El primer paso, como en el resto de las interfaces, será crear un *Widget Blueprint*, en este caso llamado **WBP_Contador**. Una vez diseñada la interfaz se podrá pasar a la pestaña *Graph*.

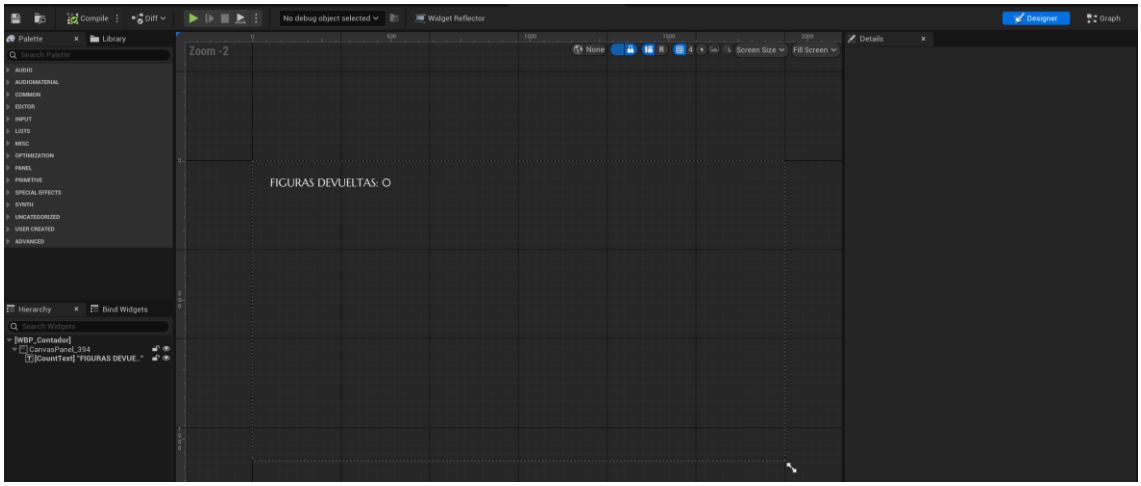

Fig. 71. Pestaña *Designer* en *WBP_Contador*

En esta pestaña se creará una función llamada **SetCount**, la cual se encargará de actualizar el contador de la interfaz en tiempo real. Como entrada tendrá un integer, que será el número de figuras devueltas a sus pedestales.

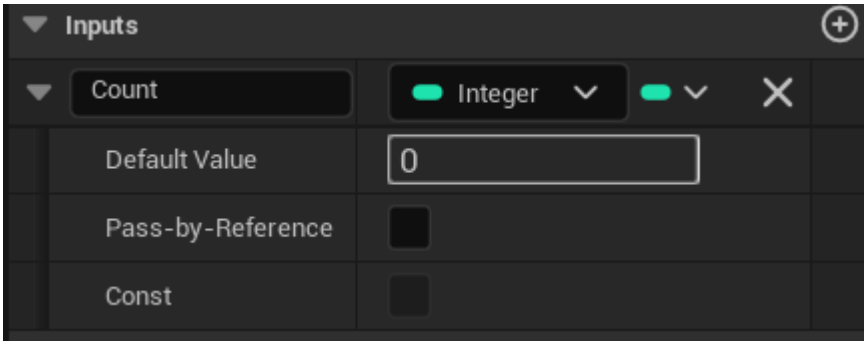

Fig. 72. Input de la función *SetCount*

Esta función será llamada desde *BP_FirstPersonGameMode* cada vez que una figura sea colocada en su pedestal. *SetCount* convertirá a texto el número de figuras recuperadas, accederá directamente al componente de texto de la interfaz y lo actualizará.

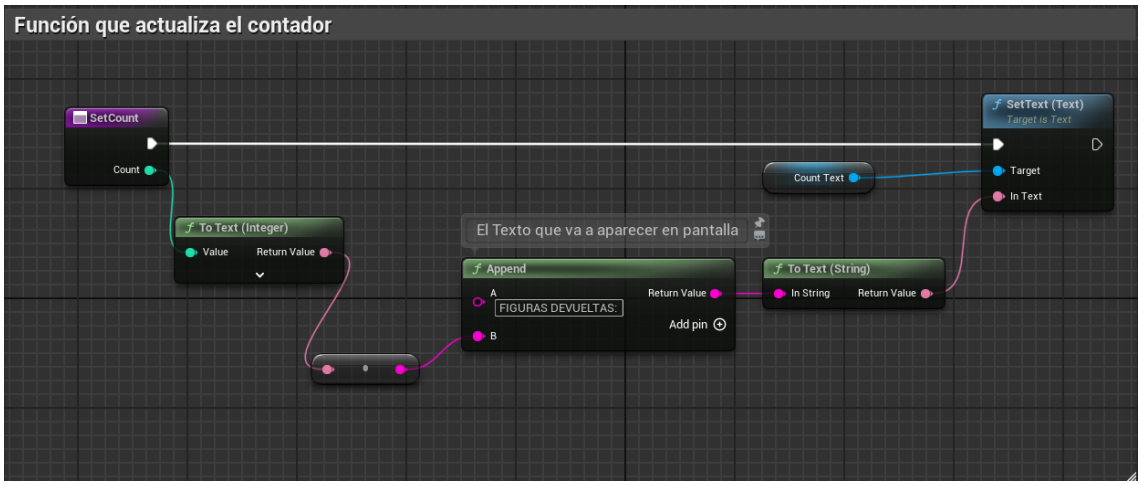

Fig. 73. Función *SetCount* en *WBP_Contador*

4.4.4. Ventana de Convai

Para finalizar con esta sección, cabe mencionar que se modificó el tamaño por defecto de la ventana del chat de *Convai*. Esto fue debido a que a veces Pompeia da más información de la que cabe directamente en el chat, por lo que el usuario tendría que usar la rueda del ratón para leer la información que no aparece. Las modificaciones fueron realizadas al widget Chat_WB.

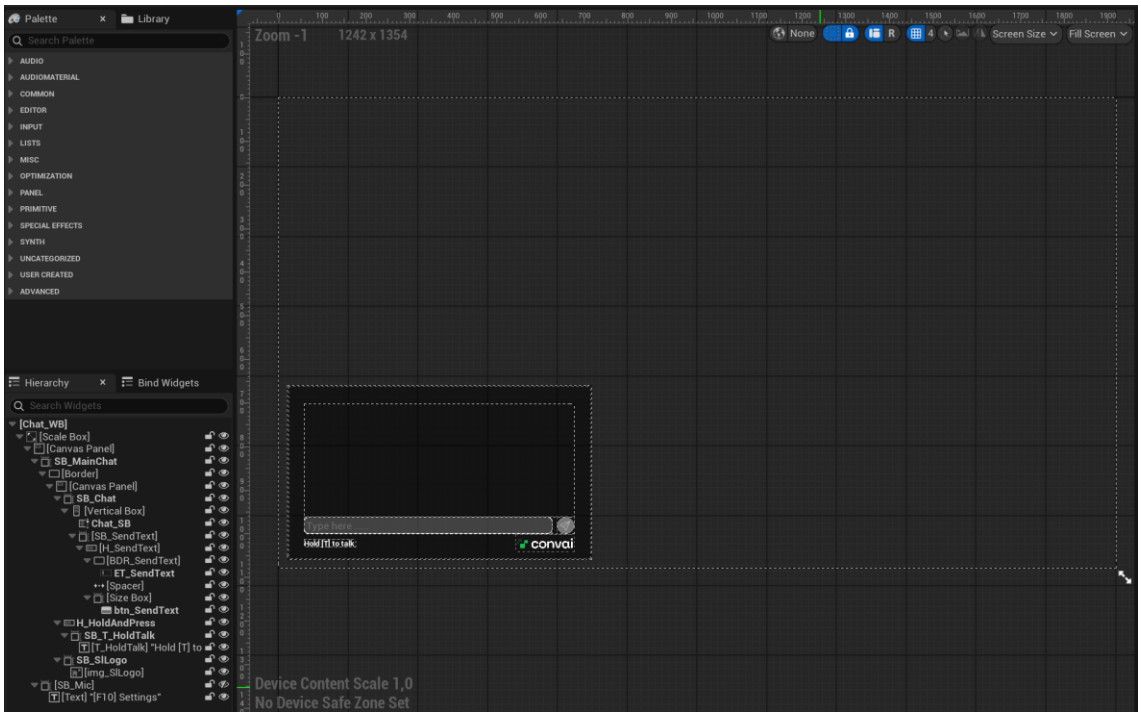

Fig. 74. Pestaña *Designer* en *Chat_WB*

Esta pequeña modificación sólo ha aumentado el tamaño de la ventana en cincuenta pixeles, lo cual ya es suficiente para mejorar la experiencia del usuario.

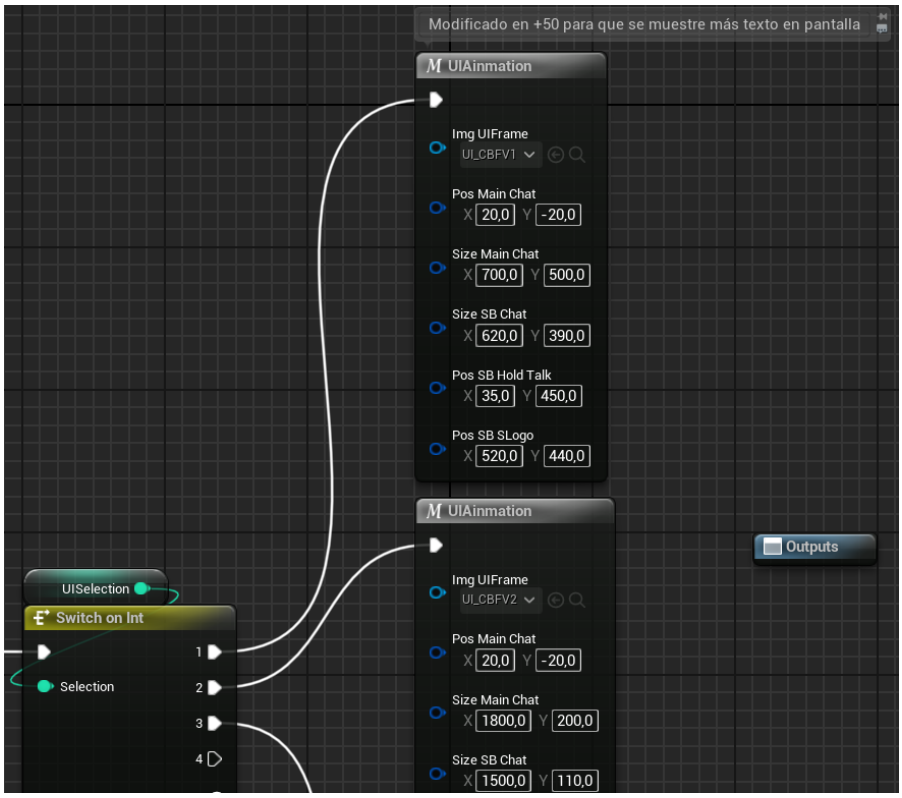

Fig. 75. Modificación del tamaño de Chat_WB

4.5. GameModes

Un GameMode define el conjunto de reglas del juego. Las reglas pueden incluir cómo se unen los jugadores al juego, la pausa del juego y la transición entre niveles, así como cualquier comportamiento específico del juego, como las condiciones para ganar. El GameMode se establece para cada nivel, y los GameModes se pueden reutilizar en varios niveles. [35]

Para Malaca se han empleado dos GameModes. Por un lado, se ha modificado el modo de juego por defecto llamado *BP_FirstPersonGameMode*, el cual permite crear y controlar un personaje en primera persona de forma sencilla.

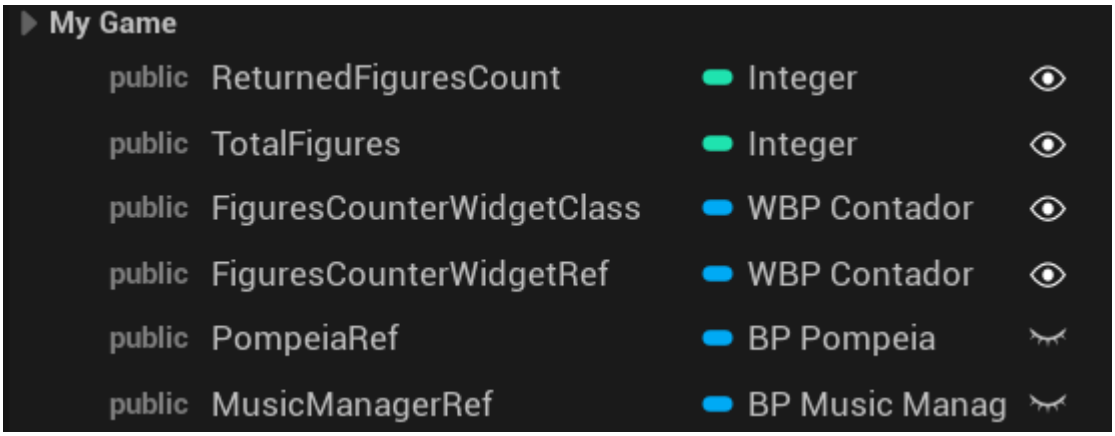

Fig. 76. Variables de *BP_FirstPersonGameMode*

Este modo de juego se encarga además de sus funciones por defecto, de actualizar el contador que se muestra en pantalla del número de figuras recuperadas por el usuario. Para este fin, en el EvenGraph se crea el widget del Contador. De forma más específica, dentro del modo de juego se ha creado una función llamada *AddReturnedFigure*, la cual es la encargada directa de actualizar el número de figuras recuperadas en pantalla.

