TRABAJOS FIN DE GRADO

Universidad de Jaén

Escuela Politécnica Superior de Jaén

APLICACIÓN AR PARA MANIPULACIÓN Y PUESTA EN VALOR DE MATERIAL ARQUEOLÓGICO

Autor: Daniel Rodríguez Martínez

Grado: Grado en Ingeniería Informática

Director: Rafael Jesús Segura Sánchez, José Manuel Fuertes García
Departamento del director: Departamento de informática

Fecha: 03/07/2024

CREA

(Página intencionalmente en blanco)

Universidad de Jaén

Departamento de Informática

Don Rafael Jesús Segura Sánchez y Don José Manuel Fuertes García, tutores del Trabajo Fin de Grado titulado: '**Aplicación AR para manipulación y puesta en valor de material arqueológico**', que presenta Don Daniel Rodríguez Martínez, otorgan el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2024

El alumno:

RODRIGUEZ MARTINEZ DANIEL - 77373610T

Firmado digitalmente por RODRIGUEZ MARTINEZ DANIEL - 77373610T
Fecha: 2024.07.03 12:26:05 +02'00'

Daniel Rodríguez Martínez

Los tutores:

Firmado por SEGURA SANCHEZ RAFAEL JESUS - ***0385** el día 03/07/2024 con un

Firmado digitalmente por FUERTES GARCIA JOSE MANUEL - 26478036E

Rafael Jesús Segura Sánchez

José Manuel Fuertes García

(Página intencionalmente en blanco)

Agradecimientos

Quiero agradecer en primer lugar a mis padres y familiares, que me han apoyado y ayudado a ser la persona que soy y a llegar hasta donde he llegado. A mi prima Estrella, que siempre ha estado pendiente de mí y me ha ayudado en muchos aspectos.

A los amigos que he hecho durante la carrera, que han estado conmigo y para mí tanto en los buenos momentos como en los malos.

También a mis profesores de la UJA, que han sabido guiarme para llegar hasta el final de esta etapa de mi vida.

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Sin especialidad
Mención (solo TFG)	Sistemas Gráficos
Idioma	Español
Tipo	General
TFT en equipo	No
Autor/a	Daniel Rodríguez Martínez
Fecha de asignación	Haga clic aquí o pulse para escribir una fecha.
Descripción corta	En los últimos años la utilización de tecnologías basadas en Realidad Aumentada (AR, de Augmented Reality) han tomado mucho auge debido fundamentalmente al enorme abaratamiento de los dispositivos móviles y a la mejora de las técnicas de posicionamiento y registrado. La AR persigue combinar el mundo real con un mundo virtual de manera que el usuario pueda, en tiempo real, interactuar con ambos mundos simultáneamente. Para ello se requiere *registrar* mundo real y mundo virtual para que la superposición sea lo más precisa como sea posible. En este TFG se persigue utilizar las Microsoft HoloLens 2 en una aplicación del ámbito de la arqueología. El trabajo consistirá en explorar las funcionalidades de dicha instrumentación y construir una aplicación AR dirigida a la manipulación de material arqueológico, poniendo especial relieve con la integración de los elementos virtuales generados como aumentadores de la realidad que rodea al usuario. Las funcionalidades serán definidas mediante la interrogación a profesorado del Grado en Arqueología, con el fin de identificar posibles acciones que puedan incorporarse y generar una aplicación con utilidad educativa.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

NORMAS UTILIZADAS COMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
*Estas normas se han utilizado **como base o referencia** para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como **proyecto** la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.*	

Contenido

Índice de ilustraciones

Índice de tablas

1 ESPECIFICACIÓN DEL TRABAJO

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos **inspirados** en los criterios y recomendaciones que establece la norma UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*".

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el apartado 6 (Definiciones y abreviaturas).

1.1 Introducción

La realidad aumentada (RA) [1] es el término que se usa para describir al conjunto de tecnologías que permiten que un usuario visualice parte del mundo real a través de un dispositivo tecnológico con información gráfica añadida por este. Dicho dispositivo es capaz de añadir información virtual a la escena real, de manera que los elementos físicos tangibles se pueden conectar con los elementos virtuales.

Se puede afirmar que la RA se caracteriza por:

- Combinar el mundo real y el virtual
- Ofrecer una interacción en tiempo real
- Adaptarse al entorno en que se insiere
- Interactuar con las capacidades físicas del entorno en tres dimensiones (3D)

Es importante resaltar la diferencia de este concepto del concepto de Realidad Virtual (RV), comúnmente confundidos y que en muchos casos se observan de manera conjunta. Mientras que la RV crea un ambiente nuevo y completamente virtual que reemplaza a lo "real", la RA proyecta información (como imágenes, gráficos, textos…) en el mundo real. También es posible definir un paradigma mixto (Realidad Mixta, RM) que combina elementos modelados en 3D para añadir posteriormente al entorno virtual.

Siendo este paradigma de interacción una oportunidad magnífica para que el usuario se acerque a un producto, entorno u otro tipo de concepto interactivo que por lo general esté lejos de su alcance, la realidad aumentada es un tema de investigación interesante con aplicaciones aún por descubrir.

Por consiguiente, en el presente trabajo se pretende aprovechar las características del dispositivo de Realidad Mixta *Microsoft HoloLens 2* [2] para crear una aplicación de utilidad en el ámbito de la Arqueología, que permita una reconstrucción básica de un enclave arqueológico mediante la colocación de paredes u otras estructuras, además de poder manipular distintos objetos previamente modelados, todo ello mediante la interacción en RA utilizando gestos.

Ilustración 1: Microsoft HoloLens2

1.2 Objetivos del trabajo

El objetivo principal del trabajo es presentar una aplicación desarrollada para las HoloLens2 a través de la cual sea posible apreciar e interactuar con elementos tales como *exvotos íberos* [3] o enclaves arqueológicos como el *Oppidum Íbero de Puente Tablas* [4]. La motivación más destacada que resaltar es la de explorar la usabilidad de la RA en entornos educativos, tratando de descubrir por el camino los posibles usos que sea posible dar a este nuevo nivel de inmersión. También estudiar la capacidad de adaptación de los usuarios al cambio de paradigma de interacción. Se desarrollará la aplicación mediante el uso de un motor de desarrollo de videojuegos junto al kit de desarrollo de software (SDK) creado específicamente para el dispositivo HoloLens2. Se destacan una serie de objetivos:

1. Estudio comparativo de las diferentes herramientas de desarrollo de aplicaciones de AR para diferentes dispositivos.
2. Conocer y comprender las dificultades de aplicación de la informática en otras disciplinas.
3. Análisis de requisitos parta la construcción de una aplicación AR para educación a nivel de Grado en Arqueología.
4. Construcción de un prototipo de aplicación basada en AR para simular acciones en el ámbito de la arqueología, atendiendo a diferentes entornos reales.
5. Generación de documentación comprensible que permita la replicación del trabajo.
6. Aprovechar la tecnología de las HoloLens2 para apoyar una interacción con la interfaz libre de controladores físicos e intuitiva, promoviendo el uso de gestos, comandos de voz u otros métodos de interacción con el objetivo de minimizar el esfuerzo de aprendizaje de la herramienta, ayudando a concentrar la atención en el contenido mostrado.

Ilustración 2: Exvotos íberos de bronce

Ilustración 3: Ciudad Íbera de Puente Tablas

1.3 Antecedentes y estado del arte

La primera vez que el concepto de RA se utilizó fue en 1901. *Frank L. Baum* [5] incluyó la idea de la creación de unas gafas electrónicas que superponían datos sobre las personas que visualizaban en una novela titulada *The master key* [6], lo llamó *character maker.* En 1957 el que fue el inventor de la RV, *Morton Heilig* [7] volvió a referirse al término cuando inventó la primera máquina de realidad virtual *Sensorama* [8], que combinaba video 3D, color, audio, vibraciones, viento y olor, aunque esta máquina no estaba controlada por ordenador. Sin embargo, no fue hasta 1989 que se acuñó el término “realidad aumentada” por el informático alemán *Jaron Lanier* [9]

Ilustración 4: La máquina Sensorama

Desde las primeras concepciones y aplicaciones, la RA ha experimentado un desarrollo significativo, con avances tecnológicos que han ampliado sus aplicaciones y mejorado su experiencia de usuario.

El avance del hardware ha sido fundamental para el desarrollo de la RA. Dispositivos como las *Microsoft HoloLens, Magic Leap* [10] y las gafas inteligentes de empresas como *Google* y *Apple* han llevado la RA más allá de los dispositivos móviles y han proporcionado experiencias más inmersivas. Se ha logrado una visualización y seguimiento mejorados en tiempo real, los cuales se aplican mediante algoritmos de procesamiento de imágenes y técnicas de visión por computador más sofisticados [11].

Ilustración 5: Dispositivo Magic Leap

Las aplicaciones de la RA también se están ampliando a una amplia gama de campos desde el entretenimiento hasta la medicina o la industria. Un ejemplo de esto puede ser el videojuego para móviles Pokémon GO o aplicaciones para la visualización de datos médicos en 3D e información para cirujanos [12].

Ilustración 6: Pokémon GO 2016

Se ha trabajado en interfaces de usuario más intuitivas que permiten a los usuarios interactuar de manera natural con la interfaz en RA, lo cual incluye comandos de voz, gestos y reconocimiento de objetos en el mundo real. A su vez se desarrollan herramientas y plataformas para facilitar la creación de contenido en RA, incluyendo SDK, frameworks y plataformas de creación de contenido que permiten a los desarrolladores el crear espacios de RA más eficientemente.

En cuanto al ámbito de la educación se refiere, integración de la RA ha evolucionado considerablemente desde sus primeras aplicaciones, que se centraban en demostraciones simples y actividades de visualización. Hoy en día, las aplicaciones de RA abarcan una amplia variedad de disciplinas y niveles educativos, desde la educación primaria hasta la educación superior y la formación profesional. Algunas de las aplicaciones más destacadas incluyen:

1. **Visualización de Contenidos Complejos**: La RA permite a los estudiantes visualizar conceptos abstractos y complejos de manera más tangible. Por ejemplo, en ciencias, los estudiantes pueden explorar modelos tridimensionales de moléculas, sistemas solares o anatomía humana,

facilitando una comprensión más profunda de estos temas. Un estudio mostró que el uso de RA en la enseñanza de la biología mejoró significativamente la comprensión conceptual de los estudiantes. [13]

2. **Aprendizaje Interactivo y Colaborativo:** Las aplicaciones de RA pueden fomentar la interactividad y la colaboración entre los estudiantes. Mediante actividades y juegos educativos basados en RA, los estudiantes pueden trabajar en equipo para resolver problemas y completar tareas, lo que puede mejorar sus habilidades sociales y de resolución de problemas. Un estudio encontró que la RA puede aumentar la motivación y el compromiso de los estudiantes en actividades de aprendizaje colaborativo. [14]
3. **Inclusión y Accesibilidad:** La RA tiene el potencial de hacer la educación más inclusiva y accesible. Por ejemplo, los estudiantes con discapacidades pueden beneficiarse de aplicaciones de RA diseñadas para adaptar el contenido educativo a sus necesidades específicas, facilitando su participación plena en el proceso de aprendizaje. Diversos estudios sobre RA concluyen que la tecnología puede ser una herramienta poderosa para mejorar la accesibilidad educativa. [15]

El estado del arte de la RA está, por tanto, marcada por significativos avances en hardware, software y aplicaciones, lo cual amplía su alcance y utilidad en una variedad de campos. Sin embargo, sigue siendo un área de rápido desarrollo y con potencial para seguir creciendo y expandiéndose.

1.4 Requisitos iniciales

Cabe destacar que no se cuenta con requisitos concretos a la hora de desarrollar el proyecto. Se busca sobre todo la experimentación con las HoloLens2 y la intención de desarrollar una aplicación para el contexto docente. Definimos a continuación los requisitos a alto nivel que se busca satisfacer en la versión final del proyecto:

- Debería poderse visualizar un entorno arqueológico a escala, de manera que sea posible rodearlo y observarlo desde donde se desee.
- Debería poderse cargar en la escena diferentes objetos de interés que se cargarán en la aplicación de manera previa a su ejecución.

- Debería desarrollarse un método de adaptar la interfaz al entorno virtual, de manera que sea posible acceder a las diferentes funcionalidades de la manera más cómoda posible.

1.5 Hipótesis y restricciones

Previa a la realización del trabajo se ha obtenido experiencia con motores gráficos de videojuegos y sobre el dispositivo de realidad mixta HoloLens 2, además de conocimientos en lenguajes de programación como Java, C#, C++ y Python que suelen ser de gran ayuda para la programación de aplicaciones gráficas. La obtención de dichos conocimientos resultará de gran ayuda en la programación y desarrollo de la aplicación.

El proyecto usará modelos 3D proporcionados por el Instituto Universitario de Investigación en Arqueología Ibérica, el cual tiene a disposición pública muchos materiales escaneados y texturizados en Sketch Fab [16]. Sin embargo, en lo que se refiere a futuros usos de la herramienta, si se busca añadir nuevos modelados o enclaves arqueológicos, será necesaria una previa digitalización. Puesto que el Instituto proporciona algunos de estos modelos, no será necesario llevar a cabo dicho proceso en lo que al desarrollo del proyecto se refiere.

Se hará uso de un SDK especializado. El uso de dicha herramienta combinado con el motor de videojuegos proporcionará funcionalidades útiles y ayudará a ahorrar tiempo. Se describirá con detalle este elemento en el apartado 1.7.1.

El TFT se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo. Sin embargo, las herramientas escogidas y la colaboración del Instituto de Arqueología contribuirán a agilizar el proceso.

1.6 Estudio de alternativas y viabilidad

Durante la primera etapa del proyecto se ha realizado un estudio de las diferentes herramientas que serán utilizadas en el desarrollo. A continuación, se presentarán las diferentes alternativas y elecciones realizadas.

1.6.1 Motor gráfico

La elección de un motor gráfico es una decisión importante para la realización de este tipo de proyectos y por lo tanto se ha realizado un estudio comparativo de diferentes motores

1.6.1.1 Unreal Engine

Unreal Engine es un motor de videojuegos de código abierto con pagos de regalías para uso comercial, creado por la compañía *Epic Games* y dado a conocer inicialmente en el *shooter* en primera persona *Unreal* en 1998. Requiere conocimientos de programación con *C++.* La última versión estable del motor es *Unreal Engine 5* y fue presentada en 2020 [17].

Una de las características principales del motor es el uso de la tecnología *Nanite* [18], la cual es una tecnología que permite importar material fotográfico de gran detalle y que permite importar casi cualquier representación tridimensional preexistente de objetos y entornos, permitiendo así el uso de elementos de alta calidad cinematográfica. También gestiona los niveles de detalle de estos objetos teniendo en cuenta factores como la plataforma de destino.

Otra característica del motor es el uso de la tecnología *Lumen* [19], descrita como *"una herramienta de iluminación y reflejos globales completamente dinámicos que te permite crear una iluminación indirecta que se adapta sobre la marcha a los cambios producidos en la iluminación directa o la geometría*". Esta herramienta elimina la necesidad de los artistas y desarrolladores de elaborar un mapa de iluminación, calculando los reflejos de la luz y las sombras en tiempo real.

Por el tipo de características que destacan sobre la herramienta, se puede deducir que la gran ventaja de este motor sobre otros es la excelente calidad gráfica

que ofrecen las aplicaciones desarrolladas usando el motor, sobre todo videojuegos. Algunos ejemplos son:

- Tekken 8, Ilustración 7
- Fortnite
- Dragon Ball FighterZ
- Final Fantasy VII Remake, Ilustración 8

A modo de resumen de las características de la herramienta se proporciona la Tabla 1: Características de Unreal Engine:

Característica	Valoración
Lenguaje de programación	C++
Calidad de gráficos	Alta
Coste de licencia	Gratuito (en el contexto del proyecto)
Herramientas de soporte	Lumen, Nanite

Tabla 1: Características de Unreal Engine

Ilustración 8: Final Fantasy VII Remake, 2020

Ilustración 7: Tekken 8, 2024

1.6.1.2 Unity 3D

Unity es un motor de videojuegos multiplataforma creado por *Unity Technologies* y anunciado por primera vez en 2005, originalmente para Mac Os X. Hoy en día es uno de los motores más frecuentes dado a su flexibilidad, su curva de aprendizaje y sus funcionalidades entre otras características.

Entre las principales características de *Unity* se encuentra la compatibilidad de uso junto a otras herramientas como *Blender, 3ds Max, Maya…* El motor gráfico utiliza OpenGL (en Windows, Mac y Linux), Direct3D (solo en Windows), OpenGL ES (en Android y iOS), e interfaces propietarias (Wii). Tiene soporte para mapeado de relieve, mapeado de reflejos, mapeado por paralaje, oclusión ambiental en espacio de pantalla, sombras dinámicas utilizando mapas de sombras, render a textura y efectos de post-procesamiento de pantalla completa.

En cuanto al uso de sus licencias, *Unity* es un software privativo con distintas opciones de licencia. Es posible acceder de manera gratuita a licencias de uso personal, aunque si se busca un enfoque con posibilidades de comercialización lo recomendable es adquirir una licencia Pro ya que ofrece herramientas exclusivas y soporte personalizado con prioridad. También es un lenguaje común para las asignaturas con temática de informática gráfica y ofrece una licencia educativa para uso gratuito de individuos con una cuenta de correo institucional, para uso exclusivo de educación.