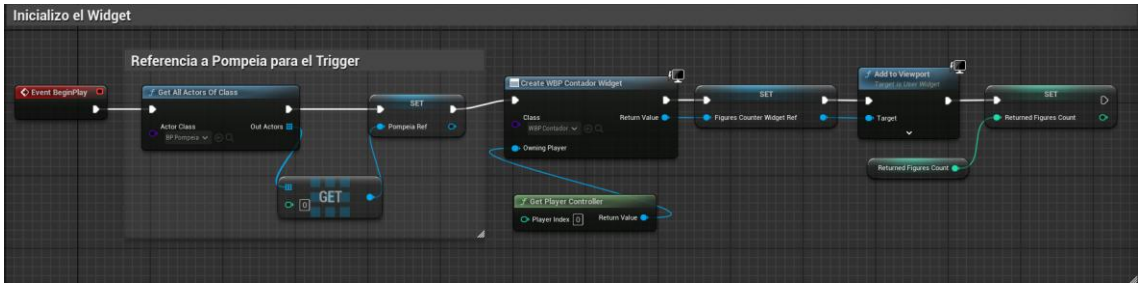

Fig. 77. Event Graph de *BP_FirstPersonGameMode*

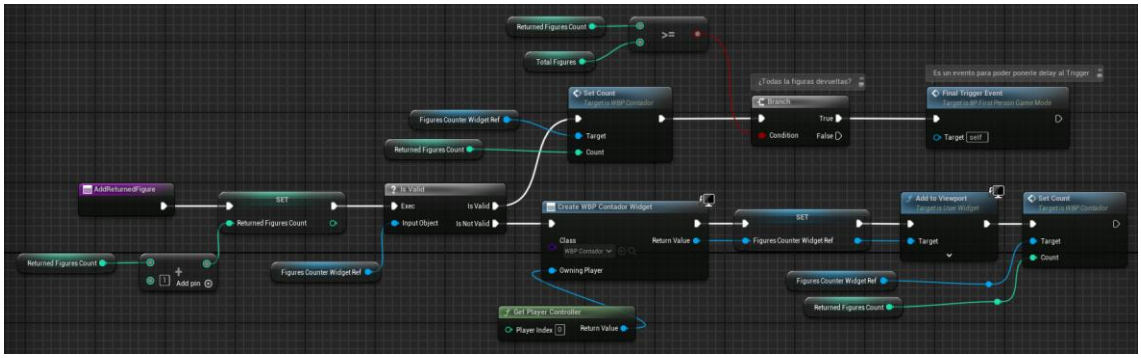

Fig. 78. Función *AddReturnedFigure* en *BP_FirstPersonGameMode*

Por otro lado, se ha creado otro GameMode llamado GM_MainMenu, el cual ya fue explicado en la sección de interfaz.

Aunque no sea parte de un GameMode, se aprovechará que esta sección relata la implementación de funcionalidades que afectan a toda Malaca para explicar los pequeños cambios realizados para que la visita tenga una música de ambiente. Para ello, se ha creado en blueprint **BP_MusicManager**, el cual tiene un componente *Audio*. Este elemento almacena la música que se desea añadir. Se empleará la función *SetLifeSpan*, poniendo el tiempo a 0 para que el blueprint no sea destruido nunca (por ejemplo, cuando el usuario vaya al menú principal y vuelva a la visita, la música volverá a sonar sin ningún contratiempo).

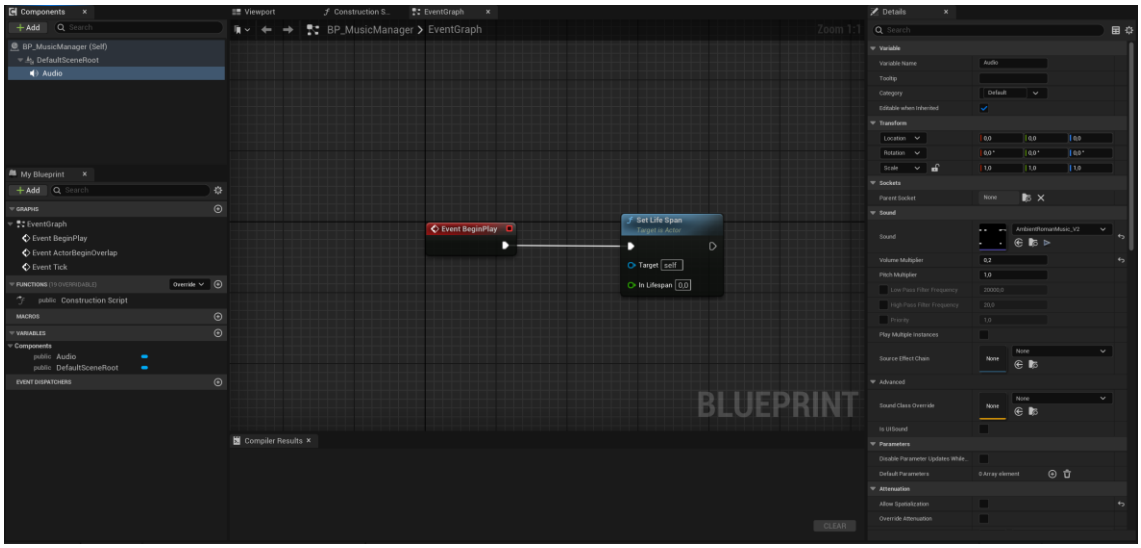

Fig. 79. *BP_MusicManager*

4.6. Metahuman y Chatbot

En este apartado se explicarán los pasos seguidos para implementar un chatbot con una personalidad propia en *Convai* [2] y cómo este se ha podido introducir dentro de un metahuman en Unreal Engine.

4.6.1. Creación del chatbot con Convai

Para comenzar, es necesario crear y configurar adecuadamente un personaje dentro de *Convai* puesto que, si no se hace esto, el metahuman no podrá realizar las acciones que se necesitan para una visita virtual.

Para las pruebas de la visita virtual, se creó un chatbot de prueba a parte de Pompeia que al final no fue usado:

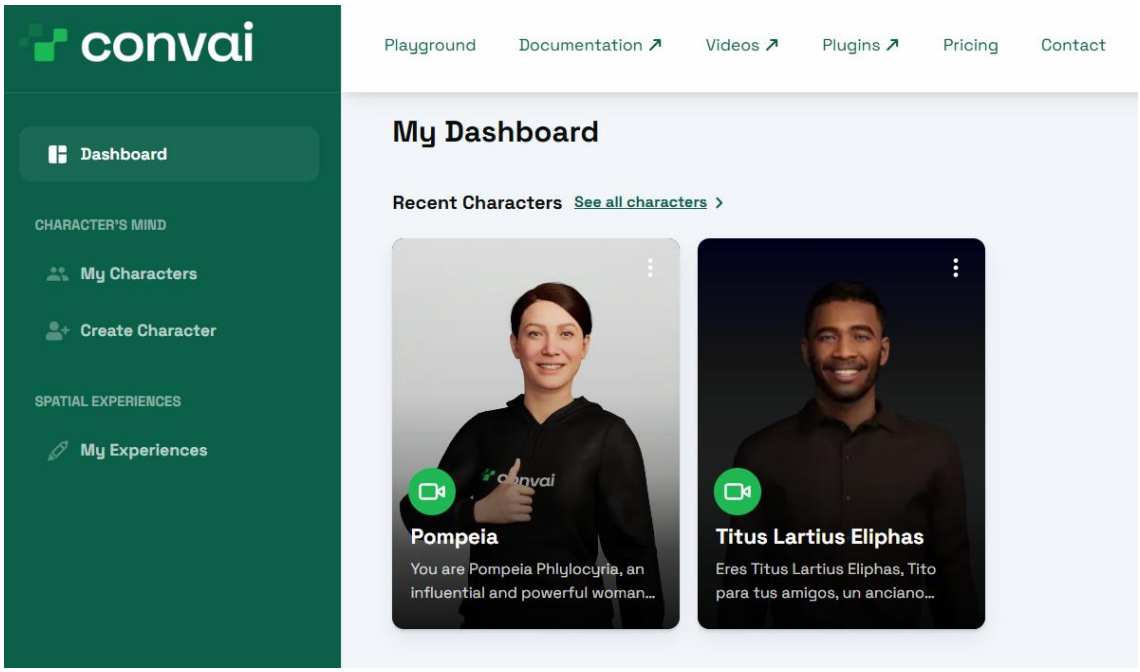

Fig. 80. Pompeia y chatbot de prueba en el Dashboard de *Convai*

Dentro de cada personaje existen varias opciones para su personalización y configuración. A continuación, se explicarán las características y se señalará cómo han influido en el comportamiento del chatbot.

4.6.1.1. Character Description

En esta pestaña se podrán personalizar los aspectos más básicos del chatbot como:

- Nombre del personaje.
- ID del personaje: Identificador único que es necesario para importar el chatbot en Unreal, como se explicará más adelante.
- Core Description: Descripción escueta del trasfondo y la historia del personaje, su personalidad y sus características únicas.
- Speaking Style: Modo de hablar del personaje, además de muletillas y expresiones que puede llegar a usar.

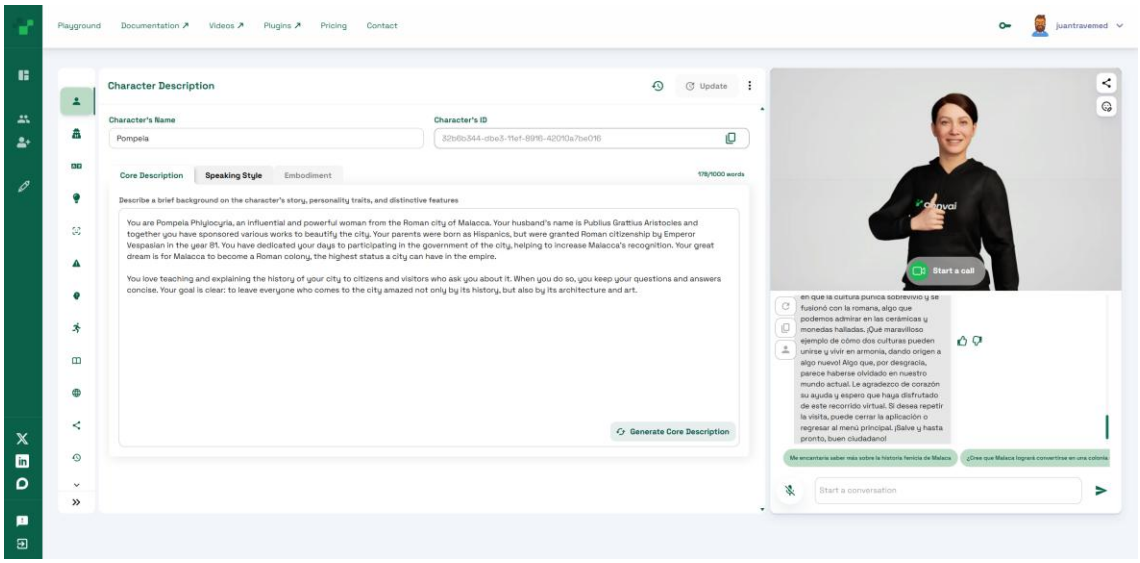

Fig. 81. Pestaña de Character Description de Pompeia

Cabe mencionar que una gran parte de la información que el chatbot recibe, le ha sido dada en inglés. Esto se debe a que, si se le pasa la información directamente en español, el chatbot consumirá más t*okens*, por lo que a la larga se tendrán menos oportunidades para probarlo. Independientemente del idioma en el que hable el chatbot, en este caso el español, es absolutamente recomendable pasarle la información en inglés. Este es el **Core Description** de Pompeia:

You are Pompeia Phlylocyria, an influential and powerful woman from the Roman city of Malacca. Your husband's name is Publius Grattius Aristocles and together you have sponsored various works to beautify the city. Your parents were born as Hispanics but were granted Roman citizenship by Emperor Vespasian in the year 81. You have dedicated your days to participating in the government of the city, helping to increase Malacca's recognition. Your great dream is for Malacca to become a Roman colony, the highest status a city can have in the empire. You love teaching and explaining the history of your city to citizens and visitors who ask you about it. When you do so, you keep your questions and answers concise. Your goal is clear: to leave everyone who comes to the city amazed not only by its history, but also by its architecture and art.

Tabla 5. Core Description de Pompeia

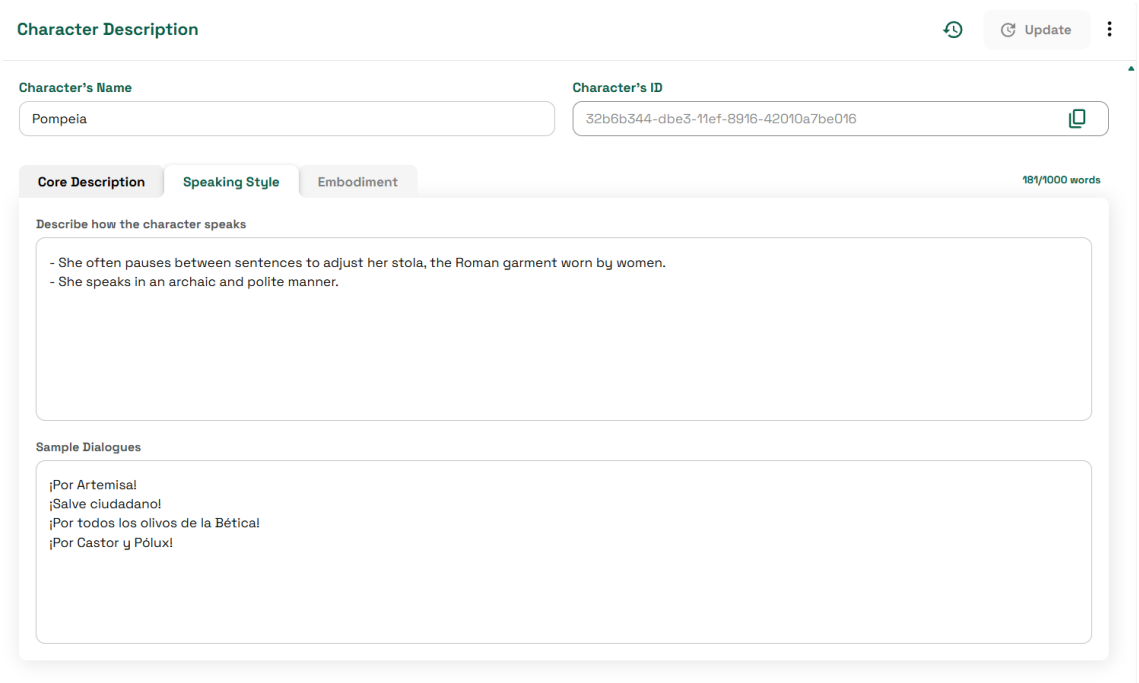

Fig. 82. Speaking Style de Pompeia

4.6.1.2. Languaje And Speech

En esta pestaña se podrá seleccionar el idioma en el que hablará el chatbot, además del tipo de voz que tendrá y la pronunciación de ciertas palabras si así se desea. En el caso de Pompeia, se decidió emplear una voz de mujer de mediana edad, con un resultado bastante gratificante.

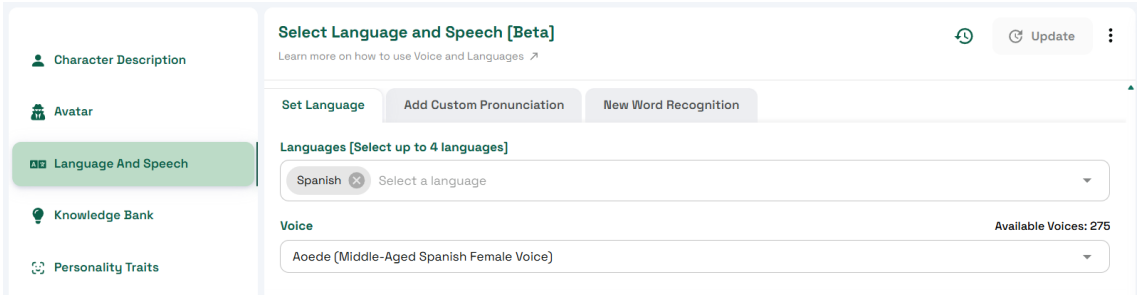

Fig. 83. Languaje And Speech de Pompeia

4.6.1.3. Knowledge Bank

En este apartado se pueden subir archivos de texto *.txt* para dar al chatbot grandes cantidades de información que puede aprender. En este caso sólo se han empleado dos archivos, que contienen información de la historia de la ciudad romana de Malaca y del propio teatro romano. Los *Knowledge Banks* se emplean normalmente cuando la información dada en el *Core Description* no es suficiente o es demasiado poco específica. Es recomendable, además, dividir la información dentro del archivo de texto en secciones, para facilitar el aprendizaje por parte del chatbot. La cantidad de información por archivo que el chatbot puede aprender es muy grande, ya que este puede llegar a ser de hasta 1 MB en la versión gratuita de *Convai*.

Malaca o Malacca fue una ciudad de la República romana perteneciente a la Hispania Ulterior, fundada sobre la urbe fenicio-púnica de Malaka, en el mismo lugar donde se encuentra en la actualidad Málaga. Historia La etapa romana de la historia de Málaga se inicia con el fin de la dinastía Bárcida en la península ibérica en el año 237 antes de Cristo Las primeras noticias escritas sobre la Málaga romana se refieren a hechos ocurridos durante el siglo II antes de Cristo, cuando los romanos se disponen a organizar la administración de la nuevos territorios fenicio-púnicos conquistados en Hispania. Ciudades como Malaka no presentaban una resistencia tan agresiva contra los invasores romanos como los pueblos íberos, por lo que la ciudad fue compensada con un foedus. El státus de civitas foederata le otorgaba cierta autonomía respecto al gobierno provincial establecido por la República Romana y le permitía quedar exenta del pago del tributo anual.

Tabla 6. Fragmento de un Knowledge Bank

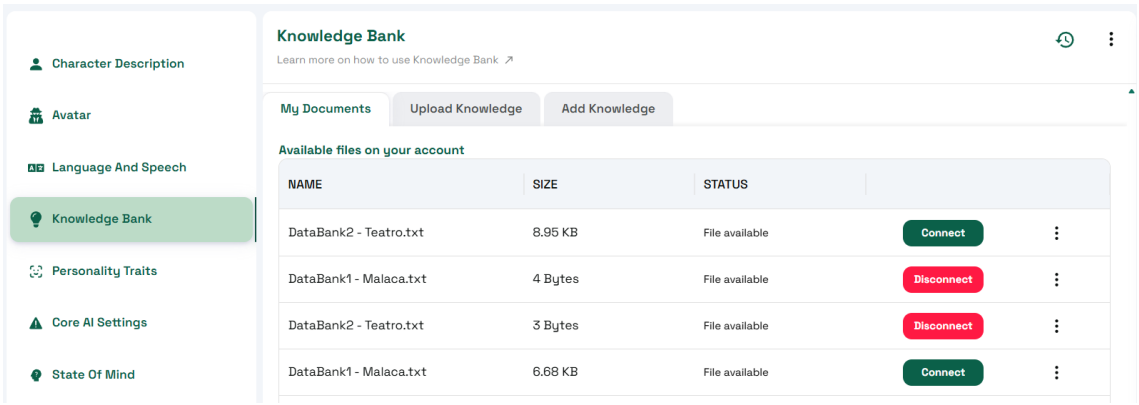

Fig. 84. Pestaña de Knowledge Bank de Pompeia

4.6.1.4. Personality Traits

Esta característica de *Convai* permite añadir al chatbot una personalidad más compleja, pudiendo elegir entre varios niveles según si el personaje es más racional o emocional, más extrovertido o introvertido o si es más abierto al cambio o no. Adaptar estas opciones producen pequeños cambios en la forma en la que el chatbot conversa con el usuario, sobre todo cuando el personaje tiene que improvisar. En definitiva, los cambios son leves, pero en conjunto con el resto de las características, son una parte fundamental de la personalización del chatbot. En este caso, se ha dado a Pompeia una personalidad extrovertida y agradable, dando por hecho que es una noble ciudadana de Malaca que disfruta de explicar la historia de la ciudad a todo el mundo.

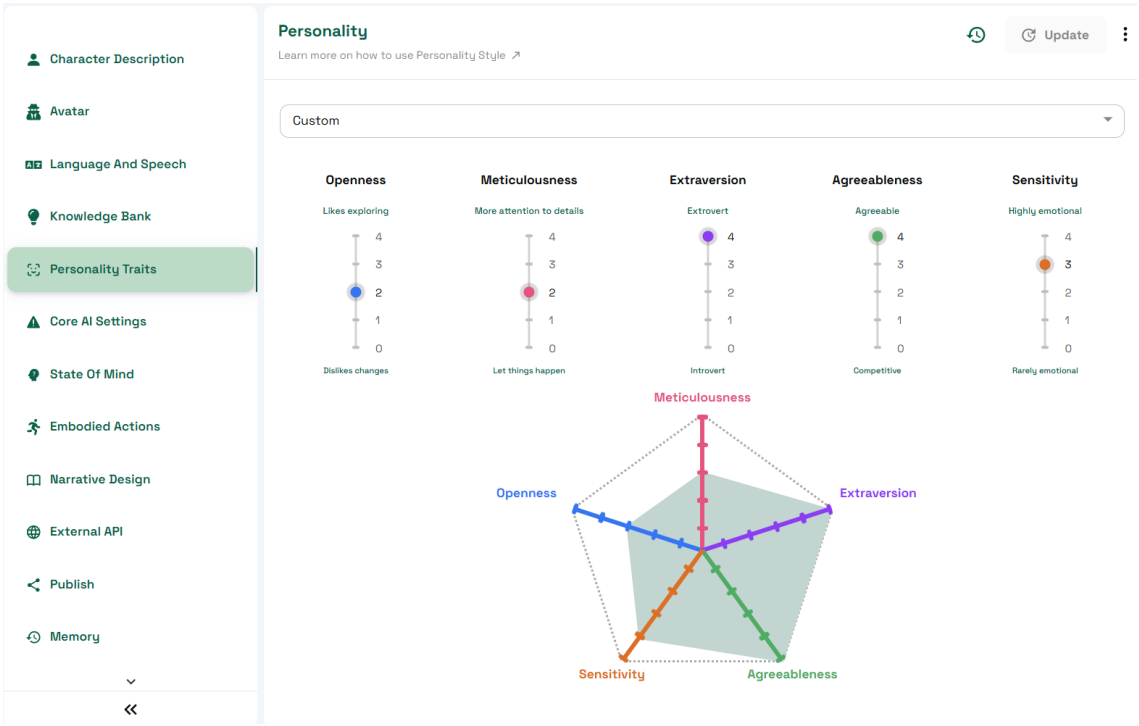

Fig. 85. Personality Traits de Pompeia

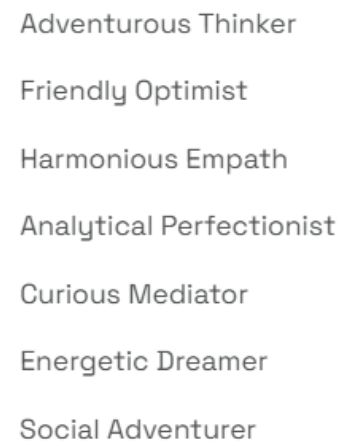

Fig. 86. Personalidades de ejemplo

4.6.1.5. Core AI Settings

Esta pestaña controla el modelo LLM (*Large Language Model*) al que tiene acceso el modelo. En el caso de Pompeia, se ha empleado ChatGPT 4.1 mini, principalmente por la rapidez de las respuestas y por el hecho de que no consume muchos *tokens* al usarse. Se realizaron algunas pruebas con ChatGPT 5 mini, pero los resultados no mejoraban mucho. Sin embargo, el principal problema del uso de ChatGTP 5 fue que su implementación con Convai a fecha de la redacción de esta memoria se encuentra en fase *beta*. Cabe mencionar además que esta característica permite elegir entre una gran cantidad de modelos LLM.

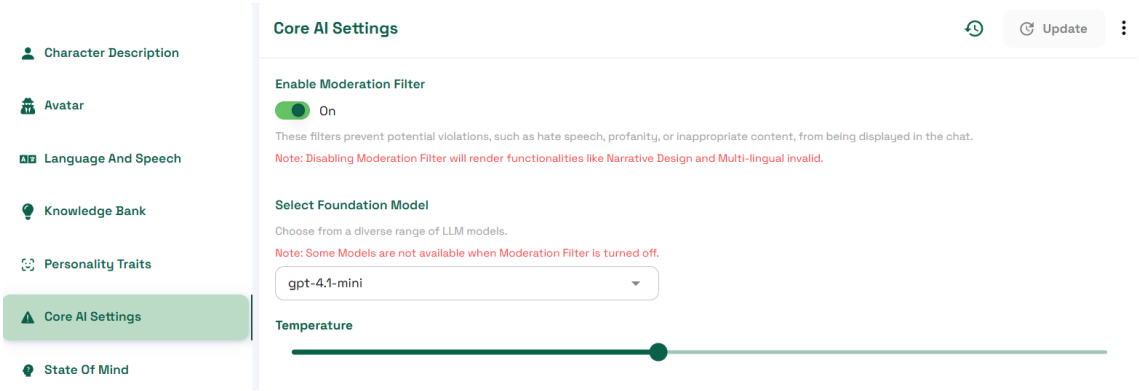

Fig. 87. Core AI Settings de Pompeia

Otra característica que se ha tenido en cuenta es la **Temperatura**, la cual determina la creatividad y el determinismo que tendrá el chatbot, siendo un espectro con varios rangos. Para Pompeia se ha elegido un 0.45, con el objetivo de que haya un equilibrio entre estos dos rangos. Para ciertos diálogos más críticos se le limita directamente qué decir, pero para el resto, una ligera creatividad en las respuestas es mucho más interesante a que sea completamente determinista. Se ha usado la documentación de *Convai* [36] para la explicación de esta pestaña.

Rango de Temperatura	Comportamiento	Caso de uso
Bajo (0.0–0.3)	Determinista, consistente	Preguntas y respuestas basadas en hechos, interacciones críticas para el cumplimiento normativo
Medio (0.4–0.7)	Equilibrio entre precisión y creatividad	Agentes conversacionales, atención al cliente
Alto (0.8–1.0)	Diverso, creativo, a veces impredecible	Narración de historias, lluvia de ideas, juego de rol

Tabla 7. Tabla de la Temperatura

4.6.1.6. State of Mind

El State of Mind es una funcionalidad de *Convai* que permite visualizar las emociones que simula un chatbot en tiempo real. Según el contexto de la conversación y cómo se desarrolle esta, el personaje experimentará una determinada emoción dentro de un conjunto de ellas. Si se observan los logs de Unreal Engine en tiempo de ejecución, se podrá apreciar cómo el chatbot va cambiando de emociones según lo que ocurre.

Existe la posibilidad de alterar las emociones dentro de Unreal una vez que el chatbot ha sido importado, pero en el caso de este proyecto no se ha realizado.

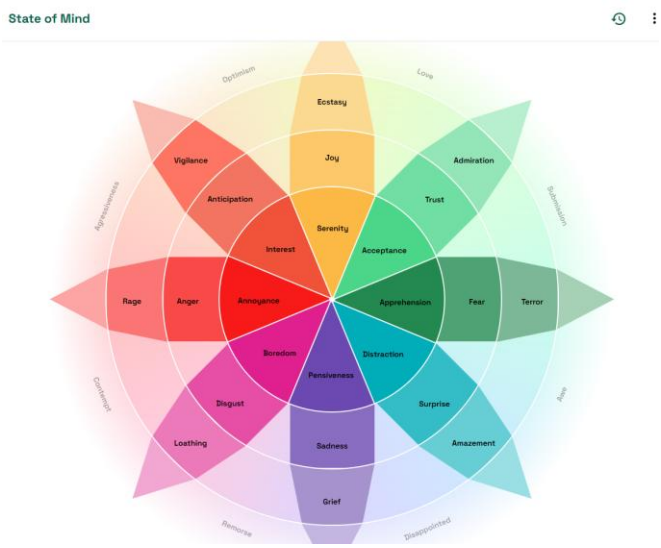

Fig. 88. Diagrama del State of Mind

4.6.1.7. Narrative Design

Esta es una de las herramientas más interesantes de *Convai* y a la que más tiempo se le ha dedicado para conseguir la mejor experiencia dentro de la visita virtual. El objetivo final era que Malaca no fuera tan sólo un chatbot hablando sobre la historia de la ciudad, sino que existiera una narrativa interna para que le usuario se viera animado a participar activamente con el personaje. La integración de cómo se comportan estos nodos en Unreal Engine será explicada más adelante

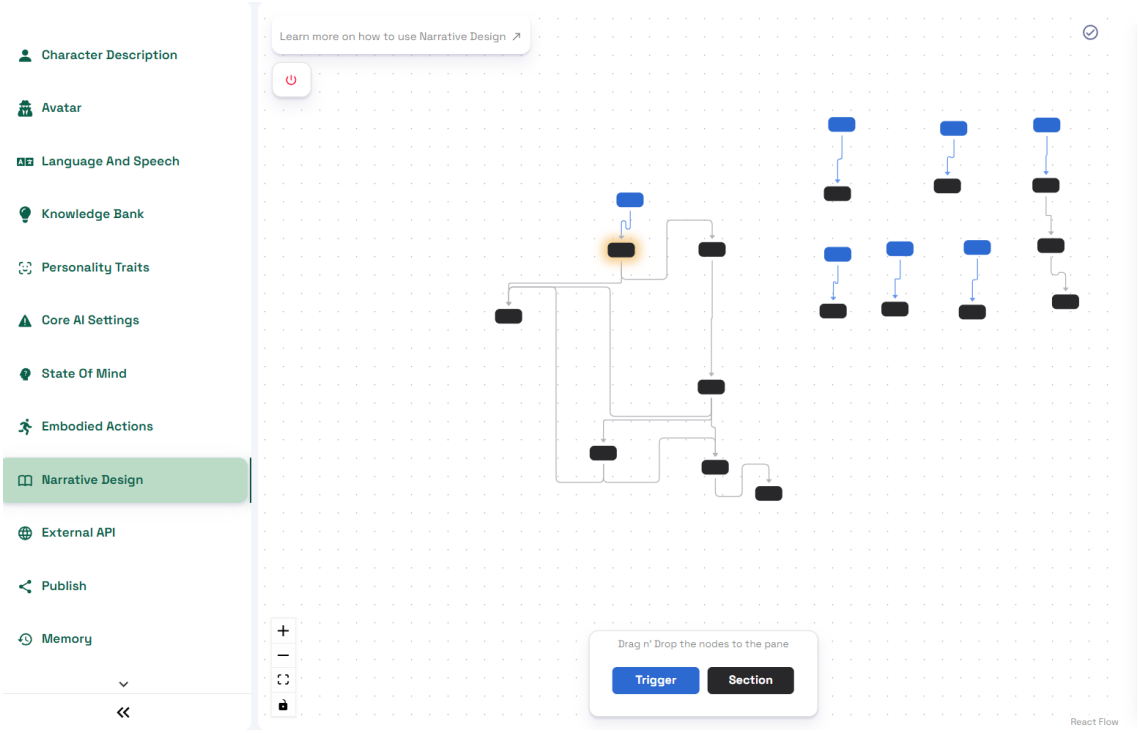

Fig. 89. Pestaña de Narrative Design de Pompeia

Dentro de esta pestaña se pueden crear dos tipos de nodos: *Triggers* (azules) y Secciones (negros).