Unity también tiene acceso a la *Unity Asset Store* [20], un recurso disponible para el editor que ofrece acceso a los usuarios a una amplia gama de categorías, incluyendo modelos 3D, texturas, materiales, efectos de sonido, partículas e incluso otros proyectos. Además, posibilita el comercio de activos *in-game* a desarrolladores y jugadores.

En resumen, *Unity* es un motor para tener en cuenta tanto para programadores como para usuarios inexpertos, dado que la gran cantidad de Assets disponibles en la asset store sumado al scripting en C# ofrece muchas posibilidades para aligerar el aprendizaje de la herramienta.

A modo de resumen de las características del motor se proporciona la tabla Tabla 2:

Características	Valoración
Lenguaje de programación	C# y Java
Calidad de gráficos	Buena
Coste de licencia	Gratuito (en el contexto del proyecto)
Herramientas de soporte	Unity Asset Store

Tabla 2: Características de Unity

Algunos ejemplos de juegos desarrollados usando el motor son:

- Hollow Knight, Ilustración 9
- Pokemon GO, Ilustración 6: Pokémon GO 2016
- Valheim

Ilustración 9: Hollow Knight, 2017

Ilustración 10: Valheim, 2023

1.6.2 Entorno de desarrollo (IDE)

El entorno de programación influirá mucho en la rapidez con la que se desarrollará el código fuente de la aplicación. Influirán aspectos como el lenguaje de programación que se utilizará o los plugin disponibles en el entorno que puedan ofrecer alguna funcionalidad o facilidad. Pasaremos por tanto a explorar algunas de las opciones planteadas al ser las más utilizadas a lo largo de la carrera.

1.6.2.1 Visual Studio

Visual Studio es un IDE compatible con múltiples lenguajes de programación y entornos web comúnmente utilizado para la creación de servicios web y aplicaciones que requieren de comunicación entre páginas web, estaciones de trabajo, dispositivos móviles o embebidos, entre otros [21].

Cuenta con extensiones específicas que aportan una gran comodidad a la hora de trabajar en determinados ámbitos, incluida una extensión para Unity.

1.6.2.2 Apache NetBeans

NetBeans es un IDE diseñado especialmente para el lenguaje de programación Java, aunque existe un gran número de módulos para extenderlo. Es un IDE gratuito y de código abierto [22].

Su principal atractivo con respecto al IDE rival escogido es el depurador, puesto que es una herramienta que se ha utilizado a lo largo de todo el grado que agiliza la corrección de errores y con la que se ha conseguido soltura.

1.7 Descripción de la solución propuesta

En este apartado se realizará una descripción de las soluciones propuestas para la realización del proyecto en base a las opciones viables presentadas en el apartado 1.6.

1.7.1 Kit de desarrollo de software (SDK)

Se ha escogido *Mixed Reality Toolkit* [23] [24] como kit de desarrollo. Este es un software *open-source* desarrollado por Microsoft en 2016 (el cual sigue actualizándose) para el desarrollo de aplicaciones en realidad aumentada y realidad mixta que permite su uso en una amplia variedad de dispositivos y plataformas. Consiste en una serie de componentes y características diseñadas para mejorar la experiencia de los usuarios y desarrolladores. El kit fue originalmente desarrollado por la salida de las *Microsoft HoloLens 1.*

El kit incluye una amplia variedad de componentes y características que permiten a los desarrolladores crear experiencias inmersivas, así como librerías adicionales que facilitan el scripting de las aplicaciones desarrolladas con las herramientas. En la Tabla 3 se dan a conocer algunas de las características y librerías que se han usado en el desarrollo del proyecto.

Nombre	Descripción
Input System	Proporciona a los desarrolladores la habilidad de leer y manejar eventos de entrada a nivel de dispositivo desde una gran variedad de fuentes.
Hand Tracking (HoloLens 2)	Permite el control de las HoloLens 2 con las manos
Eye Tracking (HoloLens 2)	Interpreta el movimiento de los ojos del usuario para proporcionar interacción precisa
Profiles	Permite el uso de perfiles configurables para manejar el uso y la preparación de herramientas en cada escena
UI Controls	Componentes de interfaz prefabricados para acelerar la creación de interfaces de usuario
Solvers	Componentes para la resolución de problemas complejos en realidad mixta tales como la colocación de los objetos y la manipulación 3D
Multi-Scene Manager	Sistema para administrar múltiples escenas en MR incluyendo transiciones entre escenas.
Spatial Awareness	Detecta elementos del entorno físico.
Diagnostic Tool	Interfaz con verificación en tiempo real de datos para facilitar el análisis y la depuración
Boundary System	Define los límites de uso de una aplicación en un área para mayor seguridad
In-Editor Simulation	Interfaz diseñada para probar interacciones MRTK en el propio editor sin tener que compilar la aplicación
Speech and Dictation	Utiliza reconocimiento de voz y lenguaje natural para interpretar palabras claves y comandos
MRTK Standard Shader	Sistema de shaders diseñado para implementar efectos visuales de calidad y eficientes en dispositivos MR

Tabla 3: Utilidades de MRTK

1.7.2 Solución propuesta para motor gráfico

A la hora de la elección del motor gráfico se han tenido que estudiar diferentes aspectos de los motores propuestos:

1. **Experiencia previa:** se cuenta con una experiencia en *Unity* moderada al haberse trabajado con la herramienta en la carrera y al haber colaborado en un proyecto anterior del mismo tipo con el Instituto. Por otro lado, no se ha obtenido experiencia en el uso de *Unreal Engine*.
2. **Lenguaje de programación:** se cuenta con experiencia en todos los lenguajes de programación requeridos en ambos motores para el desarrollo de los diferentes elementos requeridos, sin embargo, se valorará positivamente C# y Java sobre C++ puesto que ambos son lenguajes de programación que requieren un menor esfuerzo a la hora de expandir el proyecto que C++, aunque es un lenguaje de muy alto nivel.
3. **Calidad de los gráficos:** en este apartado se debe escoger *Unreal Engine* por encima de *Unity* puesto que es una herramienta cuyas características principales tratan de optimizar el resultado gráfico. Sin embargo, al ser un proyecto basado en realidad aumentada, la fidelidad gráfica será más sencilla de lograr mediante un buen modelado y texturizado puesto a que los hologramas no permitirán una visión perfecta de los detalles de los objetos, aunque sí permitirá una visión fiel.

Unreal Engine es un motor que destaca por sus resultados en el apartado gráfico, sin embargo, es un motor con el que no contamos la experiencia necesaria para sacar su máximo partido. Teniendo en cuenta que los resultados gráficos no son una de nuestras prioridades sino más bien la experiencia y la interacción, se ha decidido que el motor que debe usarse es *Unity* ya que se cuenta con experiencia previa que facilitará el aprendizaje y el proceso de familiarización con el SDK que se usará.

1.7.3 Solución propuesta para entorno de desarrollo (IDE)

La decisión propuesta para el entorno de desarrollo ha sido *Visual Studio*. Es una decisión fácil de tomar al tener en cuenta que vamos a desarrollar el proyecto en *Unity* puesto que *Visual Studio* cuenta con una variedad de extensiones diseñadas específicamente para apoyar al desarrollador en el uso de *Unity*. Además, a través de *Visual Studio* es posible compilar las aplicaciones de *Unity* e instalar la aplicación en el dispositivo *HoloLens 2* pulsando un solo botón.

1.8 Alcance

A la finalización del proyecto se buscará haber obtenido los siguientes objetivos:

1. **Proyecto en Unity:** proyecto con el contenido de la aplicación en Unity para la versión 2020.3.48f. Este proyecto consistirá en una escena donde cargaremos los diferentes elementos con los que interactuaremos e incluirá las funcionalidades que describirán en los apartados siguientes.
2. **Documentación:** el presente documento hará las veces de documentación, siendo este una memoria del proyecto.
3. **Aplicación:** se compilará el proyecto para las *Microsoft HoloLens 2*, de manera que no sea necesaria la ejecución de la aplicación mediante *Unity.* Aunque si se quiere añadir material adicional como modelos 3D, será necesario hacerlo usando *Unity.*

1.9 Tecnologías utilizadas

- **Unity:** el motor gráfico escogido para la programación de la aplicación.
- **Visual Studio:** el IDE que usaremos para crear los scripts de Unity en C# y para compilar el proyecto.
- **Blender:** se usará esta herramienta para requisitos puntuales derivados del desarrollo de la aplicación.
- **Git:** se hará uso de Git como herramienta de control de versiones con el fin de sincronizar el proyecto dentro y fuera del laboratorio.
- **GitBash:** se ha escogido esta interfaz para el uso de Git.
- **Mixed Reality Toolkit:** conjunto de librerías para el desarrollo de aplicaciones en MR.
- **Microsoft Word:** herramienta utilizada para la creación del presente documento.
- **NormalMap-Online:** una web que permite la creación de mapas de normales [25].
- **Jotform:** aplicación online para la creación de formularios, en este caso para la creación de encuestas de satisfacción.
- **GitMind:** aplicación utilizada para la creación de diferentes tipos de diagramas como los de casos de uso.
- **Canvas:** aplicación online utilizada para la creación de diagramas como el diagrama de Gantt.
- **Flaticon:** página web que ofrece iconos para aplicaciones de manera gratuita en diferentes formatos, utilizada para los iconos de los botones de algunas de las interfaces. [26]

1.10 Metodología de desarrollo de software

La metodología de desarrollo de software [27] se refiere al conjunto de técnicas y métodos que se utilizan para diseñar una solución de software informática. Existen varias metodologías, de modo que cada equipo puede escoger una en función de sus necesidades o de los requisitos del proyecto que se quiere desarrollar. Trabajar con una metodología es imprescindible por una cuestión de organización. Por otra parte, las metodologías sirven para controlar el desarrollo del trabajo, lo cual promueve la minimización de los márgenes de errores y la anticipación de estos.

Puesto que no se cuenta con especificaciones funcionales más allá de experimentar con las HoloLens2 y crear un prototipo para uso en contexto docente, además de que contamos con un tiempo reducido a 300h para el proyecto, necesitaremos un enfoque que nos permite interactuar con la herramienta a medida que se desarrolla la misma y que permita adaptarse a los cambios de requisitos. Es por esto por lo que se ha decidido llevar a cabo una metodología de prototipado evolutivo.

1.10.1 Prototipado evolutivo

El prototipado evolutivo nos permite adaptarnos a los cambios que frecuentemente surgen en los avances en el desarrollo del producto. Este hecho hace que no sea realista trazar una línea recta entre el inicio y el producto final.

El modelo de prototipos permite que todo el sistema, o alguna de sus partes, se construyan rápidamente para comprender o aclarar aspectos. Los requerimientos o el diseño requieren la investigación repetida para asegurar que el desarrollador, el usuario, y el cliente tengan una comprensión unificada tanto de lo que se necesita como de lo que se propone como solución. [28]

En el caso del prototipado evolutivo, distinguimos una serie de etapas:

- **Planificación:** en esta etapa se planifica la gestión de desarrollo tanto cronológicamente como lo referente al consumo de recursos.
- **Implementación:** en esta etapa se realizarán las actividades que conlleva el desarrollo para la realización del proyecto.

- **Puesta en producción:** en esta etapa se presenta al usuario final y al cliente después de que se ha logrado completar el proceso de realización y que responde a los requerimientos solicitados por el cliente o usuario final.

Estas etapas pueden subdividirse de la manera presentada en la Ilustración 11:

1. **Requisitos del sistema y especificaciones del prototipo:** se recopilan los requisitos del cliente y se establecen los objetivos del proyecto. Esto implica comprender las necesidades y expectativas de los usuarios finales y definir los criterios de éxito del producto.
2. **Diseño del prototipo:** se desarrollan conceptos y se generan ideas para abordar los requisitos identificados en la etapa anterior. Se pueden crear bocetos, diagramas de flujo, mapas de usuario y otros artefactos para visualizar y comunicar las ideas propuestas.
3. **Implementación y prueba del prototipo:** se crea un prototipo inicial basado en los conceptos y diseños generados en la etapa anterior. Este prototipo suele ser de baja fidelidad y se centra en demostrar las funcionalidades clave del producto de una manera rápida y económica. Posteriormente se somete a prueba por los usuarios interesados. Lo cual ayuda a identificar áreas de mejora y validar el producto.
4. **Evaluación y comunicación para refinamiento:** con base en los comentarios recibidos durante las pruebas, se realizan ajustes y mejoras en el prototipo. Este proceso de iteración puede implicar la creación de múltiples versiones del prototipo a medida que se van abordando las preocupaciones y se van incorporando nuevas ideas. A medida que avanza el proceso de iteración, los prototipos se van refinando y aumentando su fidelidad. Esto puede implicar agregar más detalles visuales, funcionalidades adicionales o incluso desarrollar prototipos funcionales que se asemejen más al producto final.
5. **Diseño final:** una vez que se ha alcanzado un nivel de refinamiento satisfactorio, se crea una versión final del prototipo. Esta es la primera etapa para avanzar a la fase de desarrollo completo.

6. **Implementación del sistema final:** una vez implementado el prototipo diseñado en la etapa anterior se realiza una validación final. Esta fase nos asegura que cumplimos con los requisitos del cliente o usuario final para comenzar el desarrollo de la aplicación al completo.

El prototipado evolutivo se caracteriza por ser un proceso cíclico, por lo que estas etapas se repiten varias veces a lo largo del desarrollo hasta que se logra un diseño final satisfactorio.

Se adaptará esta metodología para el desarrollo de la aplicación, teniendo en cuenta la restricción de tiempo para el desarrollo del prototipo.

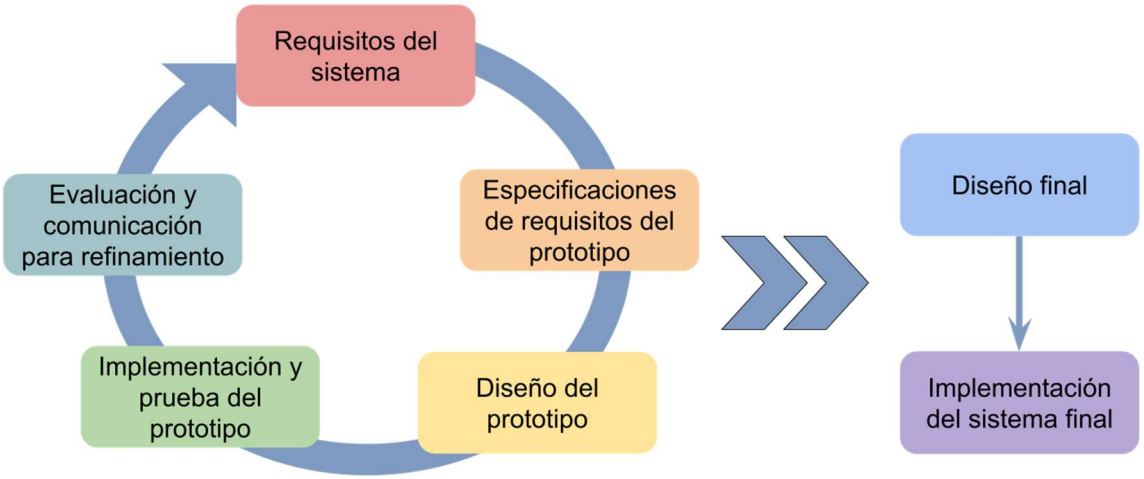


Ilustración 11: Ciclos de prototipado evolutivo

1.11 Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFT, no existen restricciones de tipo económico, sino de tipo temporal (un número aproximado de horas). Por consiguiente, los cálculos de tamaño del proyecto están supeditados el tiempo disponible. En cuanto al esfuerzo, se dispone de tan solo un efectivo (la persona autora del trabajo).

Siendo coherentes con la metodología seleccionada para nuestros fines y basándonos en el número de horas que tenemos disponibles para la realización del proyecto realizaremos la estimación y la valoración como si se tratase de una iguala [29]. Una iguala es un acuerdo en el cual se establece un monto fijo mensual o una tarifa plana para el desarrollo de un proyecto. En este caso, hemos acordado un número fijo de horas de trabajo por semana y una tarifa por hora, que nos permite estimar el esfuerzo total requerido.

Siendo así tenemos las siguientes restricciones:

- Horas de trabajo por semana: 25 horas
- Días de trabajo por semana: 5 (lunes a viernes)
- Total de horas contratadas: 300 horas
- Tarifa por hora: 14,62 €/hora (más detalles en el apartado 1.13)

Para el desglose de las actividades realizadas se ha realizado la siguiente estimación:

Actividad	Tiempo estimado (horas)
Investigación inicial	50
Instanciación de muros	40
Modelado de entornos	60
Interfaz de construcción	40
Reconocimiento de gestos	40
Interfaz de dibujo	30
Reconstrucción virtual	30

Modos de dibujo	20
Creación de documentación	20
TOTAL	**300**

Tabla 4: Estimación de horas por actividad

La estimación de esfuerzo para el desarrollo del TFG se ha realizado utilizando una iguala de 300 horas de trabajo, distribuidas en jornadas de 5 horas diarias durante 12 semanas. Una vez completadas las horas contratadas, se presentará el prototipo siendo los elementos restantes por desarrollar parte de trabajos futuros.