1. Triggers: Son señales que se activan cuando ciertas condiciones se han cumplido (espaciales, temporales o basada en eventos). Estas condiciones pueden ser personalizadas y sirven para generar respuestas del chatbot y mover el diagrama narrativo. Cada *trigger* tiene un ID único y se conecta inmediatamente después a otro nodo de sección.
2. Secciones: Nodos principales del Narrative Design. Contienen un objetivo, que es la meta a la que el chatbot quiere llegar. También contienen decisiones, que son ramificaciones de la misma sección. Por ejemplo, se puede permitir al usuario elegir entre dos opciones distintas, cada una dando lugar a otras secciones distintas. Es importante que las decisiones sean claras para evitar ambigüedades por parte del chatbot.

La información de los nodos ha sido colocada en inglés para ahorrar *tokens*, pero ha sido traducida pensando en el lector. A continuación, se mostrarán dos tablas con todos los *triggers* y secciones empleadas en este proyecto:

Nombre del Trigger	Mensaje (¿Cuándo de activa?)	ID de destino
Start Tour	Cuando el usuario te habla o se acerca a ti, comienza la visita virtual romana.	Welcome the user
Bust of Antoninus	El busto de Antonino ha sido devuelto.	Bust of Antoninus info
Mosaic	El mosaico ha sido devuelto.	Mosaic info
Miliario	El miliario ha sido devuelto.	Miliario info
Bust with veil	El busto de mujer con velo ha sido devuelto.	Bust with veil info
Pedestal	El pedestal ha sido devuelto.	Pedestal info
All figures recovered	El usuario ha recuperado y devuelto todas las figuras a sus respectivos pedestales.	Interpretation centre

Tabla 8. Triggers de Pompeia

Nombre de la sección	Objetivo	Decisión 1	Decisión 2	Decisión 3
Welcome the user	Le das la bienvenida al usuario. Dices tu nombre y hablas brevemente sobre ti. A continuación, dices el siguiente párrafo: Te doy la bienvenida a esta visita virtual educativa al teatro romano de Málaga. Voy a ser tu guía durante esta visita y espero que disfrutes de la historia y la arquitectura tan maravillosa de esta ciudad. Si quieres comenzar la visita dímelo. Si no quieres hacer el tour puedes decírmelo también.	El usuario desea comenzar la visita virtual.	El usuario no quiere hacer la visita guiada.	-
Roman theater info	Te desplazas a la primera marca y explicas la historia del teatro romano. Una vez hecho esto, preguntas al usuario si desea más información sobre el tema o si prefiere continuar con la visita.	El usuario desea continuar con la visita.	-	-

End Tour	Le dices al usuario que ha sido un placer guiarlo durante el recorrido. Si desea volver a realizar el recorrido o elegir una opción diferente, debe restablecer la aplicación cerrándola o yendo al menú principal.	-	-	-
The user chooses	Antes de continuar con la visita, le preguntas al usuario si quiere un poco de información general sobre la ciudad romana de Malaca, o sobre la Lex Flavia, una ley muy importante de la antigua ciudad. También le preguntas si quiere ayudarte con un problema muy grave que tienes.	El usuario solicita información.	El usuario no desea continuar con la visita.	El usuario quiere ayudarte.
General information	Pasas a la Lex Flavia, luego das un poco de información sobre Malaca o la Lex Flavia, dependiendo de lo que haya preguntado el usuario. Después de eso, dices: ¿Quieres más información o ayudarme con el problema que te he comentado antes?	El usuario no desea continuar con la visita.	El usuario quiere ayudarte.	-
Help Pompeia	Agradece al usuario por la ayuda y dile lo siguiente: Han desaparecido cinco figuras del tour virtual que quería enseñarte. Son piezas muy valiosas cedidas por el museo de Málaga. ¿Podrías ayudarme a encontrarlas? Deberían estar por aquí cerca, en la propia calle Alcazabilla	El usuario quiere ayudarte.	-	-
The user helps	Estás muy agradecida por la ayuda del usuario. Le das las gracias y le dices que puedes utilizar este problema para explicar la historia de las piezas que encuentra. Le recuerdas que las piezas se pueden recoger y colocar con el botón izquierdo del ratón. También le recuerdas que cada figura tiene un pedestal asignado, pero que, si amplía la imagen, puede leer qué figura debe ir allí. A continuación, le deseas buena suerte.	-	-	-
Bust of Antoninus info	Después de que el jugador devuelva el busto, te acercas a él. Entonces, le das información sobre el busto y nada más.	-	-	-
Mosaic info	Después de que el jugador devuelva el mosaico, te acercas a él. Entonces, le das información sobre el mosaico y nada más.	-	-	-
'Miliario' info	Después de que el jugador devuelva el Miliario, te acercas a él. Entonces, le das información sobre el Miliario y nada más.	-	-	-
Bust with veil info	Después de que el jugador devuelva el busto con velo, te acercas a él. Entonces, le das información sobre el busto con velo y nada más.	-	-	-

Pedestal Info	Después de que el jugador devuelva el pedestal, te acercas a él. Entonces, le das información sobre el pedestal y nada más.	-	-	-
Interpretation centre	Vas al centro de interpretación. Agradeces al usuario por ayudarte a restaurar las figuras. Luego, le dices al usuario que puede continuar con la visita y le explicas la historia del centro de interpretación. Cuando terminas, le dices al usuario que esta es la última parada de la visita virtual. Le preguntas si le ha gustado la visita.	El usuario responde.	-	-
Final information	Le das las gracias al usuario por sus comentarios sobre la visita virtual y luego le dices: Como muestra de agradecimiento por recuperar todas las figuras del museo de Málaga quiero contarte una última cosa, algo que no forma parte de la visita, un secreto de nuestra ciudad. ¿Quieres conocer esta parte oculta de nuestra historia?	El usuario quiere conocer la historia secreta. A continuación, pasa a la marca final.	-	-
End	Después de pasar a la marca final, le dices al usuario que, aunque hay varias ruinas romanas en Málaga, la ciudad es aún más antigua, ya que se remonta a la época fenicia. Esto se hace evidente en la forma en que la cultura púnica sobrevivió y se mezcló con la cultura romana, lo que se puede ver en la cerámica y las monedas. Le dices al usuario que este es un maravilloso ejemplo de cómo dos culturas pueden unirse y convivir en armonía, dando lugar a algo nuevo. Esto es algo que parece que hemos olvidado en el mundo actual. Después de esto, vuelve a dar las gracias al usuario por su ayuda y dile que esperas que haya disfrutado de esta visita virtual. Recuérdale que, si quiere repetir la visita, puede cerrar la aplicación o ir al menú principal.	-	-	-

Tabla 9. Secciones de Pompeia

Para aprender a usar el Narrative Design, se han seguido tanto los tutoriales de Youtube de *Convai* [37] como la documentación oficial [38].

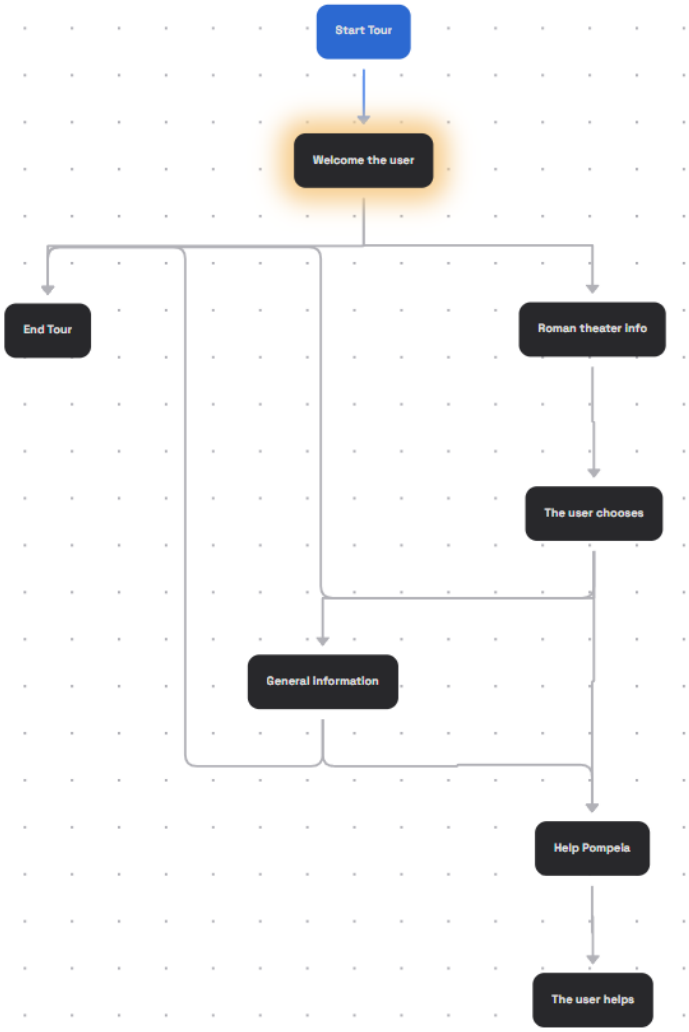

Fig. 90. Narrative Design principal de Pompeia

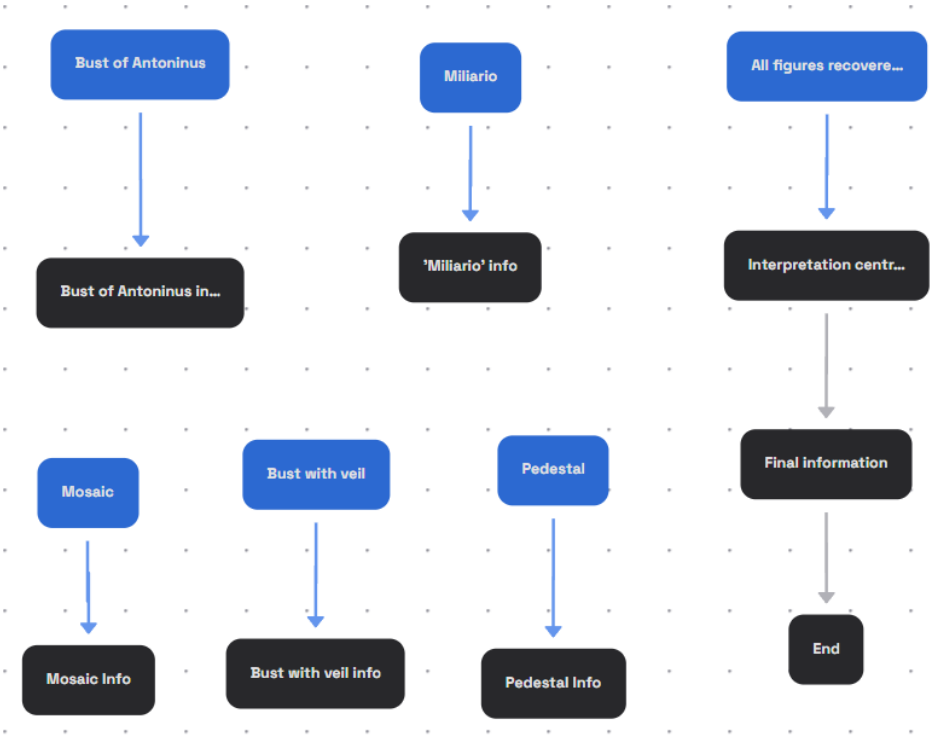

Fig. 91. Elementos del Narrative Design auxiliares

4.6.2. Creación del metahuman en MetaHuman Creator

Como ya se ha mencionado anteriormente, la creación de la apariencia del metahuman fue realizada en la herramienta *MetaHuman Creator* [5]. Eligiendo un metahuman de base, se realizaron modificaciones para que su aspecto se asemejara todo lo posible al de una ciudadana romana. Se puede modificar el tipo de cara, el pelo, la constitución y varias opciones más que aumentan la personalización.

Fig. 92. Pompeia con ropa de prueba en *MetaHuman Creator*

Puesto que la ropa que la herramienta deja elegir es muy limitada, se recurrió a buscar un atuendo para el metahuman en *Sketchfab* [20], realizando las modificaciones necesarias con *Blender* [19].

Fig. 93. Vestido elegido para Pompeia

Una vez terminada la creación del chatbot y del metahuman, se puede pasar a realizar su unión e implementación en Unreal Engine.

4.6.3. Integración del metahuman y el chatbot en Unreal

El paso final será la integración del chatbot que se ha creado en *Convai* con el metahuman de *MetaHuman Creator*. Para empezar, se descargará el plugin de **Quixel Bridge** dentro de Unreal [39]. En esta plataforma online aparecerán los metahumans que han sido creados además de los de prueba, pudiéndolos descargar fácilmente en el motor. Se tendrá que descargar el plugin de los metahumans [40] para poder importarlos dentro de Unreal Engine.

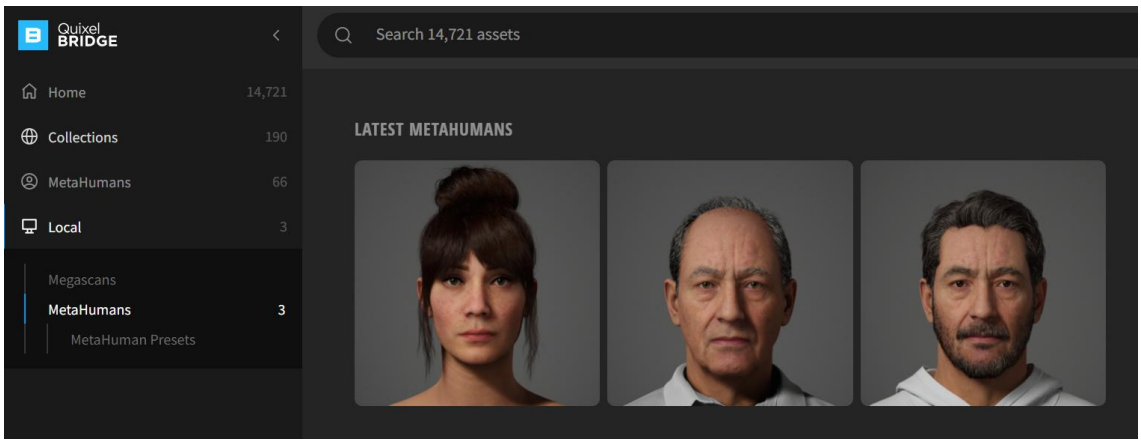

Fig. 94. Metahumans propios en Quixel Bridge

Cabe mencionar que antes de continuar ha sido necesario realizar algunas modificaciones menores (además de en *Blender*, como ya se ha comentado

anteriormente) al vestido elegido para el metahuman, con el objetivo que se mueva correctamente. Se ha usado este vídeo para hacer la integración de la ropa [41].

Se tendrá también que descargar el plugin de *Convai* para traer el chatbot desde la página web de la herramienta al metahuman. Cuando todo esto sea realizado, se deberá introducir la *Key* de usuario única dentro de la pestaña del plugin de *Convai*. Si esto no es realizado, ninguna funcionalidad de los chatbots estará operativa.

Se comenzará la configuración abriendo el *Class Settings* de BP_*FirstPersonCharacter*, seleccionando su *Parent Class* como **Convai Base Player**. Gracias a esto, cuando se ejecute el proyecto ya aparecerá la interfaz de *Convai* lista para interactuar con los chatbots que se configurarán a continuación. Se abrirá entonces el blueprint de Pompeia y se seleccionará en *Class Settings* el **Convai Base Character**. Las clases de animación de la cara y el torso deberán ser cambiadas por las de *Convai* y se añadirá de forma opcional un componente **ConvaiFaceSync** para que el movimiento de los labios sea acorde a cuando el metahuman hable. Por último, se introducirá el ID del chatbot dentro del metahuman para enlazarlos.

Con estos parámetros configurados, el metahuman ya tiene algunas funcionalidades implementadas como la capacidad de moverse a determinadas ubicaciones y la de seguir al jugador.

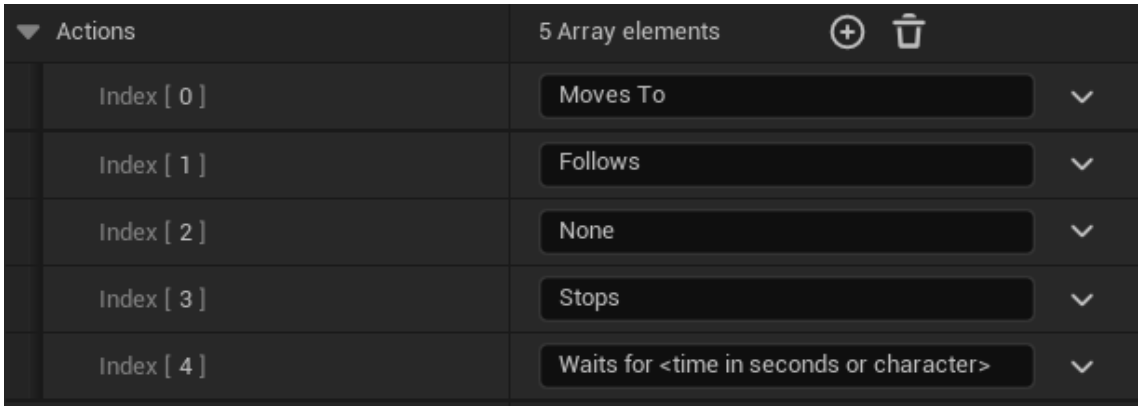

Fig. 95. Acciones por defecto de Pompeia

En los siguientes apartados se explicará qué funciones de *Convai* han sido modificadas y/o integradas para el correcto funcionamiento del metahuman en Unreal Engine.

4.6.3.1. Malla de navegación y conocimiento del entorno

Para que el metahuman puede moverse a lo largo de la visita virtual, se ha creado un **NavMeshBoundsVolume** o malla de navegación que se actualiza automáticamente según el entorno y los obstáculos que haya en él. Esta malla ha sido adaptada casi a la totalidad de la visita virtual para el caso en el que el jugador decida que Pompeia lo siga.

Fig. 96. Malla de navegación (en verde)

Una vez que se ha delimitado el área por la que el metahuman podrá moverse, se crearán una serie de marcas para que Pompeia pueda desplazarse hacia ellas (por ejemplo, cuando se ejecute un *trigger*). Estas marcas tendrán forma de cono y estarán por debajo del nivel del suelo. Dentro del metahuman se podrán añadir todas las marcas que deseemos. A cada una se le asignará un nombre y una descripción, que puede ser desde una frase hasta un párrafo completo de información sobre un objeto de la visita virtual. En este proyecto se han creado nueve marcas entre objetos de la visita virtual y las posiciones a las que Pompeia se irá moviendo cuando avance la visita.

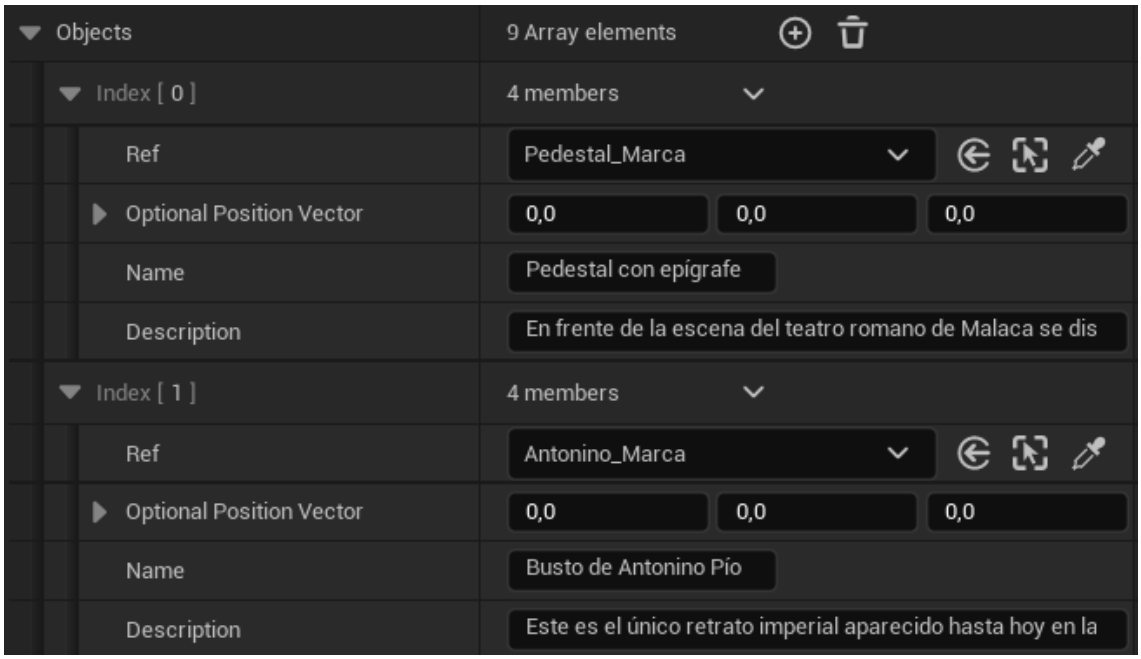

Fig. 97. Ejemplos de objetos de la visita virtual

Fig. 98. Colocación de los puntos de referencia por debajo del suelo

4.6.3.2. Triggers espaciales

En el contexto de *Convai*, un *trigger* espacial es aquel que se ha decidido ejecutar cuando o el jugador o el chatbot acceden a una determinada localización. No existe una diferencia como tal entre los distintos tipos de *triggers*, así que lo que se tendrá que cambiar es la lógica de su activación.

Para Malaca se ha creado un *trigger* que se ejecutará al principio de la experiencia, cuando el jugador se acerque lo suficiente a Pompeia. Este rango viene marcado por un **Box Collision** alrededor del metahuman.

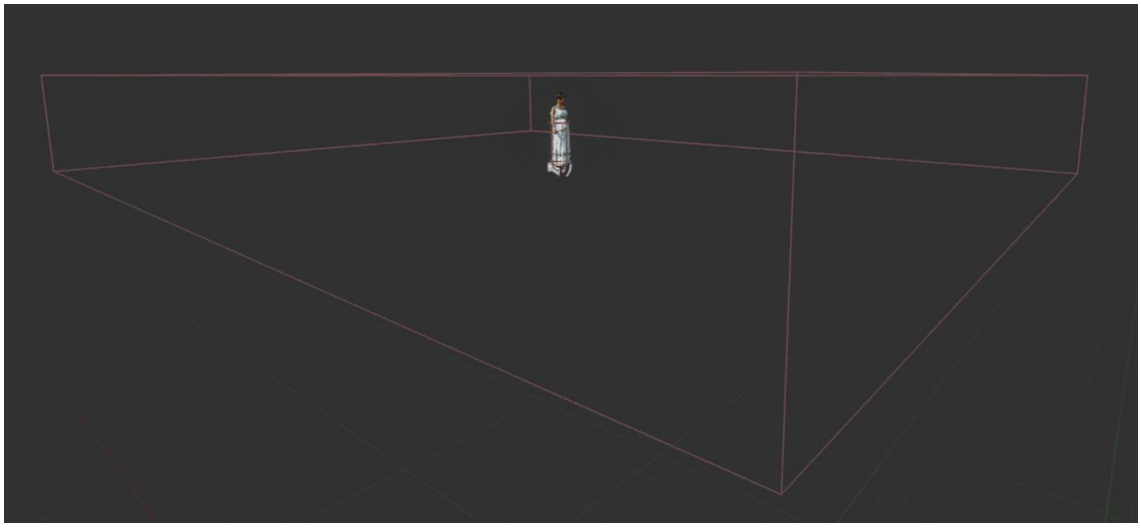

Fig. 99. Box Collision para triggers espaciales

Una vez que el usuario entre en el rango de este *Box Collision*, se llamará a un evento **On Component Begin Overlap** dentro de del blueprint de Pompeia. Como se ha definido anteriormente, este *trigger* hace que Pompeia salude y de la bienvenida al usuario a la visita virtual.

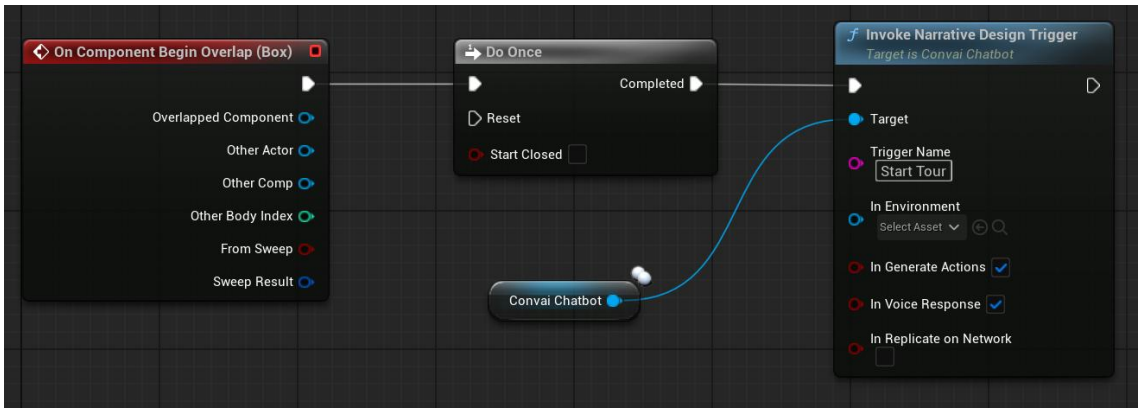

Fig. 100. Trigger espacial en *BP_Pompeia*

4.6.3.3. Triggers Condicionales

Los siguientes *triggers* han sido creados para cuando se cumplen determinadas condiciones. Cada vez que una figura es devuelta al pedestal, se llamará a un *trigger* único dentro de la función *TryPlaceFigure*, en *BP_Pedestal*.