1.12 Planificación temporal

El planteamiento del presente proyecto se produjo en noviembre. Exceptuando los periodos de exámenes, se estima que el proyecto junto con la documentación estará listo en junio, lo que nos proporciona un periodo de 6 meses para llevar a cabo el proyecto.

En la Ilustración 12 se presenta el diagrama de Gantt con la previsión temporal. Las actividades se distribuyen según las diferentes iteraciones identificadas en el desarrollo evolutivo del prototipo final. Dado que seguimos una metodología ágil, se dedica un apartado específico a la fase de desarrollo del proyecto y otro a la fase de documentación. No se incluye un apartado separado para el diseño o el planteamiento inicial, ya que, al ser un proceso iterativo, este se realiza al comienzo de cada iteración y se integra en el tiempo asignado a las primeras actividades. Esta metodología permite ajustar y refinar continuamente el diseño a lo largo del desarrollo, asegurando que el proyecto evoluciona de manera coherente y eficiente.

Ilustración 12: Diagrama de Gantt

Dada la naturaleza ágil de la metodología utilizada el diagrama se ha actualizado en la etapa final del proyecto.

1.13 Presupuesto

Para el presente apartado se aportan las siguientes tablas donde podremos observar el desglose de coste para material y personal. Asumiendo entonces un periodo de amortización de unos 5 años, tenemos el siguiente desglose:

Material	Precio (€)	Amortización parcial (€/mes)	Tiempo de uso (meses)	Coste (€)
Portátil ASUS	954,75	15,91	6	95,46
Ratón MX máster	90,08	1,50	6	9
Windows 11 Home	144,99	2,42	6	14,52
Microsoft HoloLens 2	3.849,00	64,15	6	384,90
MRTK	0	0	4	0

Unity 3D	0	0	5	0
Visual Studio 2022	0	0	5	0
Blender	0	0	1	0
GitBash	0	0	5	0
TOTAL				**503,88**

Tabla 5: Costes de material

En la Tabla 5 se puede comprobar la totalidad del coste de los materiales habiendo tenido en cuenta una amortización parcial con un periodo de 5 años.

Para concluir el apartado se proporciona a continuación la tabla con el coste d personal, teniendo en cuenta el número de horas trabajadas, el personal implicado y costes derivados del sueldo y seguridad social. Para el cálculo de estos elementos se han tenido en cuenta sueldos promedio de puestos relacionados en la comunidad de Andalucía. [30]

Tarea	Puesto	Horas	Salario (€/hora)	S.S. (€)	Coste (€)
Análisis	Analista	25	16,15	121,13	524,88
Investigación	Programador	40	14,62	175,44	760,24
Programación	Programador	195	14,62	855,27	3.708,17
Documentación	Programador	30	14,62	131,58	570,18
Validación	Tester	10	14,10	42,3	183,30
TOTAL					**5.746,77**

Tabla 6: Costes de personal

Ya que las tareas de modelado no son necesarias o quedan en una cuantía mínima se incluyen en el tiempo de programación y no se requiere de un modelador. Los efectos de sonido utilizados vienen proporcionados por el SDK así que esos gastos se pueden ignorar. Para terminar el presupuesto añadimos unos costes indirectos del 10%.

Gasto	Coste (€)
Gastos en material	503,88
Gastos en personal	5.746,77
Gastos indirectos	625,07
TOTAL	**6.875,72**

Tabla 7: Tabla de costes total

Concluimos en que el precio total del proyecto es de **6.875,72 €**.

2 DISEÑO INICIAL

En el presente apartado se especificarán los requisitos iniciales de diseño para el prototipo. El objetivo es producir una idea general sobre la que iremos añadiendo cambios a medida que se vayan sucediendo las iteraciones.

2.1 Especificaciones del sistema

En este apartado se presentará una especificación de los requisitos iniciales del proyecto a raíz de la primera especificación de requisitos, que forma parte de la primera iteración. Sin embargo, para facilitar el seguimiento del documento se presentarán todos los requisitos como si se hubieran pasado por todas las iteraciones. En los apartados posteriores, profundizaremos en la consecución de los requisitos definidos en el presente apartado.

2.1.1 Requisitos funcionales

Definimos los requisitos funcionales como aquellos que definen los servicios que presenta el sistema. Estos requisitos pueden tener entradas de los usuarios (interacciones) o de otros sistemas, como por ejemplo procesos predefinidos o respuestas automáticas. Definen “lo que hace” el sistema. A continuación, se enumeran los requisitos:

- RF-01: Se debe poder colocar muros de manera que se solapen con el modelo para su visualización.

- RF-02: Se deben aportar las suficientes herramientas para que la interacción con el entorno sea transparente y cómoda.
- RF-03: Se debe poder transformar los objetos colocados de maneras intuitivas y vistosas.
- RF-04: Se debe implementar una herramienta de anotaciones, de manera que sea posible plasmar dibujos sobre la escena o sobre el modelo presentado.
- RF-05: Se debe poder manipular algún objeto o modelo importado para si visualización.

2.1.2 Requisitos no funcionales

Definimos los requisitos no funcionales como aquellos que no definen un servicio o funcionalidad, sino que define las propiedades del sistema, como la velocidad, sencillez, seguridad, disponibilidad, etc. Definen “cómo lo hace” el sistema. A continuación, se enumeran los requisitos:

- RNF-01: simplificar la interacción para minimizar el esfuerzo de aprendizaje de la herramienta.
- RNF-02: garantizar un buen rendimiento para prevenir mareos o incomodidad con el uso de las gafas
- RNF-03: minimizar el uso de interfaces y priorizar el manejo con gestos para agilizar el uso de la herramienta, siendo esta un recurso didáctico.

2.2 Análisis y diseño del sistema

Se presente en este apartado el análisis y los diseños de los sistemas a implementar para el prototipo.

2.2.1 Diagrama de casos de uso

Un diagrama de casos de uso es de gran utilidad en el análisis y diseño de sistemas ya que ayudan a capturar los requisitos funcionales del sistema a desarrollar.

Se distinguen en el diagrama las relaciones ***include*** y ***extend***, refiriéndose a elementos necesarios para el funcionamiento y añadidos opcionales a disposición del usuario respectivamente.

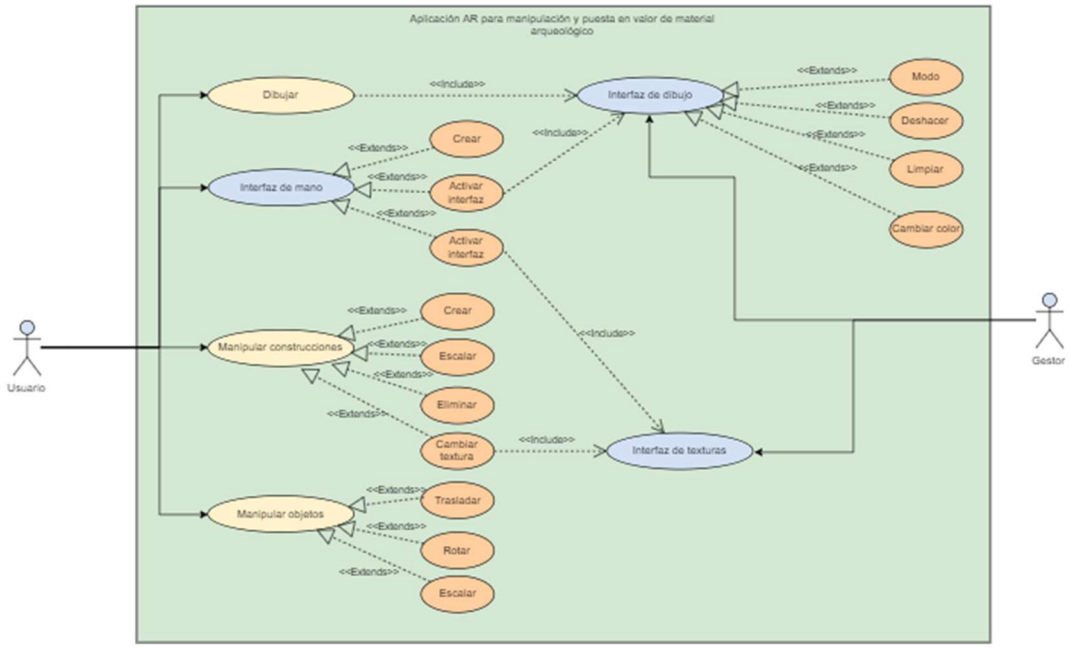


Ilustración 13: Diagrama de casos de uso

2.2.2 Casos de uso

En el presente apartado se desarrollarán los casos de usos presentes en Ilustración 13. Haremos uso de una tabla que parte del siguiente formato:

Id: Nombre	
Nombre	Nombre del caso de uso
Actor	Actor que dirige el caso de uso
Pre-condición	Condición que debe darse previamente al inicio del caso de uso
Flujo normal	1. Flujo de sucesos común 2. Flujo de sucesos común 3. Flujo de sucesos común 4. …
Flujos alternativos	1. Flujo de sucesos alternativo 2. Flujo de sucesos alternativo 3. Flujo de sucesos alternativo 4. …

Tabla 8: Plantilla de casos de uso

Partiendo de la plantilla ya definida, se procede a la descripción de los casos de uso:

CU-01: Manipular construcciones	
Nombre	Manipular construcciones
Actor	Usuario
Pre-condición	Ninguna
Flujo normal	1. El usuario realiza los gestos correspondientes a la manipulación. 2. El sistema muestra los cambios producidos en las construcciones.

Tabla 9: Caso de uso - 01

CU-02: Dibujar	
Nombre	Dibujar
Actor	Usuario
Pre-condición	La interfaz de dibujo debe estar activa
Flujo normal	1. El usuario realiza el gesto para sacar la interfaz de mano. 2. El usuario activa la interfaz de dibujo. 3. El usuario configura la interfaz para poder dibujar. 4. El usuario selecciona un modo de dibujo. 5. El usuario dibuja.

Tabla 10: Caso de uso - 02

CU-03: Interfaz de mano	
Nombre	Interfaz de mano
Actor	Usuario
Pre-condición	Ninguna
Flujo normal	1. El usuario realiza el gesto para sacar la interfaz de mano. 2. El usuario interactúa con la interfaz.

Tabla 11: Caso de uso – 03

CU-04: Crear	
Nombre	Crear
Actor	Usuario
Pre-condición	La interfaz de mano debe estar presente
Flujo normal	1. El usuario realiza el gesto para sacar la interfaz de mano. 2. El usuario pulsa el botón correspondiente a “crear”. 3. Se instancia un objeto transformable delante del usuario.

Tabla 12: Caso de uso – 04

CU-05: Activar/Desactivar interfaz de dibujo	
Nombre	Activar/Desactivar interfaz de dibujo
Actor	Usuario
Pre-condición	La interfaz de mano debe estar presente
Flujo normal	1. El usuario realiza el gesto para sacar la interfaz de mano. 2. El usuario pulsa el botón a la acción. 3. La interfaz de dibujo se activa o desactiva en función de su estado.

Tabla 13: Caso de uso – 05

CU-06: Activar/Desactivar interfaz de texturas	
Nombre	Activar/Desactivar interfaz de texturas
Actor	Usuario
Pre-condición	La interfaz de mano debe estar presente
Flujo normal	1. El usuario realiza el gesto para sacar la interfaz de mano. 2. El usuario pulsa el botón a la acción. 3. La interfaz de texturas se activa o desactiva en función de su estado.

Tabla 14: Caso de uso - 06

CU-07: Manipular objetos	
Nombre	Manipular objetos
Actor	Usuario
Pre-condición	Ninguna
Flujo normal	1. El usuario puede realizar transformaciones básicas sobre el objeto usando sus manos. 2. El usuario finaliza las transformaciones soltando el objeto.

Tabla 15: Caso de uso – 07

CU-08: Trasladar	
Nombre	Trasladar
Actor	Usuario
Pre-condición	El Usuario debe manipular un objeto
Flujo normal	1. El usuario traslada un objeto realizando movimientos con las manos. 2. El usuario finaliza la interacción con el objeto soltándolo.

Tabla 16: Caso de uso – 08

CU-09: Rotar	
Nombre	Rotar
Actor	Usuario
Pre-condición	El Usuario debe manipular un objeto
Flujo normal	1. El usuario rota un objeto realizando rotaciones con las manos. 2. El usuario finaliza la interacción con el objeto soltándolo.

Tabla 17: Caso de uso – 09

CU-10: Escalar	
Nombre	Escalar
Actor	Usuario
Pre-condición	El Usuario debe manipular un objeto
Flujo normal	1. El usuario escala un objeto agarrando uno de los mangos de escalado predefinidos. 2. El usuario finaliza la interacción con el objeto soltándolo.

Tabla 18: Caso de uso - 10

CU-11: Manipular construcciones	
Nombre	Manipular construcciones
Actor	Usuario
Pre-condición	Ninguna
Flujo normal	1. El usuario realiza gestos para interactuar con las construcciones.

Tabla 19: Caso de uso - 11

CU-12: Crear construcción	
Nombre	Crear construcción
Actor	Usuario
Pre-condición	El Usuario debe manipular una construcción
Flujo normal	1. El usuario define con gestos una línea sobre la que crear una construcción. 2. El sistema instancia la construcción sobre la línea.

Tabla 20: Caso de uso – 12

CU-13: Escalar construcción	
Nombre	Escalar construcción
Actor	Usuario
Pre-condición	Debe haberse creado una construcción
Flujo normal	1. El usuario utiliza los mangos de escalado situados sobre las construcciones para escalar el objeto. 2. El usuario suelta el mango para finalizar el escalado.

Tabla 21: Caso de uso - 13

CU-14: Eliminar construcción	
Nombre	Escalar construcción
Actor	Usuario
Pre-condición	Debe haberse creado una construcción
Flujo normal	1. El usuario señala una construcción. 2. El usuario realiza el gesto para eliminar la construcción. 3. La construcción es eliminada.

Tabla 22: Caso de uso – 14

CU-15: Cambiar textura	
Nombre	Cambiar textura
Actor	Usuario
Pre-condición	Debe activarse la interfaz de texturas
Flujo normal	1. El usuario usa la interfaz de mano para activar la interfaz de texturas. 2. El usuario selecciona la textura deseada.
Flujo alternativo	1. 2A El usuario altera el componente alfa de la textura transparente usando el componente de la interfaz de texturas slider.

Tabla 23: Caso de uso – 15

CU-16: Cambiar modo de dibujo	
Nombre	Cambiar modo de dibujo
Actor	Usuario
Pre-condición	Debe activarse la interfaz de dibujo
Flujo normal	1. El usuario activa la interfaz de dibujo. 2. El usuario pulsa el botón “modo”. 3. El modo de dibujo cambia al siguiente modo implementado.

Tabla 24: Caso de uso – 16

CU-17: Deshacer dibujo	
Nombre	Deshacer dibujo
Actor	Usuario
Pre-condición	Debe activarse la interfaz de dibujo
Flujo normal	1. El usuario activa la interfaz de dibujo. 2. El usuario pulsa el botón “deshacer”. 3. El gestor deshace el último dibujo realizado.

Tabla 25: Caso de uso – 17

CU-18: Limpiar dibujos	
Nombre	Limpiar dibujos
Actor	Usuario
Pre-condición	Debe activarse la interfaz de dibujo
Flujo normal	1. El usuario activa la interfaz de dibujo. 2. El usuario pulsa el botón “limpiar”. 3. El gestor elimina todos los dibujos realizados.

Tabla 26: Caso de uso – 18

CU-19: Cambiar color	
Nombre	Cambiar color
Actor	Usuario
Pre-condición	Debe activarse la interfaz de dibujo
Flujo normal	1. El usuario activa la interfaz de dibujo. 2. El usuario pulsa el botón del color deseado. 3. El gestor cambia el color de los dibujos futuros al color seleccionado por el Usuario.