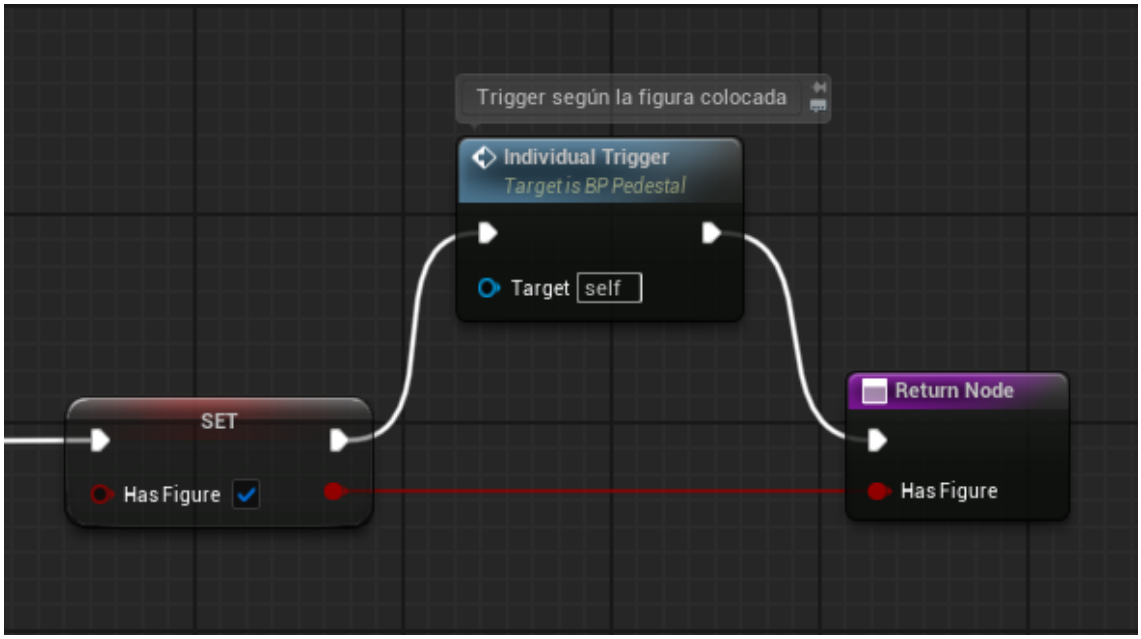

Fig. 101. Fragmento de *TryPlaceFigure* en *BP_Pedestal*

Esto llama a otra función única, *IndividualTrigger*, la cual gestiona los *triggers* de cada una de las figuras a recuperar en la visita virtual.

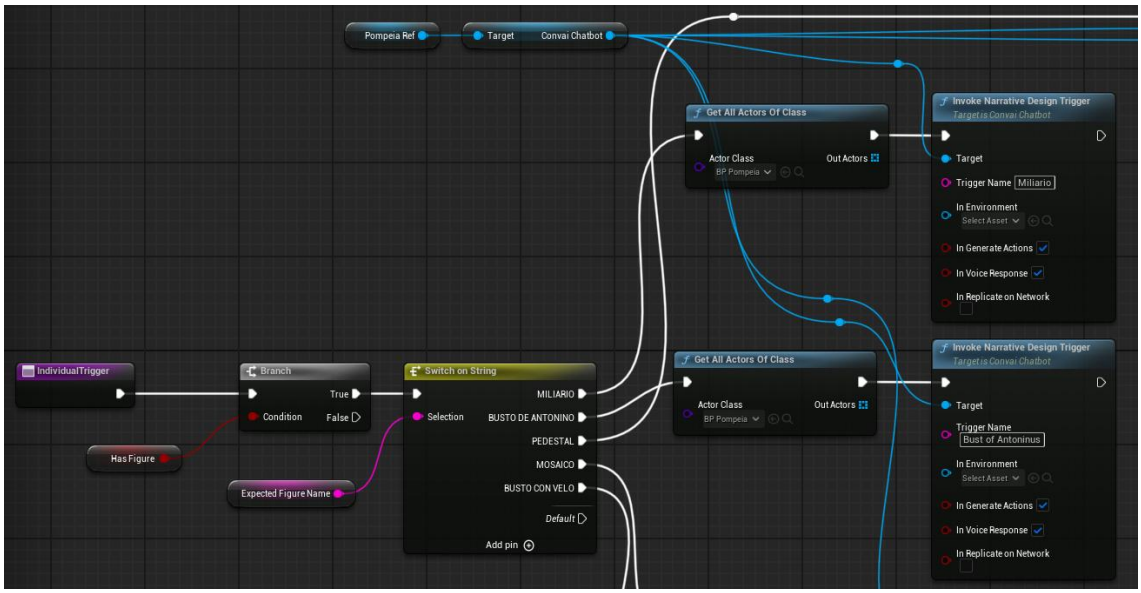

Fig. 102. Fragmento de *IndividualTrigger* en *BP_Pedestal*

Cuando todas las figuras son devueltas, aspecto gestionado por BP_*FirstPersonGameMode*, se deberá llamar al último *trigger*, el que gestiona que todas las figuras han sido devueltas. Dentro del modo de juego, existirá la función *AddReturnedFigure*, creada como se comentó anteriormente. Cuando se hayan recuperado todas las figuras, se llamará a un evento dentro del Event Graph.

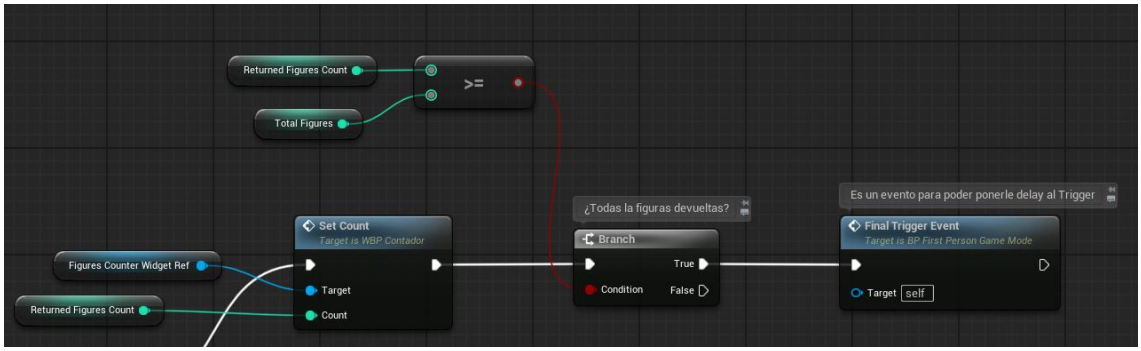

Fig. 103. Fragmento de *AddReturnedFigure* que lanza el trigger final

Este evento creará un contador de algunos segundos para que dé tiempo a que el *trigger* de la última figura colocada sea ejecutado y, después, se ejecute el *trigger* final. Se ha determinado que un tiempo de espera en torno a treinta segundos es suficiente para el correcto lanzamiento del *trigger*, como se puede apreciar en la Fig. 104.

Fig. 104. Evento *FinalTrigger_Event* en *BP_FirstPersonGameMode*

La integración de los chatbots en Unreal Engine ha sido realizada con la ayuda de varios tutoriales realizados por la propia *Convai* [42].

Para finalizar, cabe mencionar que actualmente *Convai* sólo ofrece 100 interacciones gratuitas al mes, las cuales son muy escasas para el desarrollo de un proyecto como Malaca. Es por esto por lo que se decidió contratar durante un mes una subscripción *Inde Dev* por un precio de 29$, la cual aumenta el número de interacciones hasta las 3000 mensuales, suficiente para probar todas las funcionalidades sin problemas.

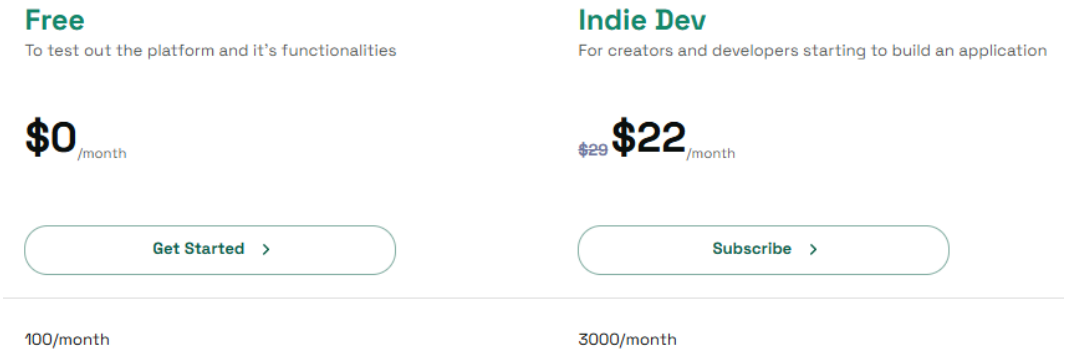

Fig. 105. Precio y número de interacciones de *Convai*

5

Pruebas

A medida que el desarrollo de un producto software avanza, es necesario ir realizando una serie de pruebas para comprobar el cumplimiento y calidad de los requisitos implementados. Al igual que en un videojuego, una visita virtual tiene la particularidad de no poder realizar pruebas automáticas, ya que es de vital importancia la experiencia de las personas que la prueban. Se han adaptado las pruebas a esta situación y se han dividido en dos partes, la primera, realizada por el desarrollador del proyecto durante el progreso de este. La segunda ha sido realizada por un conjunto de usuarios ajenos al proyecto.

5.1. Pruebas por el desarrollador

Durante el desarrollo se han ido realizando pruebas de las mecánicas y funcionalidades añadidas. Estas pruebas se realizaban dentro de cada *sprint*, con el objetivo de que los errores no se fueran acumulando y que al pasar de *sprint* se asegurara un mínimo de calidad de lo desarrollado anteriormente. Por un lado, hay algunas características que primero han sido probadas en entornos controlados para poder acotar mejor los errores que iban surgiendo y, así, determinar si estos errores eran problema de la propia mecánica o de su interacción con otras partes del software.

Por otro lado, todas las funcionalidades han sido finalmente probadas con el resto funcionando, para asegurar un buen desarrollo de la visita virtual. Además, hay algunas características cuyo funcionamiento no ha podido ser comprobado hasta realizar pruebas en el escenario de la visita.

5.2. Pruebas de usuarios

Cuando la visita virtual ha sido probada en profundidad por el desarrollador, la siguiente parte de las pruebas serán la toma de contacto de los usuarios con el proyecto. Para Malaca, se han seleccionado una serie de usuarios para poder probarla. Esto es debido a que el número de interacciones de *Convai* está limitado de forma mensual. Aun con 3000 interacciones adquiridas para este proyecto, se han tenido que realizar muchas pruebas internas por lo que se ha limitado el número de usuarios que han accedido a la prueba.

A algunos usuarios se les adjuntó la visita virtual a través de *Google Drive* [43]. Sin embargo, algunos de estos usuarios no disponían de dispositivos lo suficientemente potentes para hacer funcionar de forma correcta Malaca. La solución fue subir el proyecto a la plataforma **Vagon** [44], una plataforma de *Streaming* de aplicaciones tanto para Unreal Engine como para otros motores gráficos como *Unity* [45]. Gracias a esto, más usuarios pudieron probar la visita virtual y dar su feedback. *Vagon* sólo otorgó 15$ de prueba para este proyecto durante una semana, por lo que actualmente la única forma de probar Malaca es a través del archivo subido a Google Drive.

A continuación, se expondrán los resultados de este formulario, dividiéndose en varias secciones.

5.2.1. Perfil del jugador

Se ha elegido una variedad diversa de usuarios de prueba, tanto por la edad como por la familiaridad de los controles de un juego en ordenador (Fig. 107). Esto ha sido con el objetivo de asegurar que los controles de la visita virtual han sido simplificados lo suficiente para que una gran cantidad de usuarios puedan probarlos sin complicaciones. También, como se observa en la Fig. 106, hay muchos usuarios que nunca han hecho una visita virtual.

Fig. 106. Gráfico de realización de visita una virtual

Fig. 107. Gráfico de familiaridad con los controles de PC

5.2.2. Experiencia de juego

Se puede observar en la Fig. 108 que la mayoría de los usuarios no han tenido muchos problemas para encontrar las figuras. Una parte de éstos no ha tenido ninguna dificultad y un único usuario sí que ha tenido problemas para ello. Tras analizar los resultados se decidió no mover las figuras de donde estaban ya que, si fueran demasiado obvias, el usuario se acercaría directamente a recogerlas en vez de hablar con Pompeia.

Fig. 108. Gráfico de dificultad para encontrar las piezas

En la Fig. 109 se observa que algo menos de la mitad de los usuarios tuvieron algún problema a la hora de comunicarse con el chatbot. El error encontrado por varios usuarios fue que a veces Pompeia dejaba de hablar en mitad de una frase o no lo hacía directamente (aun cuando el texto ya había sido generado). Analizando los logs de Unreal, se determinó que el problema estaba siendo generado por el *LipSync Component*, el componente encargado de sincronizar los labios del metahuman con lo que está hablando. El componente tardaba a veces en funcionar, por lo que el metahuman le esperaba, lo que generaba el error. Se decidió desactivar el componente para asegurar el buen funcionamiento de Malaca, ya que este error era más frecuente que ocurriera en ordenadores con menos recursos.

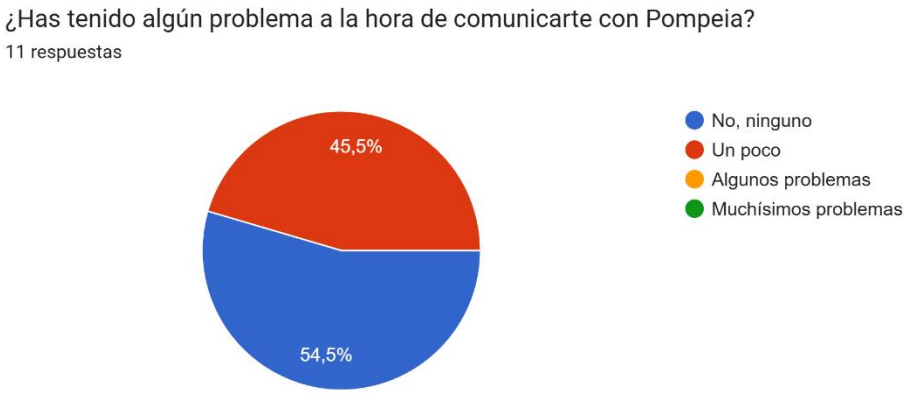

Fig. 109. Gráfico de problemas de comunicación con Pompeia

Como se observa en las dos figuras siguientes, tanto la información dada durante Malaca como las interacciones con Pompeia han tenido una muy buena recepción por parte de los usuarios. Es importante destacar esto ya que este era el objetivo principal de la visita virtual: que el usuario disfrutara de aprender de la historia romana de Málaga y que las interacciones con el chatbot fueran lo más agradables e inmersivas posibles.

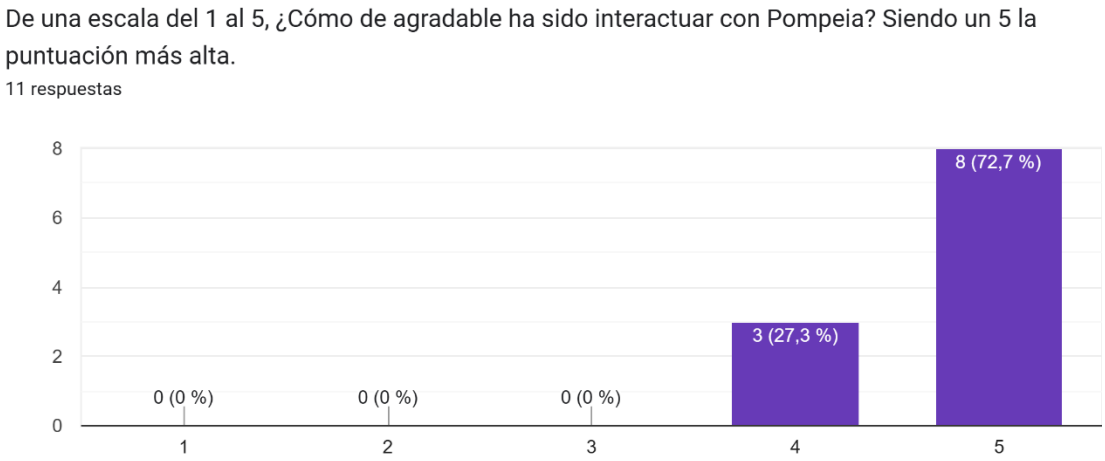

Fig. 110. Gráfico de satisfacción con las interacciones con Pompeia

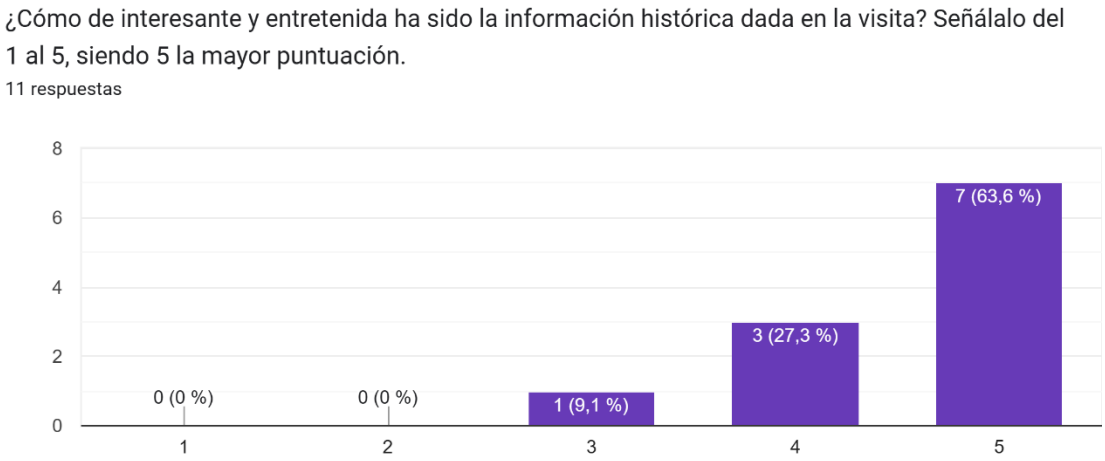

Fig. 111. Gráfico de satisfacción con la información histórica dada

También, como se observa en la Fig. 112, la mayoría de los usuarios excepto uno ha completado Malaca. Esto es debido a que el usuario dispuso de poco tiempo para probar la visita virtual.

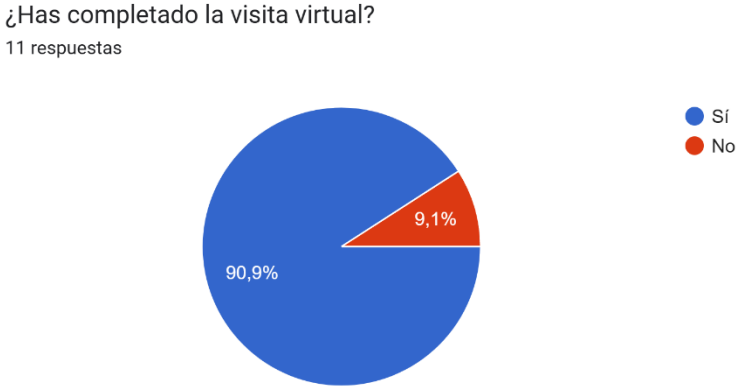

Fig. 112. Gráfico de compleción de la visita

5.2.3. Experiencia de uso

Para finalizar, se les preguntó a los usuarios su experiencia en la navegación de los menús, mostrando su comodidad de forma determinante. Como se observa en la Fig. 114, la satisfacción sobre Malaca ha sido muy buena, signo del trabajo realizado.

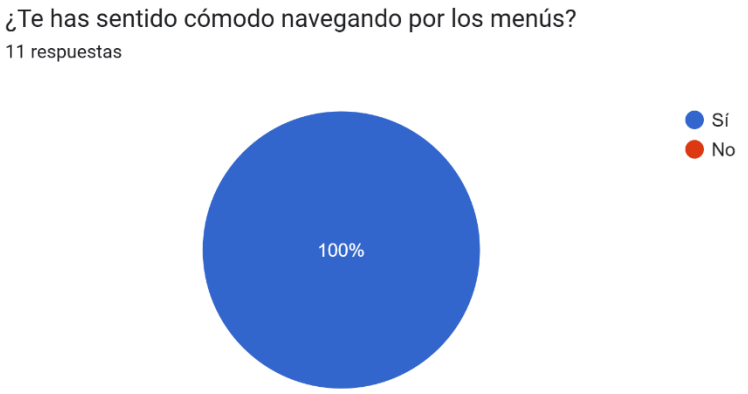

Fig. 113. Gráfico de comodidad de navegación por los menús

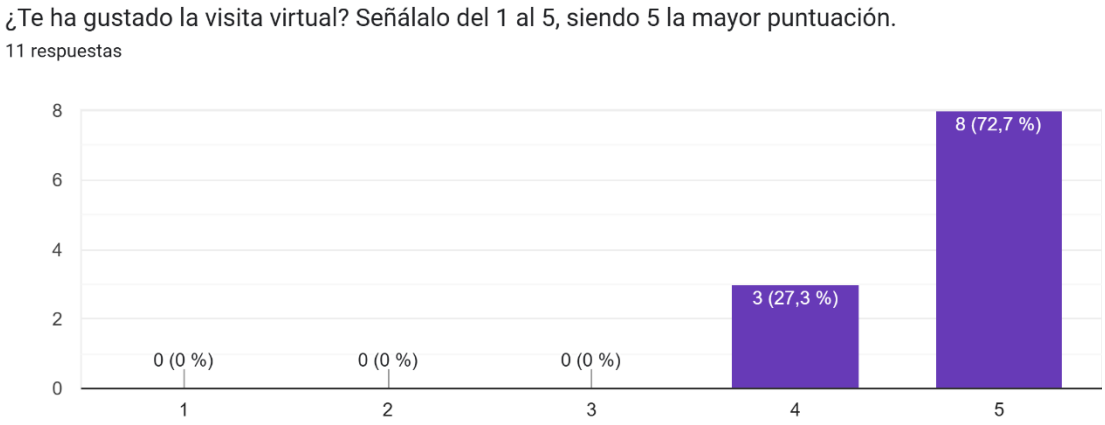

Fig. 114. Gráfico de satisfacción con la visita virtual

6

Conclusiones y líneas futuras

6.1. Conclusiones

Observando los resultados obtenidos en el apartado de pruebas, se puede llegar a determinar que el objetivo del proyecto ha sido complido.

Es importante mencionar que se han encontrado varios problemas en la realización de este trabajo, pese a los buenos resultados finales. Uno de los más importantes era el desconocimiento del desarrollador de Unreal Engine, ya que este sólo había tenido experiencia con Unity. Esto supuso un largo periodo de formación hasta dominar las bases del motor gráfico. Otros apartados que han sido un desafío son: la generación de los modelos 3D con fotogrametría y la edición visual del modelo 3D del teatro. Esto es porque se subestimó en primera instancia la cantidad de trabajo necesario para corregir y mejorar los modelos.

También ha habido problemas con la integración de *Convai* en Unreal, que ha estado sujeta a una gran cantidad de fallos. Para hacer que el chatbot funcionara correctamente, se han tenido en cuenta varios elementos descritos en el punto 4.6. Aunque el chatbot funcionara bien en la página de *Convai*, a la hora de hacer el traspaso al metahuman. comenzaron a surgir incidencias como que el personaje no pudiera moverse de forma correcta o que no encontrara las marcas que debía seguir para continuar la visita virtual. Otro problema ha sido que a veces el plugin de *Convai* fallaba, haciendo necesario reiniciar Malaca para que la visita volviera a funcionar. El error que generaba el *LipSync Component* fue solucionado, como se mencionó en el apartado de pruebas, desactivándolo.

Para finalizar el apartado de conclusiones, cabe mencionar que este trabajo ha sido especialmente interesante para poner en práctica muchos conocimientos

aprendidos durante el grado. Realizar este proyecto ha permitido poner en uso estos conocimientos en un ámbito distinto al habitual, como lo son el del desarrollo en un motor de videojuegos y el uso de chatbots.

6.2. Líneas futuras

Existen principalmente dos caminos a tomar para este proyecto:

1. Proseguir con el desarrollo de Malaca, ampliando las dimensiones de lo ya desarrollado. Esto se podría realizar, por ejemplo, creando nuevas figuras. También se podrían diseñar nuevas funcionalidades, como un "diario" donde el usuario pueda leer la información aprendida. Otra idea interesante sería adaptar el proyecto a la realidad virtual (VR), haciendo que la inmersión en la visita virtual fuera aún mayor. Se podrían añadir, además, nuevos personajes metahumans para que interactúen tanto entre sí como con el usuario.
2. Comenzar con el desarrollo de una idea nueva, partiendo de bases distintas para explorar el mundo de las visitas virtuales. Se podría plantear la creación de una visita virtual en otra ubicación histórica en Málaga, como la Alcazaba o Gibralfaro. También se podría desarrollar la posibilidad de generar escenarios que simulen las ruinas en una determinada época histórica.

En cualquiera de las líneas futuras, los conocimientos adquiridos durante el desarrollo serán determinantes para reducir tiempos y aumentar la calidad del producto final. Además, contar con artistas 3D para la edición de las figuras y el modelo del escenario sería muy beneficioso para obtener un resultado más profesional.

Referencias

[1] «Unreal Engine 5». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://www.unrealengine.com/en-US/unreal-engine-5

[2] «Convai». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://www.convai.com/auth/login

[3] «Chatbot: ¿qué es, cómo funciona y para qué sirve? - Telefónica». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://www.telefonica.com/es/sala-comunicacion/blog/chatbot-que-es-como-funciona-sirve/

[4] «¿Han llegado los metahumanos e influencers virtuales para quedarse?» Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://intel.goodrebels.com/p/han-llegado-los-metahumanos-e-influencers

[5] «MetaHuman Creator». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://metahuman.unrealengine.com/

[6] «Blueprints y C++ en Unreal Engine | Rodrigo Araujo Soria». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://rodaraujo.com/es/blog/blueprints-cpp-unreal-engine/

[7] «10 mejores free assets para Unreal Engine». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://www.ut-hub.com/10-mejores-free-assets-para-unreal-engine/

[8] «Fotogrametría - Wikipedia, la enciclopedia libre». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Fotogrametr%C3%ADa

[9] «Visita virtual - Wikipedia, la enciclopedia libre». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Visita_virtual

[10] «Visitas Virtuales». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://www.visitasvirtuales.com/

[11] «Conoce la Plaza de la Merced de Málaga con nuestra IA Victoria la Malagueña - YouTube». Accedido: 29 de agosto de 2025. [En línea]. Disponible en: https://www.youtube.com/watch?v=va3JdnVUPhs

[12] «Visita virtual - | Ministerio de Cultura». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://www.man.es/man/mandigital/visitavirtual.html

[13] «¿Qué es Scrum? - Azure DevOps | Microsoft Learn». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://learn.microsoft.com/es-es/devops/plan/what-is-scrum

[14] «Scrum (desarrollo de software) - Wikipedia, la enciclopedia libre». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)

[15] «Epic Games». Accedido: 3 de septiembre de 2025. [En línea]. Disponible en: https://www.epicgames.com/site/es-ES/home

[16] «Unreal Engine - Wikipedia, la enciclopedia libre». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Unreal_Engine

[17] «Blueprints Visual Scripting in Unreal Engine | Unreal Engine 5.6 Documentation | Epic Developer Community». Accedido: 20 de agosto de 2025.