Tabla 27: Caso de uso – 19

2.2.3 Arquitectura del sistema

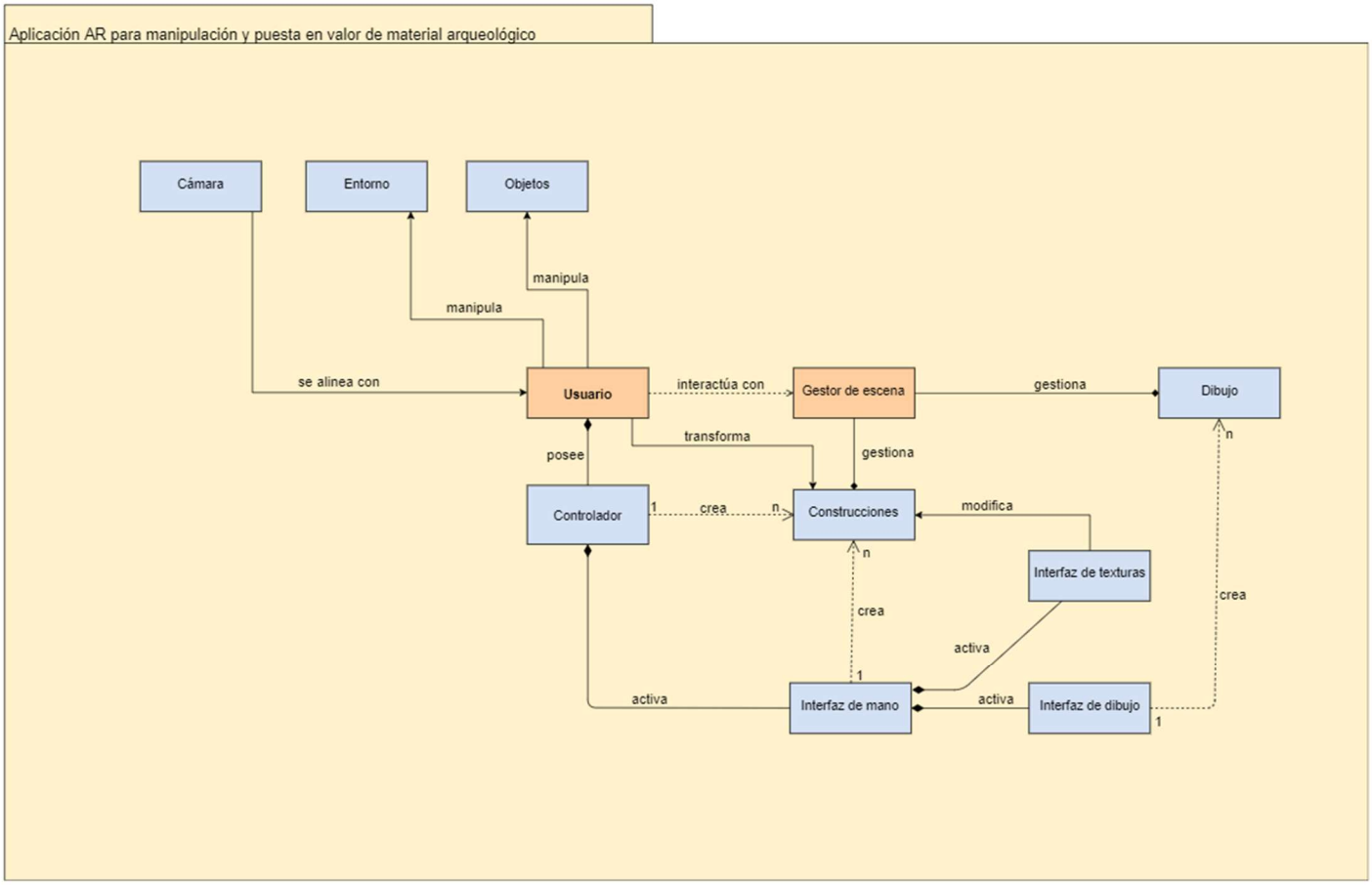


Ilustración 14: Diagrama UML de Arquitectura del Sistema

Se ha destacado en rojo los elementos que funcionarán como ejes para el proyecto y en azul las clases que se pretenderán implementar para la consecución del prototipo.

3 DESARROLLO

Para el siguiente apartado es necesario recordar que la metodología que se ha llevado a cabo, ya explicada en el apartado 1.10, consiste en un prototipado evolutivo el cual se basa en una serie de iteraciones, aportando mejoras a funcionalidades o aplicando nuevas funcionalidades en cada iteración.

En el presente apartado se explicará con mayor detalle las iteraciones implementadas.

3.1 Primera Iteración

La primera iteración ha sido una toma de contacto con las primeras funcionalidades que se buscan desarrollan y parten de la primera idea sacada de la primera reunión con los tutores del proyecto. Los elementos desarrollados en esta iteración han sentado un precedente para ciertas bases como las interfaces y las características de las edificaciones, además de la creación de la clase de gestión para algunas de las funcionalidades de la aplicación.

Se separará el contenido en subsecciones para facilitar su revisión.

3.1.1 Entorno

A raíz de la presente ilustración de la primera versión se procederá a detallar los elementos descritos:

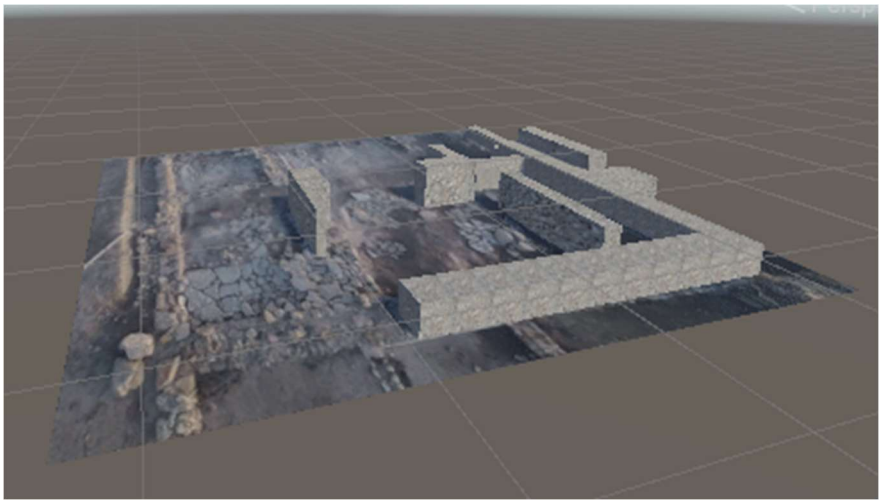


Ilustración 15: Primera iteración. Construcción de edificios

Para las primeras ideas se tiene como base una fotografía real realizada en el santuario de puente tablas. La idea era comprobar la visualización en realidad aumentada de los distintos elementos configurados de la manera que se presenta en la ilustración.

Se ha diseñado en esta misma iteración una interfaz interactiva, la cual se puede activar y desactivar con un movimiento de supinación de la muñeca. Esta interfaz seguirá la posición de la muñeca, dando una percepción certera de la posición en el espacio de la interfaz, sin que la misma moleste a la hora de interactuar con el entorno. Además, es posible mover y colocar esta interfaz en cualquier lugar del espacio, bloqueando su posición al terminar de colocarse. Esta funcionalidad se encuentra explicada en más detalle en el apartado 3.3.2.2.

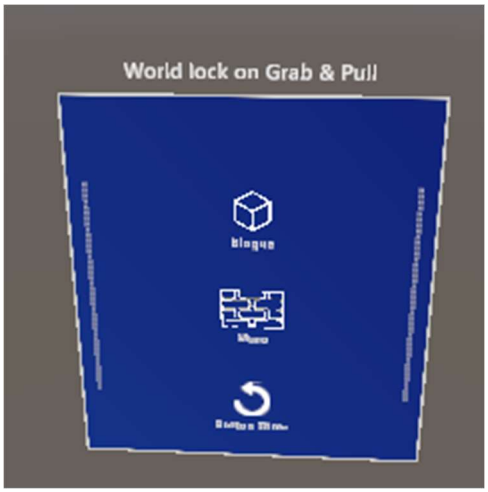


Ilustración 16: Primera iteración. Interfaz de mano

La interfaz presenta 3 opciones en esta versión:

1. ***Bloque*:** se instancia un cubo de tamaño predeterminado, este se escala de manera uniforme.
2. ***Muro:*** se instancia un objeto en forma de prisma rectangular, este objeto se escala de manera no uniforme.
3. ***Reiniciar escena:*** este elemento se ha eliminado de la interfaz final, su funcionalidad ha sido de utilidad a la hora de realizar pruebas de usabilidad y depuración de la aplicación.

Instanciar los objetos se vuelve una tarea sencilla con la implementación del patrón de diseño *Factory*, que nos permite crear objetos bajo demanda y sustituir el constructor principal de estos objetos. Sin embargo, tras una evaluación detenida, se ha concluido que la configuración de estos objetos, al usar los componentes del SDK, requeriría muchas líneas de código y sería muy poco flexible en caso de necesitar realizar cambios. Por lo tanto, por el momento no es conveniente aplicar este patrón.

En la siguiente ilustración se muestra el escalado de las construcciones mediante gestos.

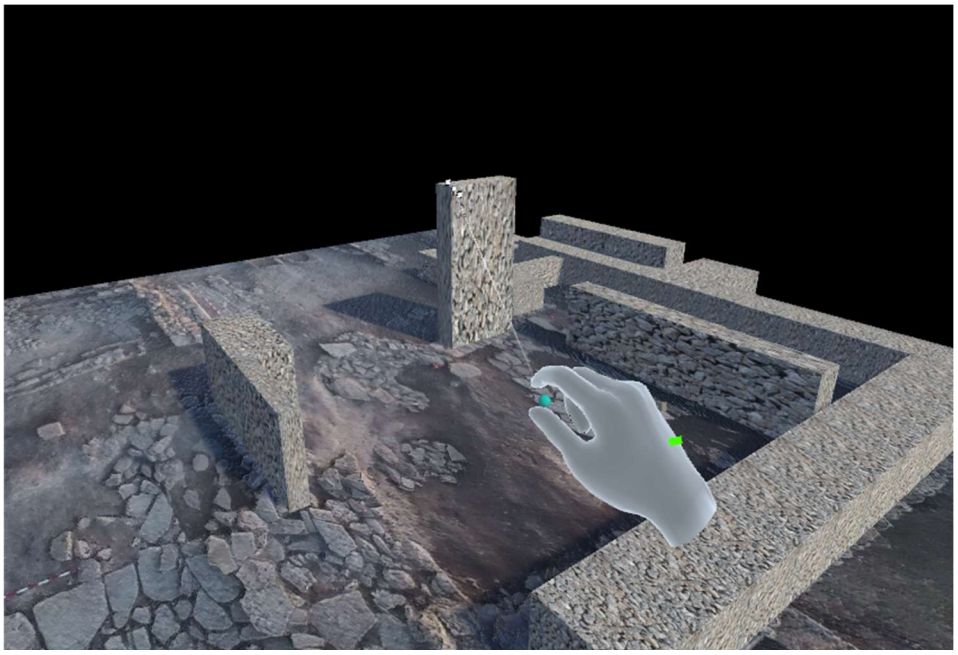


Ilustración 17: Primera iteración. Manipulación de edificios

Como se puede observar simplemente se requiere realizar un gesto que imite al agarre de un objeto mientras apuntamos a una esquina del objeto, después simplemente se arrastra el objeto como si se estuviera moldeando. De la misma manera se puede escalar usando dos manos. En esta versión es posible a su vez rotar el objeto y desplazarlo utilizando los mismos gestos y realizando las transformaciones como si se trataran de objetos físicos reales.

3.1.2 Gestión de la escena

Para la gestión general de los objetos que el jugador incluirá a la escena mientras use la aplicación se ha creado un objeto utilizando el patrón de diseño *Singleton*, ya que en esta temprana etapa del desarrollo se desconoce la totalidad del uso que se le dará a este componente y es conveniente adecuar su diseño para adaptarse a la escalabilidad que sin duda se requerirá.

De momento lo único que se incluye en este componente son funciones para incorporar la funcionalidad correspondiente a los botones de la interfaz mostrada en la Ilustración 16. Además, para apoyar la escalabilidad de la aplicación, se ha definido un diccionario que registra diferentes objetos, los cuales entrarán dentro de la categoría “Construcciones”. Estos se refieren a los objetos que el usuario de la

aplicación podrá añadir a la escena mediante el uso de gestos, la detección de estos gestos se explicará más adelante.

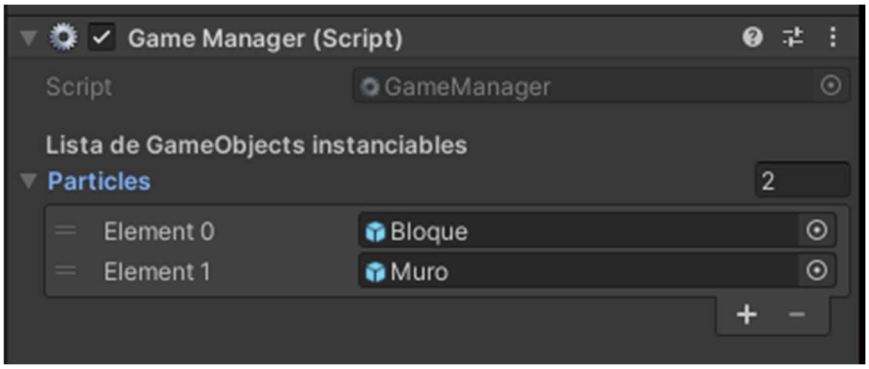


Ilustración 18: Primera iteración. Gestor de escena GameManager

En la imagen se muestra que es posible agregar nuevos objetos instanciables al asignarles una posición en la lista que aparece en el componente. Luego, un script sencillo los añade a un diccionario público, lo que permite acceder al objeto original para copiarlo cuando se solicite.

3.1.3 Validación

Es evidente que la representación del espacio deja mucho que desear sobre otras opciones. No se puede apreciar en gran medida los detalles de los entornos que se propongan si partimos de una fotografía, además en un entorno de Realidad Aumentada la diferencia de calidad es bastante más notoria.

Sin embargo, la colocación y la transformación de las construcciones en esta versión responde bien a los movimientos del usuario, aunque resulta poco satisfactorio tener que usar la interfaz de mano para instanciar cada construcción.

3.2 Segunda iteración

3.2.1 Entorno

Tras revisar la iteración previa, se ha concluido que construir objetos sobre una imagen plana no mejora la inmersión del usuario ni aprovecha completamente las capacidades de las HoloLens 2. Por ello, se ha decidido reemplazar la imagen plana con un modelo 3D del lugar requerido. Evidentemente, el uso de este recurso exige un trabajo previo de escaneado, modelado y texturizado del lugar en cuestión, aunque esa labor no forma parte de este proyecto.

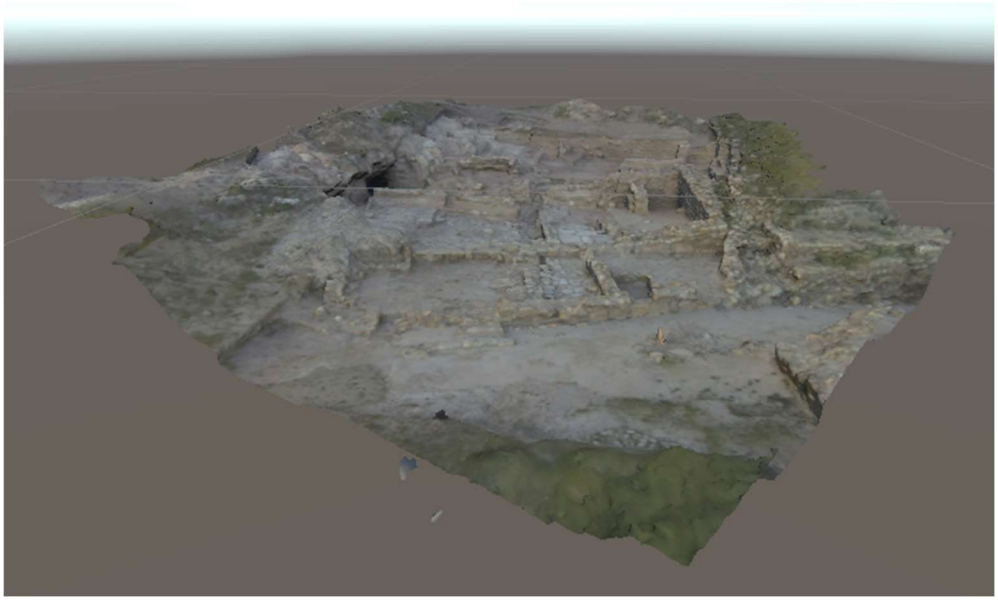


Ilustración 19: Segunda iteración. Modelo 3D incluido

En concreto este modelo pertenece al Santuario de la Puerta del Sol, situado en Puente Tablas, Jaén [31].

3.2.2 Mejoras en construcciones

Además de la inclusión del modelo se ha mejorado la interacción con las Construcciones, añadiendo un marco con agarres a cada una ajustándola a su colisionador de Unity [32], de manera que al colocar nuestro controlador sobre la construcción aparezca para dar un apoyo visual al usuario sobre los límites del objeto

y que proporcione un método de interacción con las transformaciones que se pueden aplicar sobre el objeto.

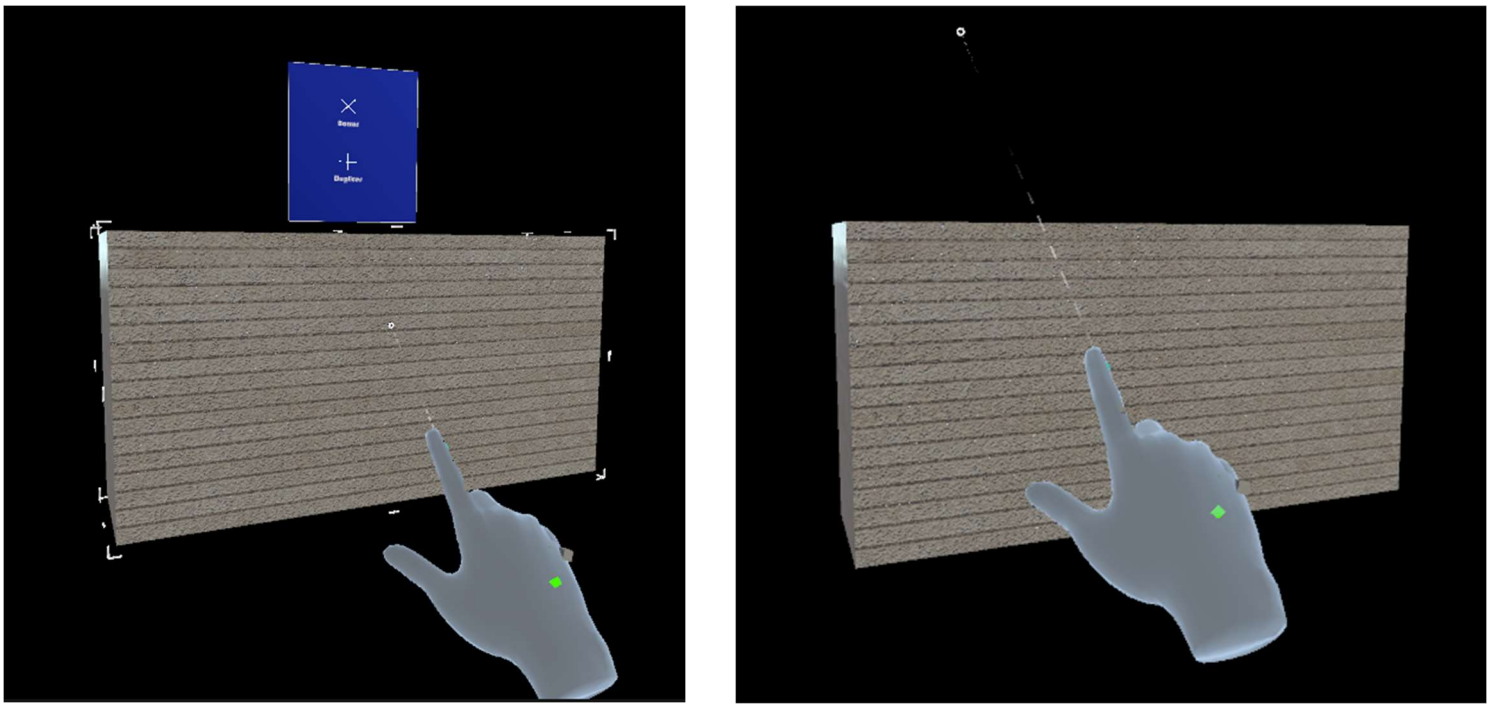


Ilustración 20: Segunda iteración. Interacción con construcciones

En la primera imagen se puede apreciar los nuevos elementos añadidos con el fin de aportar más claridad a la interacción con la construcción. En la segunda imagen estos elementos desaparecen ya que no estamos apuntando a la construcción con el controlador. También se ha añadido una pequeña interfaz con funcionalidades como borrar la construcción o duplicarla con sus mismas características y una nueva textura, distinta a la que se podía apreciar en la Ilustración 15.

En esta iteración se ha trabajado con algunos scripts auxiliares que aportan funcionalidades secundarias con el objetivo de mejorar la experiencia del usuario usando la aplicación tales como hacer que las interfaces miren a cámara, prevenir que las construcciones pudieran tener valores de escala negativos o bloquear las transformaciones para ciertos objetos. Sin embargo, estos scripts quedarán obsoletos con la incorporación de componentes proporcionados por el SDK, que no solo implementan estas funcionalidades, sino que también añaden otras mejoras para una experiencia de uso más agradable.

3.2.3 Validación

Los cambios introducidos al entorno prueban ser mucho más satisfactorios que en la primera versión. El modelo 3D del entorno es mucho más apreciable que las imágenes planas y la colocación de las construcciones sobre el modelo genera mucha más sensación de realidad que de la primera forma.

Por otro lado, el añadido de la interfaz a las construcciones supone un problema a la hora de realizar algunas transformaciones ya que el escalado deforma la interfaz y puede hacer que esta sea molesta. Además, la interacción que utiliza realidad aumentada para esta labor ha resultado ser poco cómoda y eficiente para usuarios experimentados. Es por ello por lo que sería necesario implementar algún script que corrigiera el problema o simplemente buscar una manera alternativa a las interfaces visuales de interactuar con las construcciones de esta forma.

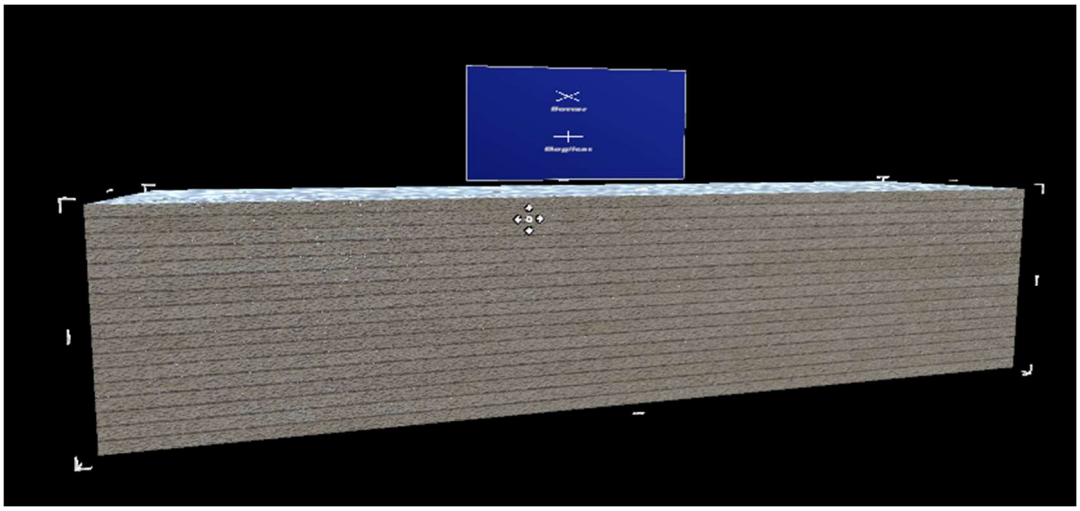


Ilustración 21: Segunda versión. Validación de interfaz

Durante la validación de esta versión se realizó una exposición a un usuario potencial y se le permitió probar la herramienta. Sugirió que la herramienta era prometedora y que le parecía una interacción adecuada, además pidió una herramienta para tomar anotaciones ya que el objetivo del proyecto es orientar el prototipo al ámbito docente para servir de apoyo en clases expositivas. La idea se valoró y se decidió implementar la herramienta en la siguiente iteración además de realizar las actualizaciones a los otros componentes.

3.3 Tercera iteración

Esta es la iteración que más tiempo ha llevado de concluir dados los objetivos que debían completarse:

1. Desarrollar una herramienta de dibujo
2. Actualizar el método de interacción con las construcciones

Ya que necesitamos crear una herramienta de dibujo, primero debemos implementar una manera de cambiar los efectos resultantes al realizar determinados gestos con las manos. Una vez tengamos esta funcionalidad resuelta podremos probar nuevas interacciones con gestos y quizá de esta manera podamos sustituir el uso de interfaces, aportando mayor agilidad al uso de la aplicación a expensas de un coste de aprendizaje ligeramente mayor.

3.3.1 Captura de gestos

La problemática de captura de gestos reside principalmente en el sistema de inputs de MRTK. Este ofrece un componente para la configuración de sistemas como los inputs, la cámara, sistemas de diagnóstico y herramientas de teletransportación para moverse por el entorno.

Nuestra preocupación para este problema es la configuración de los inputs y, para ello, MRTK ofrece una gran variedad de perfiles de interacción, como se puede ver en la siguiente ilustración.

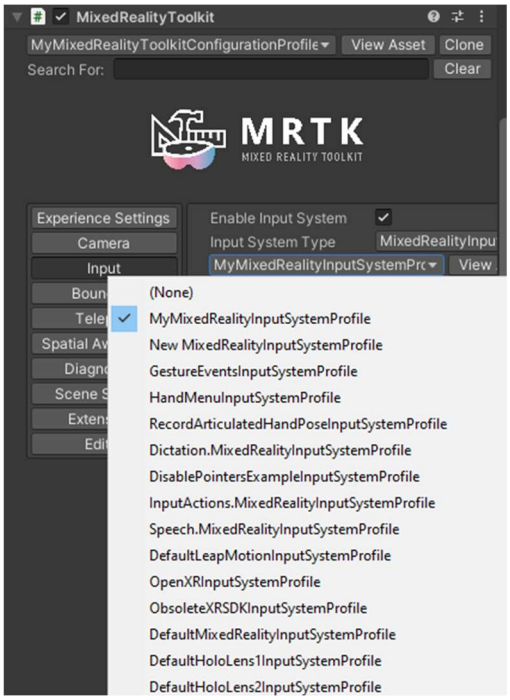


Ilustración 22: Perfiles de interacción de MRTK

Cada uno de estos sistemas ofrece un tipo de interacción, como, por ejemplo: *GestureEventsInputSystemProfile* nos ofrece interacción única y exclusivamente mediante gestos, lo cual inhabilita funcionalidades como la manipulación directa de hologramas. Puesto que requerimos de capturar ciertos gestos e incluso poder definir estos gestos, lo que nos interesa es crear un perfil propio con una configuración personalizada. También es muy útil el servicio de MRTK *HandJoint*, este servicio ofrece un seguimiento en tiempo real de nuestra mano y sus distintos huesos, falanges y articulaciones, realizando incluso distinciones entre cuál de nuestras dos manos estamos usando, lo cual nos permite diseñar gestos nosotros mismos a partir de estos datos. Este servicio se puede activar cambiando nuestro proveedor de datos de entrada en la configuración de MRTK, tal y como se muestra a continuación.

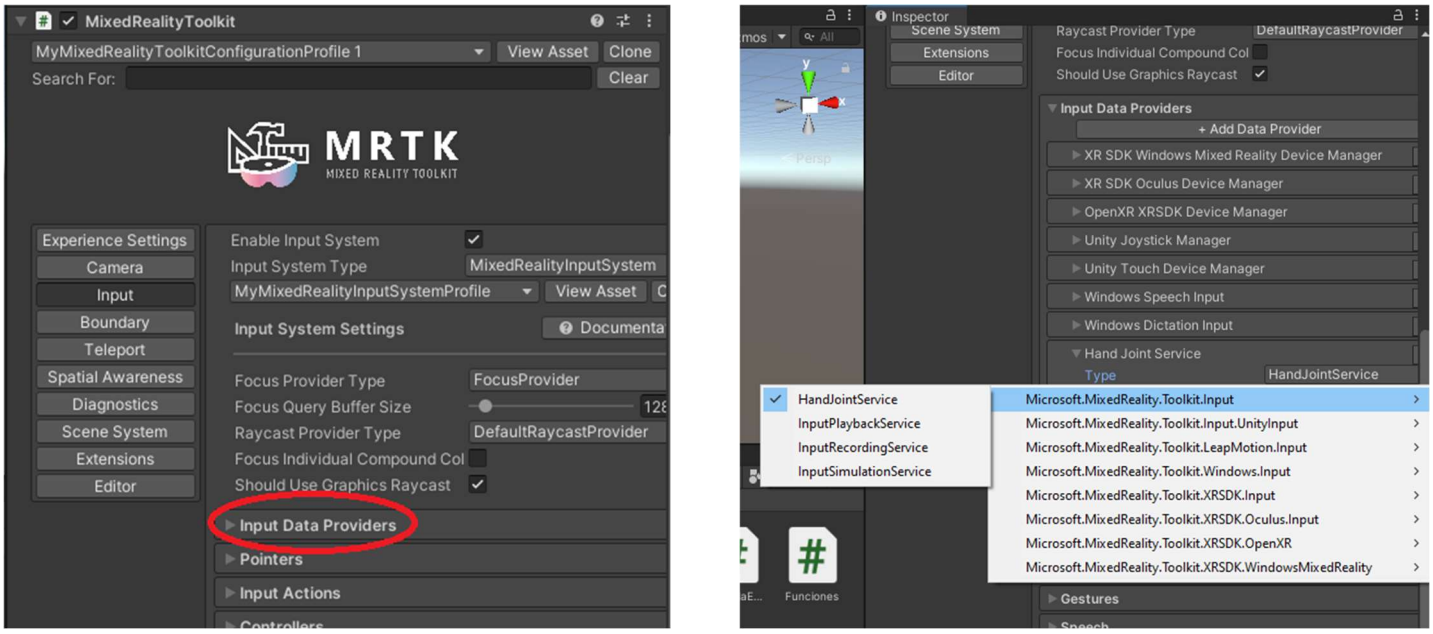


Ilustración 23: Configuración de perfil de MRTK

Una vez configurada la herramienta podemos acceder a la información sobre los controladores accediendo al tipo de nombre *TrackedHandJoint.joint,* donde *joint* es el nombre que se le da a un posicionador de la mano, el cual las gafas detectan y actualiza continuamente. [33]

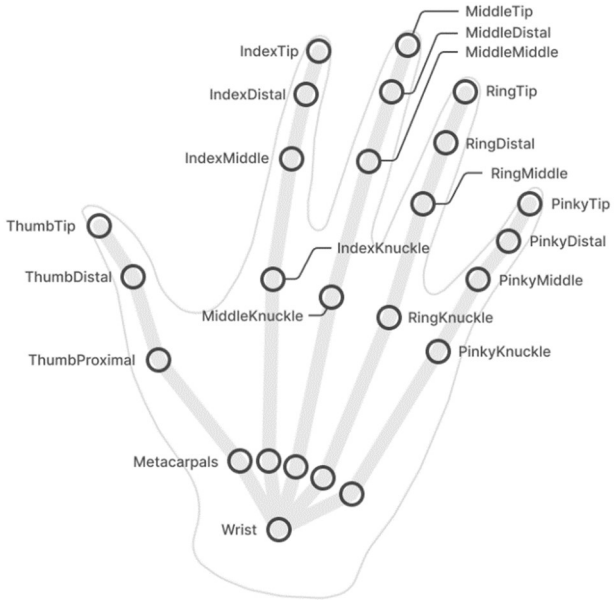


Ilustración 24: Etiquetas de los joints de la mano

Siendo este nuestra base para comenzar a definir los gestos, se explicará a continuación la metodología a nivel de script para la detección de los gestos.

```
if (HandJointUtils.TryGetJointPose(TrackedHandJoint.IndexTip, Handedness.Any,
out IndexPose))
        {
            if(HandJointUtils.TryGetJointPose(TrackedHandJoint.ThumbTip,
Handedness.Any, out ThumbPose))
            {
                //Cálculo de la distancia entre joints
                if(Vector3.Distance(IndexPose.Position, ThumbPose.Position) <=
distanciaMargen)
                {
                   //En caso de que se cumplan las condiciones del gesto
                    detectado = true;

                }
                else
                {
                   //En caso de que no se cumplan las condiciones del gesto
                   if(detectado)

                }
            }
          }
```

Ilustración 25: Código de detección de gestos

Este es el esquema que se ha seguido para la detección de un gesto. Se basa en un simple cálculo de distancia entre joints, cuando se detectan los joints especificados en los dos primeros *if,* entonces se pasa a calcular la distancia entre estos joints. Si cumple con un margen que se pasa por parámetro, se habrá detectado el gesto y podremos llevar a cabo cualquier operación que se especifique. Si se da el caso contrario, se realizará una distinción utilizando una variable booleana entre si es que no se ha llegado a realizar el gesto, o por el contrario se ha terminado de realizar el gesto.

De esta manera podemos definir dos joints diferentes y cómo interactúan, permitiendo definir gestos y los resultados de estos.

3.3.2 Gestión de dibujos

Para comenzar con esta sección es necesario concretar los elementos que se han incluido, los cuales han sido:

1. ***DrawCore:*** un componente que asume el papel de gestionar los dibujos y las funciones pertinentes de la interfaz

2. ***DrawInterface:*** este componente maneja el uso de la interfaz y gestiona los cambios de estado de esta
3. ***DrawTarget:*** este objeto representa el dibujo realizado por el usuario

3.3.2.1 DrawCore

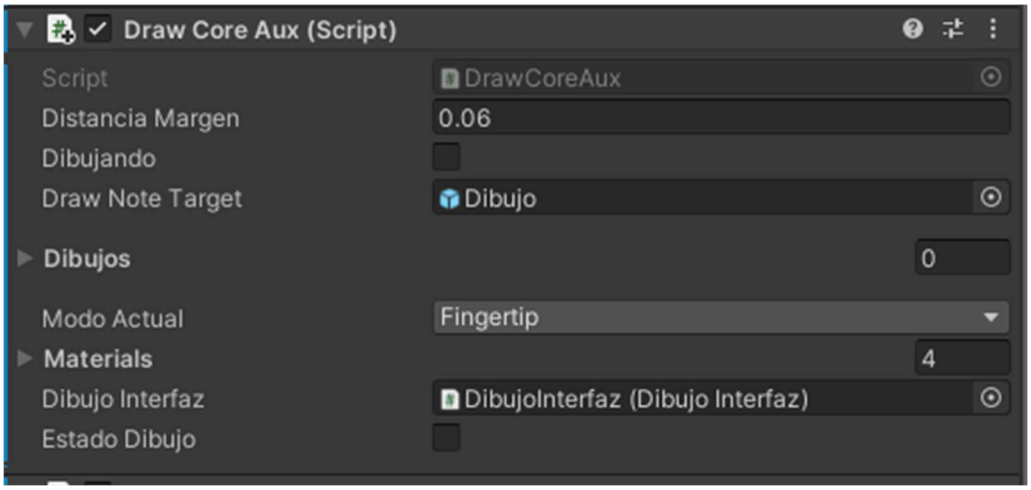


Ilustración 26: Primer diseño de DrawCore

En la ilustración vemos numerosos campos:

- **DistanciaMargen:** este parámetro se refiere a la distancia entre los *joints* para dar lugar a un gesto.
- **Dibujando:** este parámetro nos da una indicación sobre si se dan todos los requisitos para dibujar o no.
- **DrawNoteTarget:** este es el *prefab* que representa el dibujo que se usa para instanciar copias de este.
- **Dibujos:** una lista que se usa para almacenar los dibujos conforme se vayan instanciando, su uso permite la funcionalidad de deshacer y de eliminar los dibujos de la escena.
- **ModoActual:** se han definido diferentes metodologías para crear un dibujo, y se ha propuesto como un atributo enumerado para poder añadir más modos si fuera necesario. Los ejemplos se muestran en la Ilustración 27. En esta etapa hay 2 modos, aunque habrá 3 en la etapa final:

    - **Fingertip:** cuando se realiza el gesto correspondiente se procederá al dibujo como si estuviéramos sosteniendo un lápiz en la mano
    - **Mesh:** cuando se realiza el gesto correspondiente se dibujará en un puntero como si definiéramos una línea usando un puntero laser, este método sólo funciona si el puntero colisiona con el modelo del entorno.
- **Materials:** lista de materiales disponibles para las líneas de dibujo
- **DibujoInterfaz:** referencia al objeto que se usa para la interfaz de dibujo
- **EstadoDibujo:** booleano que representa cuándo está activo o no el componente interfaz.

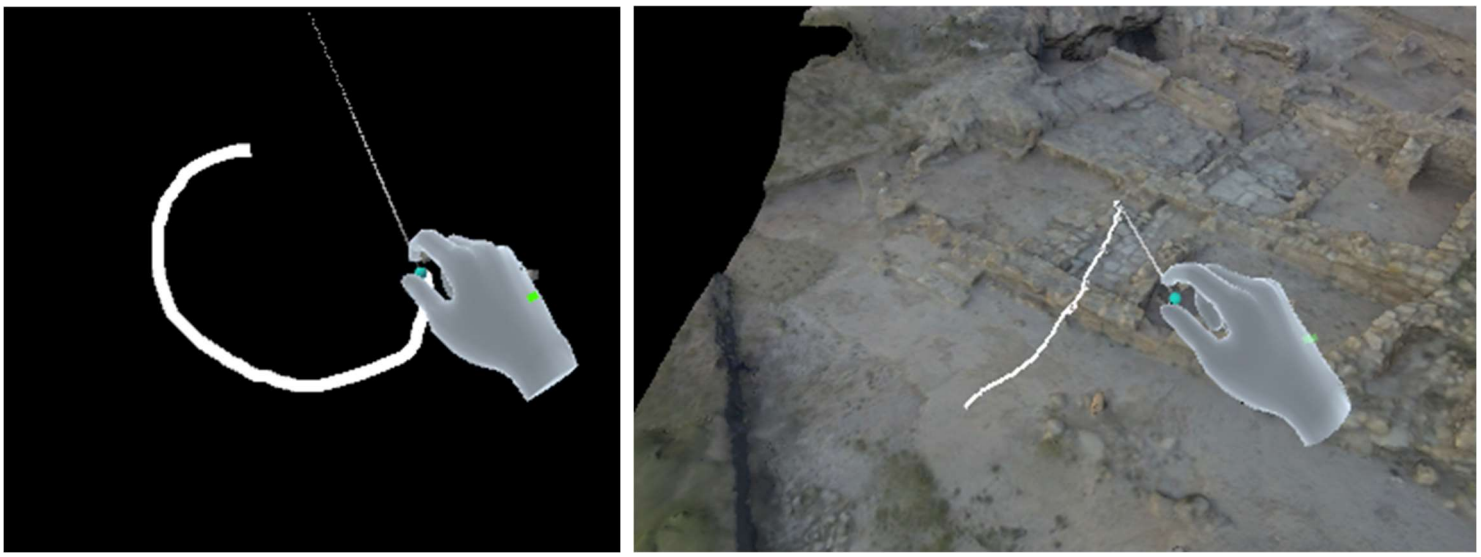


Ilustración 27: Modos de dibujo

En las ilustraciones anteriores se muestran los modos de dibujos presentes en esta versión, el componente que permite la visualización del dibujo será explicado más en detalle en el apartado 3.3.2.3.

3.3.2.2 DrawInterface

La interfaz de dibujo es el elemento que permite al usuario controlar la gestión de dibujos en tiempo de ejecución. Se compone de 8 botones que se dividen en 5 utilidades. A continuación, se muestra una ilustración de la interfaz y seguidamente las utilidades comentadas.

Ilustración 28: Interfaz de dibujo

1. **Dibujar:** este botón activa y desactiva la funcionalidad propia de dibujar. Si está activado el icono del botón cambiará al símbolo ✓ y si está desactivado cambiará a X, ambos iconos obtenidos de
2. **Modo:** este botón sirve para cambiar los modos disponibles de dibujo, el icono también cambia en función del modo seleccionado.
3. **Deshacer:** se ha implementado una función de deshacer la cual es accesible mediante este botón. Deshace el último dibujo creado eliminándolo de la lista de dibujos de **DrawCore**.
4. **Limpiar:** este botón elimina todos los dibujos de la escena y vacía la lista de dibujos de **DrawCore**
5. **Cambiar material:** en la barra inferior de la interfaz se muestran 4 opciones de materiales, estos botones cambian el material predeterminado de los dibujos al material que corresponda.

Como funcionalidad extra, es posible agarrar el menú y bloquearlo en una misma posición del espacio. De manera predeterminada el menú seguirá al jugador

manteniendo siempre una distancia mínima. Esta funcionalidad es posible gracias a los siguientes componentes que ofrece MRTK.

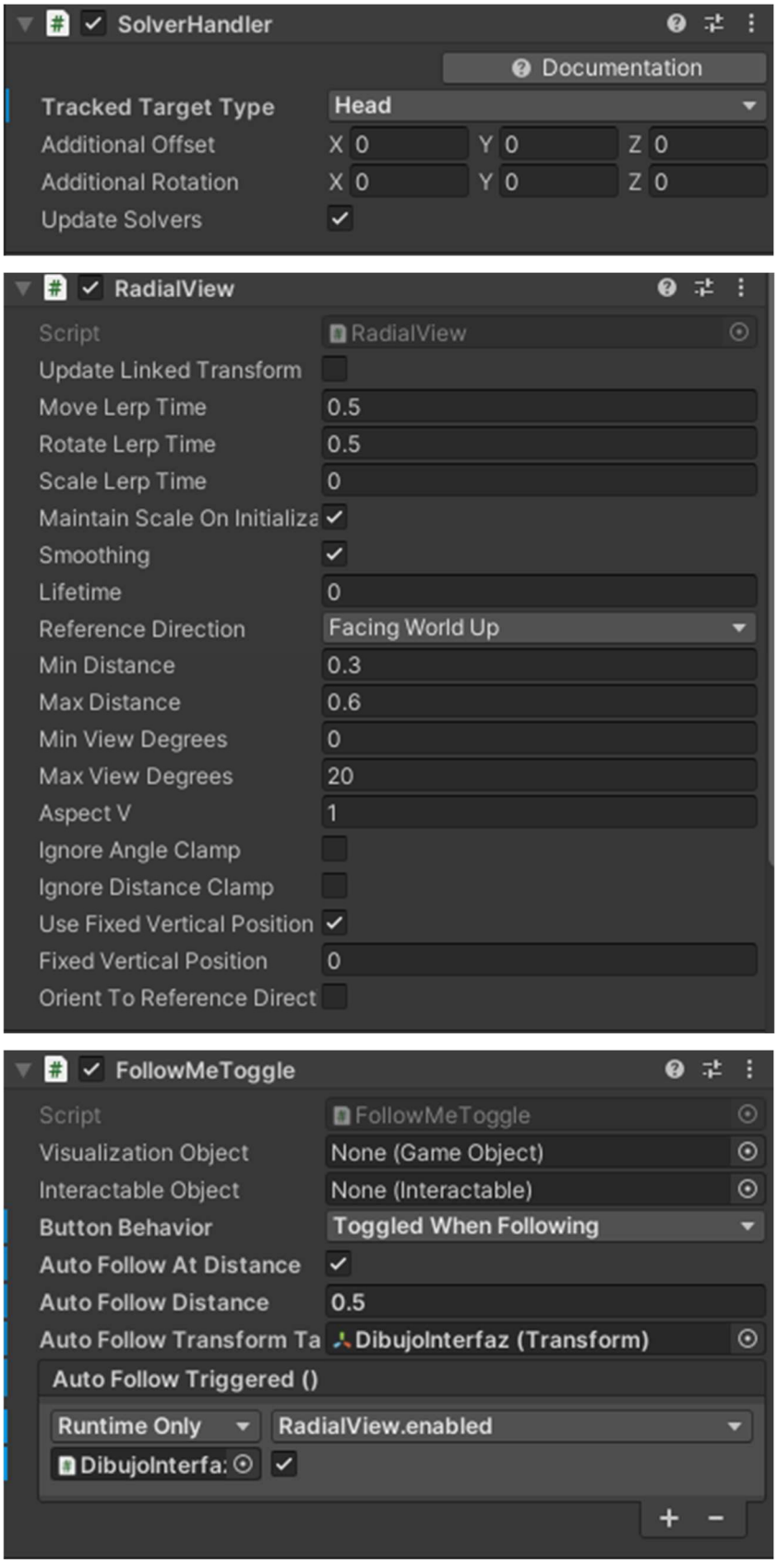


Ilustración 29: Componentes de seguimiento de MRTK

- **SolverHandler:** establece el objeto de referencia para realizar el seguimiento (por ejemplo, la transformación de la cámara principal, el rayo de mano, etc.), maneja la recopilación de componentes del solucionador y ejecuta su actualización en el orden adecuado. [34]
- **RadialView:** mantiene el objeto dentro de un cono de vista proyectado por el objeto al que se hace referencia. Ofrece algunas comodidades como la interpolación de rotaciones y movimientos para aportar realismo a las correcciones de posición y rotación de la interfaz y muchos parámetros para modificar estas interacciones. [35]
- **FollowMeToggle:** este componente se utiliza para detener el funcionamiento de los otros scripts en el momento en el que el menú se recoloca para dejarlo fijo en una posición. También tiene la funcionalidad de volver a activar estos componentes si el usuario se aleja demasiado de la interfaz.

Estos componentes se usan, a su vez, para aportar la misma funcionalidad a otras interfaces y menús, como el descrito en el apartado 3.1.1.

3.3.2.3 DrawTarget

Para la representación más literal de los dibujos es necesario explicar los componentes utilizados. Como necesitamos un objeto que imprima un dibujo conforme nosotros realicemos gestos necesitamos un método para crear dibujos que concuerde con nuestro sistema de reconocimiento de gestos. La solución a este problema es sencilla si usamos un tipo de componente que nos ofrece Unity, llamado *TrailRenderer*.

TrailRenderer representa un rastro de polígonos detrás de GameObject en movimiento. Esto se puede usar para dar una sensación de movimiento enfatizada a un objeto en movimiento, o para resaltar el camino o la posición de los objetos en movimiento [36]. Este componente nos permite crear dibujos si podemos mover un GameObject invisible que deje un rastro por donde pase. Por tanto, el objeto **DrawTarget,** encontrado entre los *assets* del proyecto como **dibujo**, debe ser un GameObject vacío que tenga como hijo un *TrailRenderer.*

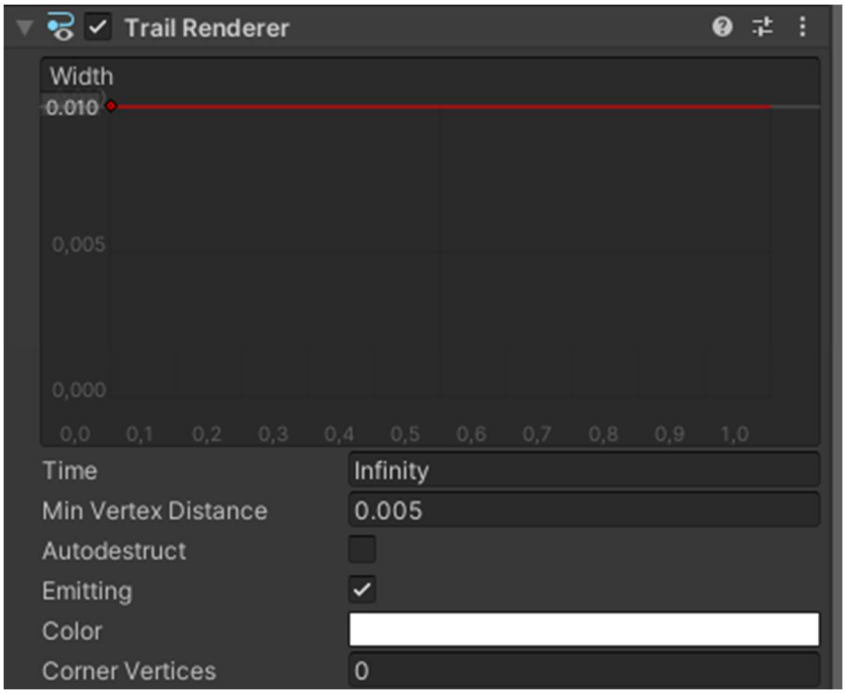


Ilustración 30: Componente TrailRenderer

3.3.3 Actualización de construcciones

En esta versión se ha incluido un nuevo material transparente para las construcciones en vista de la utilidad que ofrece a la hora de identificar las construcciones del terreno y que además ofrece la posibilidad de seguir viendo el terreno aun habiendo construido por encima.

Además de añadir este material, se ha reemplazado el menú flotante por botones más simples y transparentes en los laterales de la construcción. Esto evita abrumar al usuario con demasiadas interfaces y colores. Para reforzar la elección del material transparente, también se ha añadido un control deslizante que permite ajustar la opacidad del material, cambiando la transparencia según se desee mediante la modificación del canal alfa del componente RGB del material.

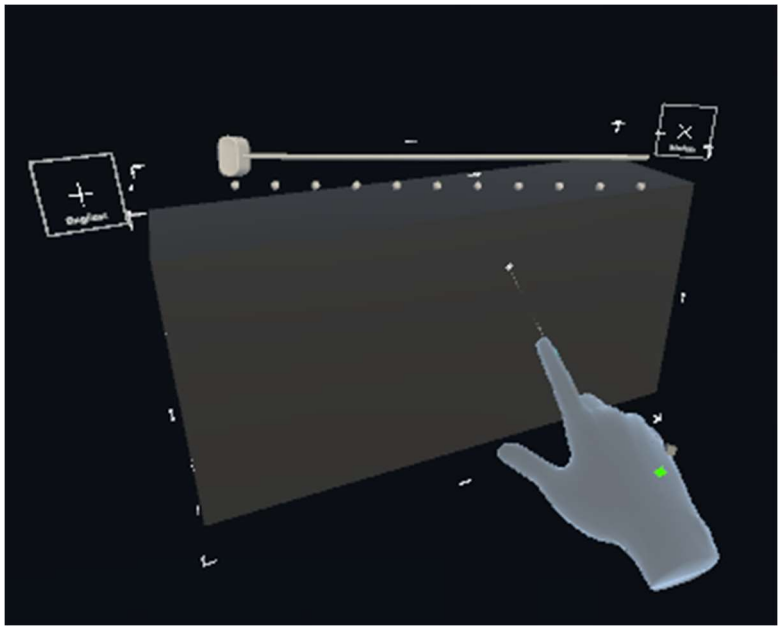


Ilustración 31: Construcción en tercera iteración

3.3.4 Validación

La herramienta de dibujo ha sido implementada de manera que resulta poco costosa, intuitiva y cómoda para el usuario. Se realizó una prueba de validación con dos usuarios potenciales y se concluyó en que su uso era cómodo y adecuado. En algunas ocasiones se usaba la herramienta para dibujar construcciones antes que instanciar las propias construcciones, ya que el proceso de instanciar una construcción y escalarla resultaba mucho más tedioso y menos práctico que simplemente dibujar el muro sobre el escenario.

El material transparente aporta mucha más claridad a la escena al aplicarla sobre las construcciones, pero sin embargo la nueva interfaz de botones frente al menú anterior ha resultado en los mismos problemas previamente. Es difícil calcular una posición cómoda de los botones cuando las construcciones se escalan a bajo tamaño, así como del slider, por lo que se ha decidido aprovechar la nueva funcionalidad de reconocimiento de gestos para replicar el comportamiento de la interfaz sin la necesidad de la presencia de esta. Por último, en vista del comportamiento de los usuarios al dibujar los muros frente a instanciarlos, se ha decidido implementar un sistema de dibujado de muros mediante gestos para instanciarlos, lo cual se asume que agilizará el proceso y resultará en una opción más viable.

3.4 Cuarta iteración

En vista de las observaciones recogidas en la última validación del prototipo y a raíz de una valoración de la dificultad de las tareas planteadas, se ha llegado a la conclusión de que en la cuarta iteración se debe idear un método para instanciar las construcciones mediante un uso alternativo de la herramienta de dibujo. Con la esperanza de que al usuario le resulte mucho más cómodo el uso que con los métodos implementados en las anteriores iteraciones.

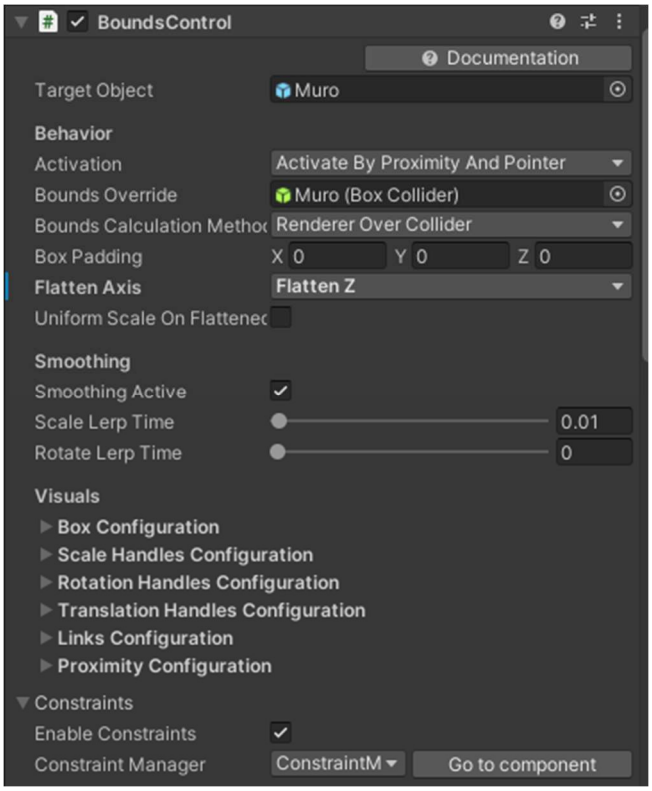
La solución presentada se basa los siguientes puntos clave:

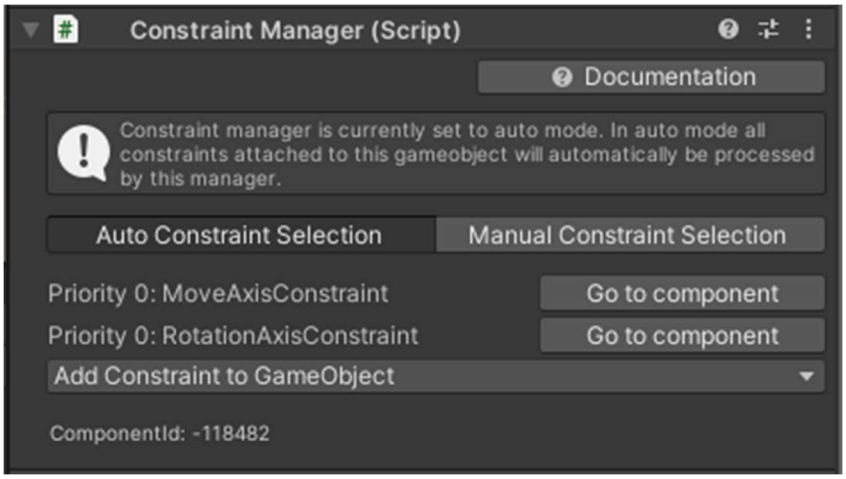
1. Plantear la creación de un nuevo tipo de construcción, que sea capaz de adaptarse a un dibujo y que pueda carecer de cualquier tipo de interfaz que dañe la estética de la escena.
2. Plantear un procedimiento para adaptar la construcción anterior a un dibujo e implantarlo ágilmente sobre el modelo del entorno.

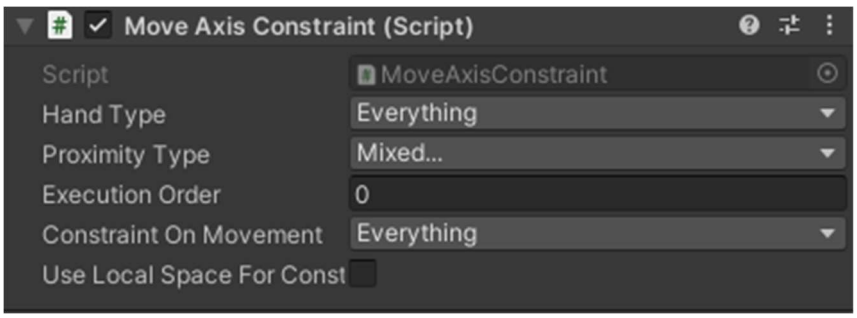
3.4.1 Objeto Building

Para la resolución del primer problema planteado para la cuarta iteración se ha decidido que los objetos que se tenían hasta el momento no podían ajustarse eficazmente a un dibujo y mantener cierta coherencia con el mismo mientras se pudiera seguir realizando transformaciones a placer sobre el objeto. De manera que se ha tenido que construir un nuevo tipo de objeto al que se ha llamado **Building**, con la capacidad de adaptarse a un dibujo de la manera que se especificará en el apartado 3.5.1.

El objeto **Building** consta de una forma rectangular sencilla con la idea de que se pueda adaptar a una recta definida sobre la malla del modelo del entorno. Para que esto ocurra usaremos algunos componentes que nos ofrece MRTK:


BoundsControl
Documentation
Target Object
Muro
Behavior
Activation
Activate By Proximity And Pointer
Bounds Override
Muro (Box Collider)
Bounds Calculation Metho
Renderer Over Collider
Box Padding
X 0
Y 0
Z 0
Flatten Axis
Flatten Z
Uniform Scale On Flattenec
Smoothing
Smoothing Active
Scale Lerp Time
0.01
Rotate Lerp Time
0
Visuals
Box Configuration
Scale Handles Configuration
Rotation Handles Configuration
Translation Handles Configuration
Links Configuration
Proximity Configuration
Constraints
Enable Constraints
Constraint Manager
ConstraintM
Go to component


Constraint Manager (Script)
Documentation
Constraint manager is currently set to auto mode. In auto mode all constraints attached to this gameobject will automatically be processed by this manager.
Auto Constraint Selection
Manual Constraint Selection
Priority 0: MoveAxisConstraint
Go to component
Priority 0: RotationAxisConstraint
Go to component
Add Constraint to GameObject
ComponentId: -118482


Move Axis Constraint (Script)
Script
MoveAxisConstraint
Hand Type
Everything
Proximity Type
Mixed...
Execution Order
0
Constraint On Movement
Everything
Use Local Space For Const

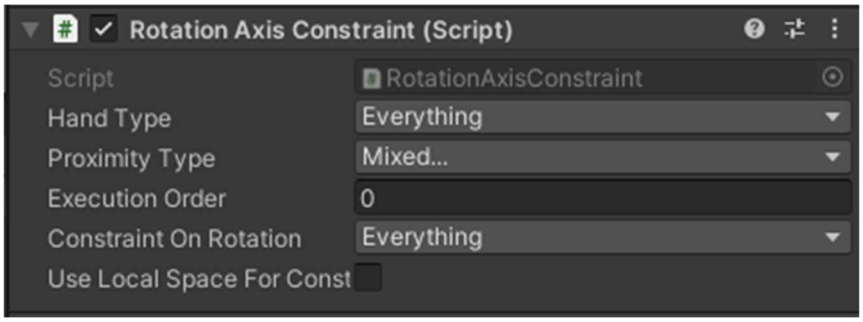


Ilustración 32: Componentes de "Building"

- **BoundsControl:** este componente mostrará un cuadro alrededor del holograma para indicar que se puede interactuar con él. Los agarres en las esquinas y bordes de la caja permiten escalar, rotar o trasladar el objeto. El control de límites también reacciona a la entrada del usuario. Para HoloLens 2 proporciona también una configuración excelente para elementos visuales, ya que aporta retroalimentación visual a la proximidad de los dedos, por ejemplo, para ayudar al usuario percibir la distancia del objeto. [37]
- **ConstraintManager:** este componente permite aplicar una serie de restricciones a las transformaciones del objeto al que se acople. Dicho componente recoge las restricciones y las aplica en un orden concreto, el cual se puede modificar. Su funcionamiento está sujeto a los componentes restantes: [38]
    - **MoveAxisConstraint:** este componente permite restringir el movimiento de un objeto a los ejes designados, sin dejar de permitir la manipulación del objeto.
    - **RotationAxisConstraint:** este componente permite restringir la rotación de un objeto a los ejes designados, sin dejar de permitir la manipulación del objeto.

El componente **BoundsControl** tiene una serie de parámetros clave que permite obtener el comportamiento que deseamos para el objeto **Building.** Estos parámetros son **FlattenAxis** y el campo **ScaleBehaviour,** que se encuentra en el apartado visual de la configuración de escalado**.**

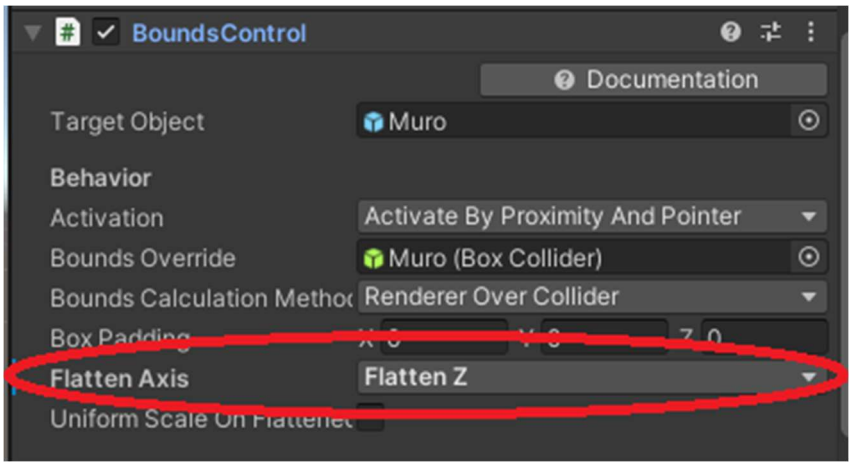


Ilustración 33: FlattenAxis de BoundsControl

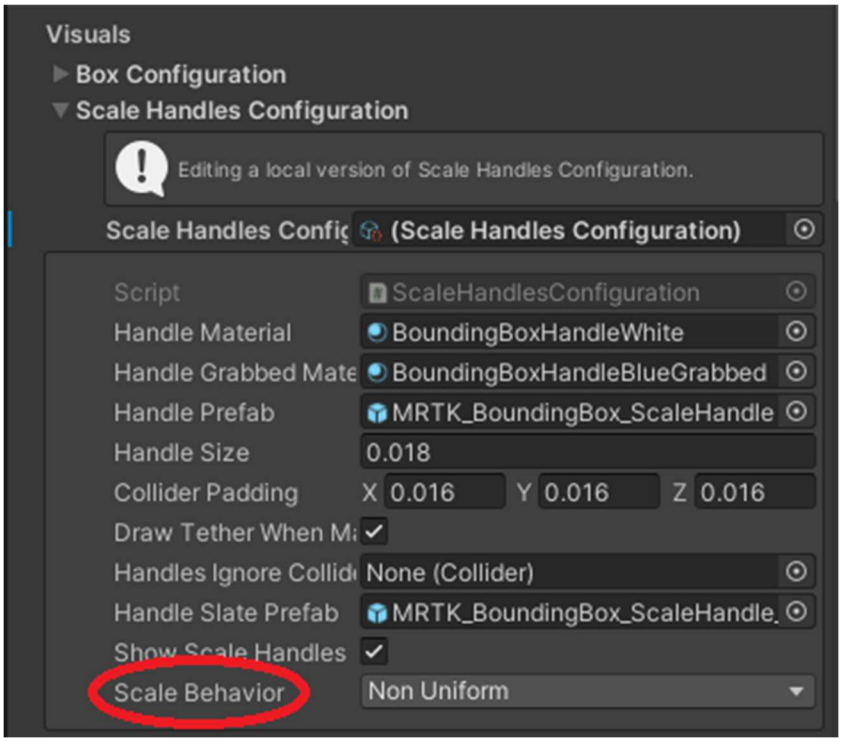


Ilustración 34: ScaleBehaviour de BoundsControl

La configuración que se muestra en las ilustraciones permite los comportamientos que esperamos en el objeto **Building**, es decir, que se ajuste a una línea definida sobre la malla del entorno y que sea posible escalar el objeto de manera no uniforme, pudiendo escalar este en los ejes X e Y. Además, se ha restringido la rotación del objeto para que no pueda modificarse con respecto a la orientación de la

línea definida y, también, se ha restringido el movimiento del objeto para que no sea posible moverlo. De esta manera conseguimos que el objeto **Building** se quede estático en la posición definida para el mismo.

3.4.2 Línea de construcción

La línea definida para la posición, orientación y tamaño aproximado del nuevo objeto **Building**. La construcción de esta línea se ha definido mediante el uso de ambas manos, de manera que se crea una capa de dificultad para prevenir gestos involuntarios que puedan llevar a errores.

Esta línea se basa en el componente *LineRenderer* que aporta Unity. Este componente es similar al componente *TrailRenderer,* pero tiene una diferencia importante, y es que *LineRenderer* [39] permite definir los puntos por los que pasa la línea. La utilidad de esta característica es que es posible crear una línea de previsualización a la posición donde se colocará el nuevo **Building,** y ayudará al usuario a comprender el procedimiento mediante la retroalimentación visual que esto conlleva. En el apartado 3.5.2, se explicará con mayor detalle este componente.

Teniendo estos elementos ya es posible la creación de construcciones mediante dibujado, siguiente los siguientes pasos:

1. Se debe apuntar a una posición de la malla con la mano derecha.
2. Con la mano izquierda se realizará un gesto de “pinza” con los dedos índice y pulgar, creando el punto inicial de la línea donde la mano derecha apunte.
3. Moviendo la mano derecha mientras se apunta a la malla se podrá observar una línea con punto inicial definido en el paso 2 y cuyo punto final se encuentra en la posición donde apuntamos con la mano derecha, la cual cambia en tiempo real.
4. Si se dejamos de realizar el gesto de pinza con la mano izquierda, la línea previsualizada desaparecerá y será sustituida con un objeto **Building** orientado y adaptado a esa línea definida.

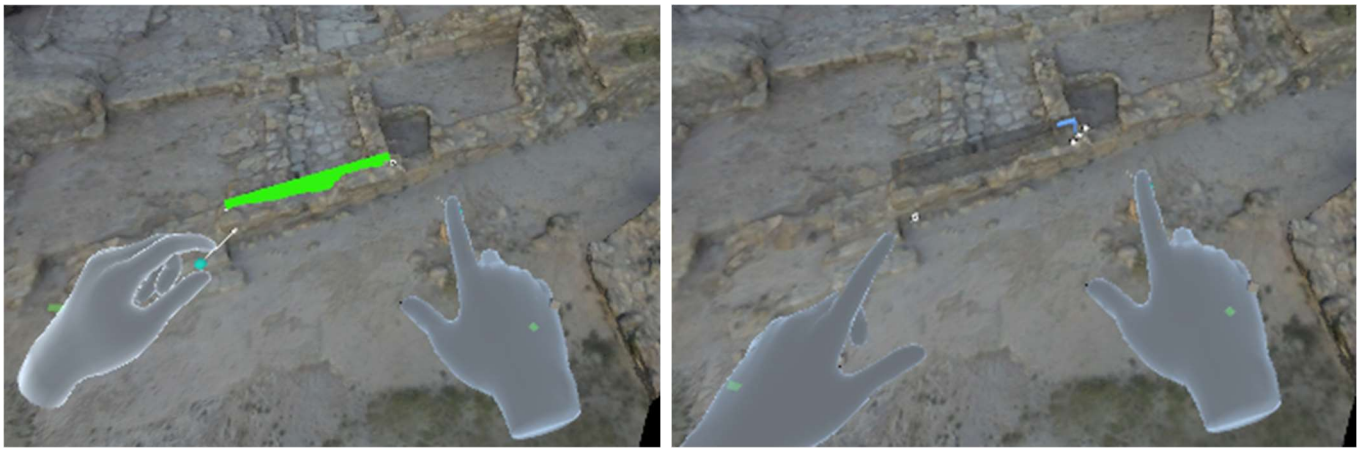


Ilustración 35: Dibujo de Building mediante gestos

3.4.3 Validación

La construcción mediante gestos ha resultado ser mucho más cómoda, ágil y eficaz a la hora de colocar las construcciones en la escena. Se ha observado una mejora sustancial en la cantidad de pasos a recorrer para obtener el mismo resultado, ya que anteriormente se tenía que instanciar un objeto y transformarlo hasta que estuviera correcto mientras que ahora simplemente con dibujar una línea en el suelo se tiene el objeto colocado, faltando únicamente el tener que escalarlo haciendo uso de los agarres proporcionados, siendo este un paso opcional.

Sobre los modos de dibujo se recalcó la imprecisión de las líneas dibujadas en el modo de dibujo *Mesh* cuando se precisaba marcar áreas del terreno, ya que requiere enteramente de la capacidad de pulso del usuario al dibujar la línea, es por ello que para la siguiente iteración se propone un modo de dibujo adicional.

Siendo que la herramienta cumple bastantes objetivos planteados al inicio como requisitos se plantea a continuación unas últimas modificaciones antes de llegar a la versión final del prototipo. Se describen estas modificaciones en el apartado 3.5.

3.5 Quinta iteración

Esta es la última iteración llevada a cabo, pudiendo considerarse la iteración que dará lugar a la versión final del prototipo. Las modificaciones descritas en este apartado aportarán una última funcionalidad y una nueva interfaz para otorgar al usuario la posibilidad de cambiar la textura de los elementos construidos. Se han discutido la inclusión de una serie de funcionalidades, pero habiendo llegado casi al límite del tiempo contratado, se ha tomado la decisión de realizar las siguientes actividades de las propuestas:

1. Incluir una nueva interfaz para modificar las texturas de las construcciones
2. Incluir un nuevo modo de dibujo a los existentes aprovechando el tipo de línea ya definido en el apartado 3.4.2.
3. Añadir objetos manipulables a la escena.

3.5.1 Interfaz de texturas

Para concluir con uno de los requisitos que quedan por implementar es necesario implementar un selector de texturas. El diseño para esta interfaz requiere una visualización clara de las texturas disponibles, ya que se busca aprovechar al máximo la visualización que permite el entorno de RA.

La solución propuesta ha sido un menú radial. El motivo de la elección de este tipo de interfaz frente a otras es que, en un entorno virtual, este tipo de menús resulta cómodo y natural en su uso, ya que permite una rápida selección de opciones mostrando la información en un formato que no obstaculiza la visualización de otros objetos y esto es de suma importancia a la hora de ofrecer una experiencia agradable y cómoda.

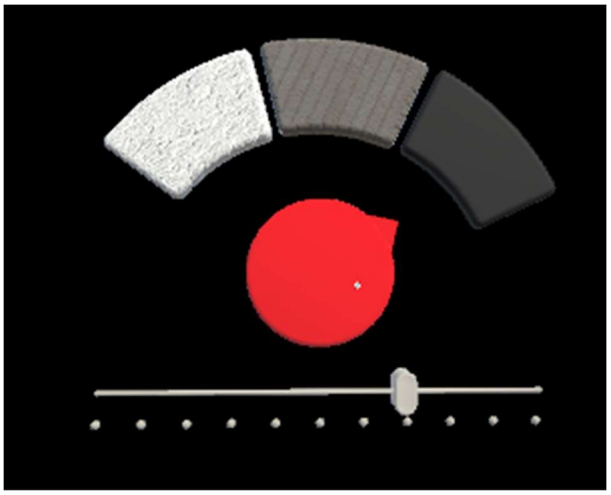


Ilustración 36: Interfaz de texturas

En la ilustración se puede comprobar la interfaz de texturas implementada. En primer lugar, se puede observar en la parte superior un semicírculo de botones a los que se les ha aplicado la textura que representan; en la parte central se muestra una rueda que se ha diseñado usando la herramienta Blender, la cual sirve de indicador visual al apuntar a la textura que se encuentra seleccionada en todo momento y, por último, en la parte inferior vemos el objeto *slider* que se describe en el apartado 3.3.3, el cual nos permite controlar la transparencia del material transparente.

A esta interfaz se le aplican los mismos componentes utilizados para la interfaz de dibujo descrita en el apartado 3.3.2.2, los cuales permiten que la interfaz siga al usuario por la escena, de manera que el usuario no pierda de vista la interfaz en caso de necesitarla. Aunque a diferencia de la interfaz de dibujo, ha sido necesario aplicar a estos componentes una rotación extra fija para que pudiera adaptarse a la rotación de la cabeza del usuario.

Ilustración 37: Rotación adicional a la interfaz

3.5.2 Dibujo con polilíneas

El modo de dibujo propuesto surge a raíz de la necesidad de algunos usuarios a delimitar zonas del terreno usando la herramienta de dibujo. Es por ello por lo que se ha propuesto añadir dibujos utilizando polilíneas, de manera que se puedan realizar delimitaciones más precisas.

Para el propósito de este procedimiento se utiliza el componente *LineRenderer* de Unity. Este componente ya se ha usado para delimitar la línea del objeto *Building* descrita en el apartado 3.4.2. Se procede a continuación a describir el funcionamiento de los atributos utilizados en más detalle:

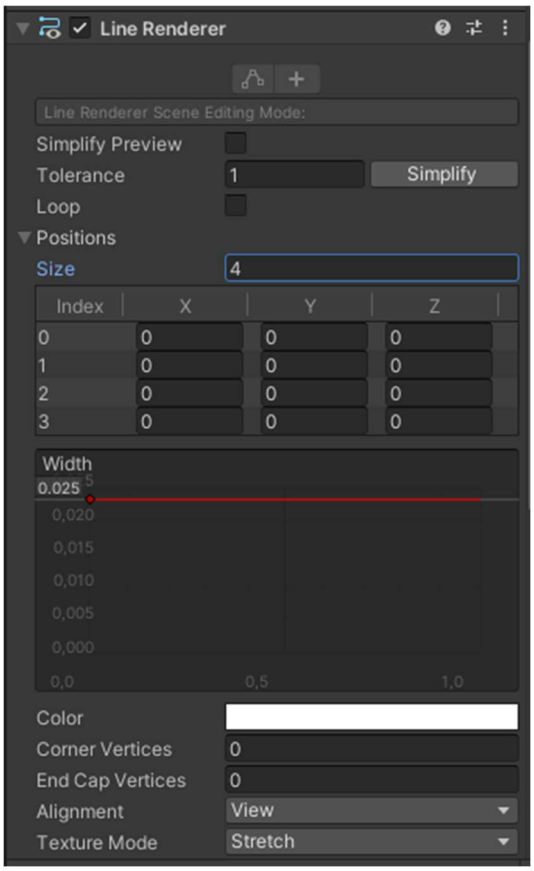


Ilustración 38: Componente LineRenderer

El componente ofrece una serie de funcionalidades muy útiles a la hora de realizar a las polilíneas, como puede ser la modificación del número de vértices y sus posiciones en el espacio. También da la posibilidad de activar un booleano con nombre *Loop*, que dibuja una línea entre el primer y el último punto de la línea para cerrar el polígono definido.

Para ello se define un nuevo modo de dibujo simplemente añadiendo un valor extra al enumerado *ModoDibujo,* presente en el script DrawCore*,* descrito en el apartado *3.3.2.1.* También es necesario definir un gesto para esta nueva funcionalidad, que simplemente definiremos como un gesto de pinza con la mano derecha habiendo activado este modo de dibujo usando la interfaz. A continuación, se definen los pasos a seguir para usar esta funcionalidad:

1. Se activa la interfaz de dibujo mediante el uso de la interfaz de mano, Ilustración 39: Actualización de interfaz de mano.
2. Se cambia el modo de dibujo a “Polyline”, utilizando el segundo botón de la interfaz de dibujo.
3. Realizando el gesto de pinza con la mano derecha se añade un vértice a la línea, y esta se actualiza.
4. Si se desea pasar a una nueva polilínea, se debe realizar un gesto de pinza con el dedo corazón y el pulgar de la mano derecha.

Ilustración 39: Actualización de interfaz de mano

A su vez se ha implementado una función para calcular los puntos intermedios entre los vértices con el objetivo de que se ajusten a la geometría del modelo sobre el que se dibuja.

La función mencionada atiende a un parámetro de precisión que se ajusta en el componente DrawCore, este parámetro se refiere al número de puntos intermedios que se ajustarán de la línea, de manera que, a mayor precisión, más ajustada estará la línea a expensas de tener que calcular un mayor número de puntos. A continuación, se puede observar el resultado obtenido:

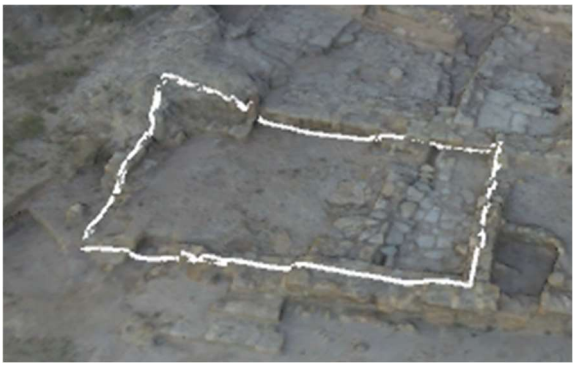


Ilustración 40: Dibujo con polilínea

3.5.3 Objetos manipulables

En cuestión de manipulación de objetos, se requieren los siguientes componentes:

- **MeshFilter:** este componente requiere una referencia a una malla, colabora con el componente MeshRenderer para renderizar una malla. [40]
- **MeshRenderer:** este componente renderiza la malla a la que MeshFilter hace referencia [41]
- **Collider:** Unity ofrece los collider para los propósitos de detección de colisiones físicas. En este caso se usará un MeshCollider. [42]
- **ObjectManipulator:** este componente hace que el objeto al que se acopla se vuelva transformable (traslación, rotación y escalado).

Con estos componentes podemos manipular cualquier objeto de la escena, y por tanto el problema se reduce a escoger los objetos que podrán ser manipulados. Para la escena que se presenta, la cual funciona de ejemplo, se ha incluido el Betilo del Santuario de la Puerta del Sol que se piensa que representa a la Diosa [43] y una

piedra tallada en la que se representa al Nokaki, el identificado como “héroe fundador del yacimiento de Puente Tablas” [44]

A su vez, se ha proporcionado la capacidad de recolocar el modelo del entorno sobre el que se trabaja aprovechando la capacidad de manipulación de los objetos, restringiendo la rotación en cualquier eje horizontal para evitar fallos indeseados en la orientación del lugar.

3.5.4 Validación

Durante las pruebas con usuarios se ha concluido en que los nuevos elementos son útiles y cómodos de usar. Los elementos manipulables son elementos curiosos de observar, aunque puede que convenga aportar explicaciones sobre lo que se está manipulando en un contexto más orientado a su historia. Sin embargo, puesto que la aplicación está destinada a ser usada para la educación, se presupone que será alguien de conocimiento el que explique estos datos, haciendo irrelevante su explicación dentro de la aplicación.

3.6 Pruebas finales

Tras la conclusión del desarrollo del proyecto se ha procedido a realizar una validación final mediante pruebas de usuario y una encuesta anónima. El propósito de esta encuesta es recopilar opiniones y comentarios que permitan identificar tanto los aspectos positivos como las áreas de mejora del prototipo.

Lamentablemente habiéndose cumplido el tiempo designado para el desarrollo del prototipo, las observaciones que se han recopilado y el feedback aportado por los usuarios quedan como sugerencias para trabajos futuros que busquen perfeccionar el prototipo en una versión más avanzada del mismo proyecto.

3.6.1 Validación de la usabilidad del sistema

En este apartado se recogen los resultados de las encuestas completadas y los comentarios observados durante las pruebas con usuarios, tanto de usuarios con experiencia como usuarios sin experiencia con formato de realidad aumentada.

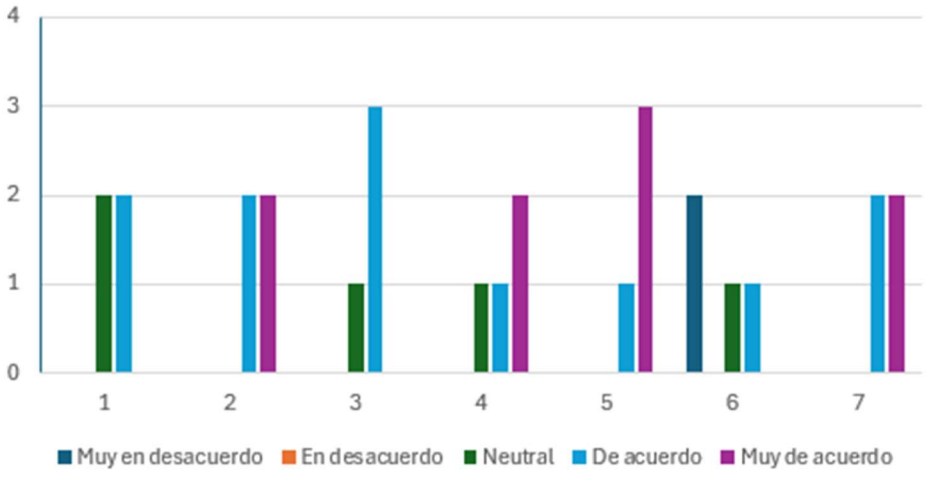


Ilustración 41: Resultados de las encuestas de validación

Las cuestiones han sido las siguientes:

1. El casco y la interacción mediante gestos resultan cómodos
2. Me gusta el aspecto de la aplicación
3. La aplicación responde bien a mis acciones y comandos
4. La aplicación me ha permitido comprender mejor la importancia del entorno simulado
5. Pienso que estudiar arqueología de esta forma puede ser divertido
6. Tengo experiencia previa con gafas de realidad aumentada
7. Estoy satisfecho/a con la experiencia general proporcionada por la aplicación

Generalmente los usuarios están contentos con la experiencia que proporciona la aplicación, encuentran utilidad a la herramienta y potencial para ser usada en entornos docentes. Es importante destacar que, aun teniendo experiencia con el paradigma de interacción, para un uso adecuado de la herramienta se requiere un proceso de entrenamiento en la dinámica a fin de comenzar a sacar partido a las

funcionalidades que ofrece la aplicación. La función que más ha gustado en general ha sido la herramienta de dibujo y se ha sugerido que se use una metodología similar para instanciar los muros. Se ha destacado que la herramienta no ofrece información relevante sobre el entorno, dando a sugerir que sería interesante proporcionar una manera de añadir algún tipo de explicación sobre los contenidos mostrados.

Los resultados de las construcciones, aunque trabajosos a falta de un periodo de entrenamiento, son generalmente satisfactorios y bien recibidos. El control de las funcionalidades parece correcto y cómodo.

4 CONCLUSIONES Y TRABAJOS FUTUROS

En términos generales, el prototipo ofrece funcionalidades atractivas para los usuarios. La herramienta ofrece una posibilidad muy interesante en el ámbito docente como nuevo enfoque a la hora de exponer información sobre un emplazamiento arqueológico o lugar emblemático de relevancia en el campo de la arqueología así de como de elementos más pequeños como exvotos o monumentos.

En vista de las observaciones obtenidas en la última validación, se tienen distintas propuestas de mejor como trabajos a futuro:

1. **Implementar un tutorial en el prototipo:** para el prototipo presentado no se ha contemplado la creación de un tutorial de uso fuera de la documentación, sería conveniente usar las herramientas de realidad aumentada para ofrecen pistas visuales sobre los usos de la aplicación.
2. **Mejoras visuales de los objetos:** en la creación del prototipo y como consecuencias directas de los requisitos técnicos para la correcta sincronización de versiones en Git, los modelos utilizados no poseen una resolución óptima y esto afecta a la visualización de algunos elementos. Mejorar este aspecto sería recomendable.
3. **Ampliar la funcionalidad de construcción:** el uso de dibujos es el más alabado por los usuarios que han podido probar la aplicación dado a que tiene un uso intuitivo y no requiere de un aprendizaje excesivo. Sería conveniente adaptar la construcción aún más a este modelo de interacción.

4. **Facilitar la adición de entornos o modelos:** en el prototipo se ha priorizado la funcionalidad de las herramientas y las interfaces sobre la interacción con los modelos y entornos. En ese aspecto y para que el prototipo adquiera toda la usabilidad posible, es recomendable que se añada la capacidad de poder cambiar el entorno y los objetos simulados a fin de simplificar la interacción con Unity a los usuarios o facilitar una herramienta alternativa para ese propósito.
5. **Mejorar la presentación de los objetos en la escena:** durante la validación final del prototipo se observó que uno de los usuarios no podía ubicar los objetos en el entorno debido a una cuestión de presentación de las HoloLens 2, al estar basada en imágenes holográficas y transparentar demasiado los objetos. El cambio más inmediato sugerido es una mejora de la presentación de los objetos, ya sea destinando un espacio para su exposición en la escena o añadiendo alguna animación que permita al objeto moverse en un estado de descanso.
6. **Mejor en el sistema de reconocimiento de gestos:** es un hecho que sin un entrenamiento previo es muy probable que se consigan resultados inesperados al usar la herramienta si es que no se tiene experiencia previa con el paradigma de interacción de realidad aumentada, es por ello conveniente realiza mejoras en el sistema de reconocimiento de gestos para evitar resultados inesperados y facilitar el aprendizaje de la herramienta.

5 APÉNDICES

En esta sección se incluirá documentación relevante para facilitar la comprensión del proyecto.

5.1 Guía original del Trabajo Fin de Título

En los últimos años la utilización de tecnologías basadas en realidad aumentada (AR, Augmented Reality) han tomado mucho auge debido fundamentalmente al enorme abaratamiento de los dispositivos móviles y a la mejora de las técnicas de posicionamiento y registrado. La AR permite combinar El mundo

real con el mundo virtual de manera que al usuario pueda en tiempo real interactuar con ambos mundos simultáneamente. Para ello se requiere registrar mundo real y mundo virtual para que la superposición sea lo más precisa como sea posible. En este TFG se persigue utilizar las Microsoft HoloLens 2 en una aplicación del ámbito de la arqueología. El trabajo consistirá en explorar las funcionalidades de dicha instrumentación y construir una aplicación AR dirigida a la manipulación de material arqueológico, y poniendo especial relieve con la integración de los elementos virtuales generados como aumentadores de la realidad que rodea al usuario. Las funcionalidades serán definidas a partir de diversas reuniones establecidas con el profesorado del grado en arqueología con el fin de identificar contenidos en sus materias susceptibles de ser apoyados por las tecnologías estudiadas en este TFG.

5.1.1 Conocimientos previos

Se recomienda haber cursado asignaturas de la mención de sistemas gráficos, con el fin de que el estudiante esté familiarizado con la terminología utilizada.

5.1.2 Objetivos del TFG

- Estudio comparativo de las diferentes herramientas de desarrollo de aplicaciones de aire para diferentes dispositivos.