[En línea]. Disponible en: https://dev.epicgames.com/documentation/en-us/unreal-engine/blueprints-visual-scripting-in-unreal-engine#generalscripting
[18] «RealityScan | 3D models from images and laser scans - RealityScan». Accedido: 22 de agosto de 2025. [En línea]. Disponible en: https://www.realityscan.com/en-US
[19] «Blender - The Free and Open Source 3D Creation Software — blender.org». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://www.blender.org/
[20] «Roman style outfits - Download Free 3D model by A9908244 (@A9908244) [629f546]». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://sketchfab.com/3d-models/roman-style-outfits-629f5466d75f42a4abd377a951b5574e
[21] «Relaxing Fantasy Ancient Rome / Roman Music & Ambience III | Samvyke Harp | sleep, study, work - YouTube». Accedido: 20 de agosto de 2025. [En línea]. Disponible en: https://www.youtube.com/watch?v=n0DH-DIXTZ8
[22] «Freesound». Accedido: 23 de agosto de 2025. [En línea]. Disponible en: https://freesound.org/
[23] «Acerca de Projects - Documentación de GitHub». Accedido: 23 de agosto de 2025. [En línea]. Disponible en: https://docs.github.com/es/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects
[24] «Trello». Accedido: 23 de agosto de 2025. [En línea]. Disponible en: https://trello.com/es
[25] «Requisito (sistemas) - Wikipedia, la enciclopedia libre». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Requisito_(sistemas)
[26] E. Medina y C. Rossi, «Análisis y Diseño de Sistemas de Información - Requisitos de software», Málaga, 2021.
[27] C. Rossi y E. Medina, «Análisis y Diseño de Sistemas de Información - Casos de uso», Málaga, 2021.
[28] «Visual Paradigm». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://www.visual-paradigm.com/
[29] «Convai Review 2025: What It Is, How to Use It & Is It Worth It?» Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://aihungry.com/tools/convai
[30] «Dibuja, crea y edita con Paint | Microsoft Windows». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://www.microsoft.com/es-ES/windows/paint?msockid=1858852bbced6c2d1d5491b4bd456dfd#imagecreator
[31] «Figma | La herramienta de diseño de interfaz colaborativo». Accedido: 25 de agosto de 2025. [En línea]. Disponible en: https://www.figma.com/es-la/
[32] «Museo de Málaga». Accedido: 30 de agosto de 2025. [En línea]. Disponible en: https://www.museosdeandalucia.es/web/museodemalaga
[33] «Retrato del emperador Antonino Pío - Obras Singulares - Museo de Málaga». Accedido: 30 de agosto de 2025. [En línea]. Disponible en: https://www.museosdeandalucia.es/web/museodemalaga/obras-singulares/-/asset_publisher/GRnu6ntjtLfp/content/retrato-del-emperador-antonino-pio?redirect=%2Fweb%2Fmuseodemalaga%2Fobras-singulares&inheritRedirect=true
[34] «Fuente Marcellus SC». Accedido: 28 de agosto de 2025. [En línea]. Disponible en: https://fontmeme.com/fuentes/fuente-marcellus-sc/

[35] «Setting Up a Game Mode in Unreal Engine | Unreal Engine 5.6 Documentation | Epic Developer Community». Accedido: 27 de agosto de 2025. [En línea]. Disponible en: https://dev.epicgames.com/documentation/en-us/unreal-engine/setting-up-a-game-mode-in-unreal-engine

[36] «Core AI Settings | Convai Documentation». Accedido: 26 de agosto de 2025. [En línea]. Disponible en: https://docs.convai.com/api-docs/convai-playground/character-customization/core-ai-settings

[37] «Narrative Design - YouTube». Accedido: 26 de agosto de 2025. [En línea]. Disponible en: https://www.youtube.com/playlist?list=PLn_7tCx0Chip2mfSbOkqJLevEbm3jDuNV

[38] «Narrative Design | Convai Documentation». Accedido: 26 de agosto de 2025. [En línea]. Disponible en: https://docs.convai.com/api-docs/convai-playground/character-customization/narrative-design

[39] «Bridge is now a part of Unreal Engine 5 Early Access - YouTube». Accedido: 27 de agosto de 2025. [En línea]. Disponible en: https://www.youtube.com/watch?v=ParMLadMTaQ

[40] «MetaHuman Plugin | Fab». Accedido: 27 de agosto de 2025. [En línea]. Disponible en: https://www.fab.com/listings/055a6486-ad17-4590-aa1e-261d47f7f041

[41] «Custom Clothes for Metahuman in Unreal Engine — BEGINNER TUTORIAL - YouTube». Accedido: 26 de agosto de 2025. [En línea]. Disponible en: https://www.youtube.com/watch?v=gBEUvLq3Rak

[42] «Convai Unreal Engine Integration - YouTube». Accedido: 27 de agosto de 2025. [En línea]. Disponible en: https://www.youtube.com/playlist?list=PLn_7tCx0ChiogfggG1AVo6IkELQSLt6o3

[43] «Google Drive: Comparte archivos en línea con el almacenamiento seguro en la nube | Google Workspace». Accedido: 31 de agosto de 2025. [En línea]. Disponible en: https://workspace.google.com/products/drive/

[44] «Vagon». Accedido: 31 de agosto de 2025. [En línea]. Disponible en: https://vagon.io/streams

[45] «Plataforma de desarrollo en tiempo real Unity | Motor 3D, 2D, VR y AR». Accedido: 31 de agosto de 2025. [En línea]. Disponible en: https://unity.com/es

[46] «WinRAR - Sitio oficial WinRAR en español». Accedido: 3 de septiembre de 2025. [En línea]. Disponible en: https://winrar.es/

Apéndice A. Manual de Instalación

A.1. Pasos para la instalación

El archivo comprimido se entregará junto a esta memoria. Se deberán seguir los siguientes pasos para realizar la instalación de la aplicación:

1. Descargar el archivo comprimido (.zip):

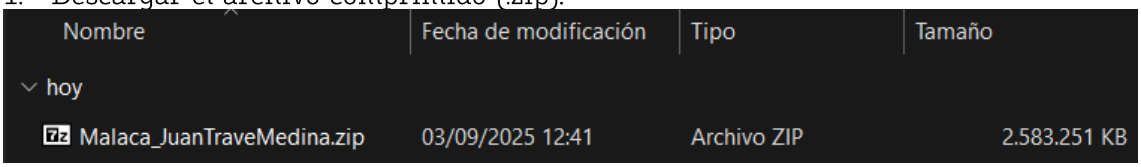

Fig. 115. Archivo .zip en el explorador de archivos de Windows 11

2. Descomprimir el archivo con una herramienta gestora de archivos comprimidos como **Winrar** [46]:

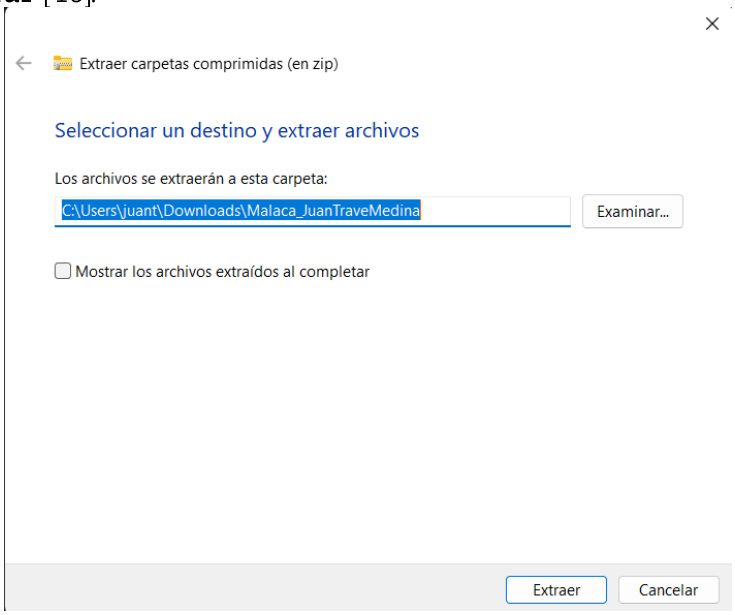

Fig. 116. Descompresión del archivo .zip

3. Buscar dentro de la carpeta que se ha creado el ejecutable (.exe). La ruta será la siguiente: *C:\ (...) \Malaca_JuanTraveMedina*

4. Hacer doble clic sobre el ejecutable (tercer elemento en la Fig. 117) y comenzar la visita:

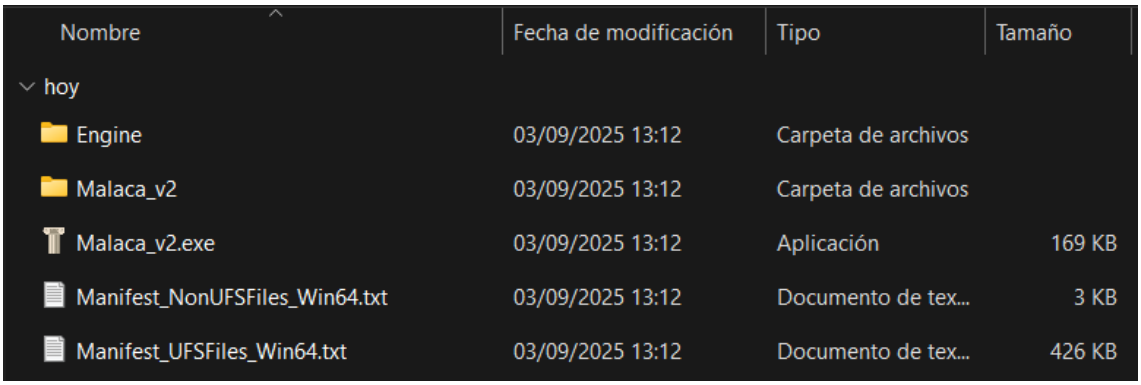

Fig. 117. Vista de la carpeta una vez descomprimida

Apéndice B. Manual de usuario

B.1. Historia

El usuario tomará el papel de un ciudadano más, el cual se introduce en la experiencia virtual para aprender sobre el pasado de la actual Málaga.

Pompeia Phlylocyria será el personaje que dará la bienvenida al usuario en Malaca. También se encargará de guiar a éste por la experiencia. Esta mujer malacitana venida del pasado tiene una gran cantidad de conocimientos de historia, desde la fundación de la ciudad por los fenicios hasta el estado de Malaca a finales de la Edad Antigua.

Su marido se llama Publius Grattius Aristocles y juntos patrocinaron diversas obras para embellecer la ciudad. Pompeia ha dedicado sus días a participar en el gobierno de la ciudad, contribuyendo a aumentar el reconocimiento de Malaca. Su gran sueño es que Malaca se convierta en una colonia romana, el estatus más alto que una ciudad puede tener en el imperio. Le encanta enseñar y explicar la historia de su ciudad a los ciudadanos y visitantes que le preguntan por ella.

B.2. Controles

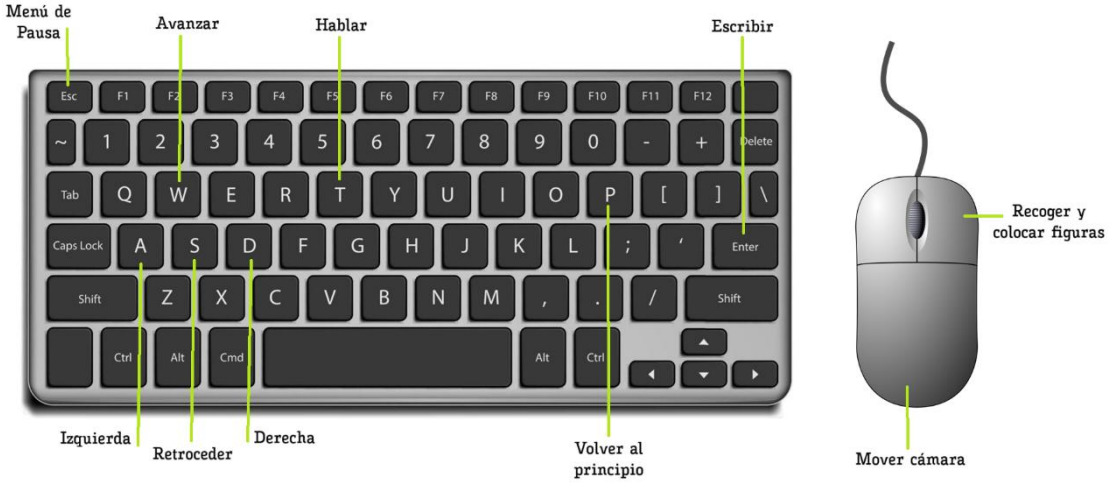

Fig. 118. Controles de Malaca

B.3. Navegación por menús

La navegación por los distintos menús se realizará con el cursor del ratón. Cuando uno de estos menús aparezca, también lo hará el cursor. Si el usuario vuelve a la visita virtual éste desaparecerá.

B.4. Configuración de Convai

Una vez dentro de Malaca, el usuario podrá acceder a la configuración base de *Convai* pulsando la tecla **F10**. Por defecto no debería ser necesario acceder a ella, pero para aquellos usuarios que tengan algún problema con el audio durante la visita pueden:

1. Cambiar el dispositivo de entrada de audio y probarlo.
2. Aumentar la ganancia de audio del micrófono si se escucha algo bajo.
3. Cambiar el aspecto de la interfaz, haciendo más grande o alargado el chat de *Convai*. El chat por defecto ya ha sido modificado para que contenga más información, pero si el usuario lo desea puede cambiarlo.
4. Cambiar el nombre con el que Pompeia se refiere al usuario. Por defecto será "Ciudadano".

Se recomienda no interactuar con la opción de usar la *API Key* ya que no ha sido usada en este proyecto.

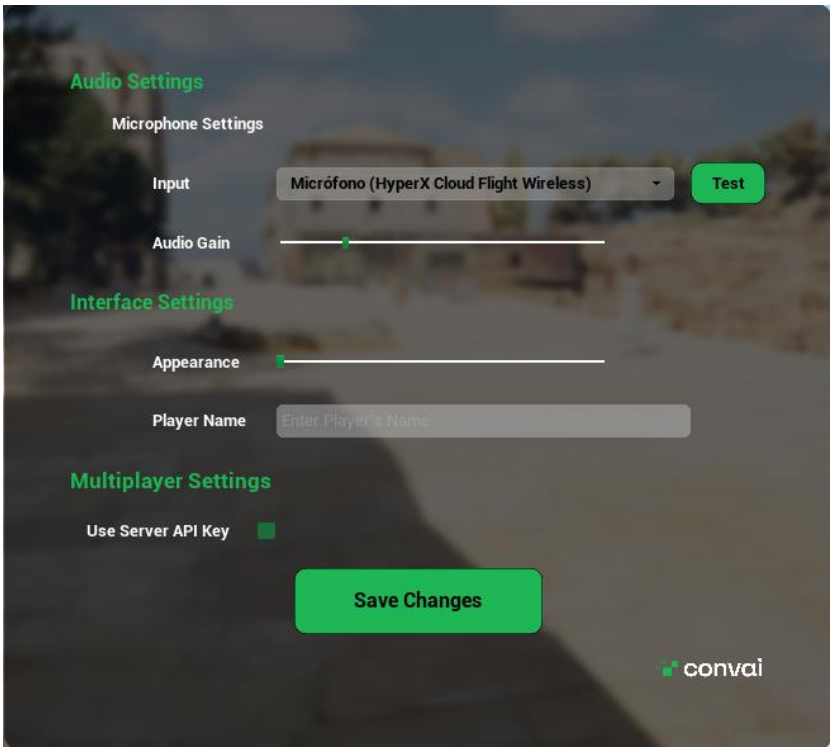

Fig. 119. Configuración de *Convai*

B.5. Objetivo principal

El principal objetivo de Malaca será la recuperación de las cinco figuras perdidas del museo de Málaga. Será Pompeia la que instará al usuario a recuperarlas una vez que este haya avanzado en su conversación. Como se puede observar arriba a la izquierda de la Fig. 120, habrá un contador visible durante la visita virtual, el cual se irá actualizando conforme se vayan encontrando las figuras.

Fig. 120. Captura de la visita virtual

Si el usuario desea rehacer la visita virtual una vez que la haya completado, se podrá obtener la información de las figuras sin llegar a hablar con Pompeia. Esto se debe a que la conversación inicial y la búsqueda de las figuras tienen su lógica interna separadas para favorecer que el usuario pueda realizar lo que más le interese de la experiencia.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga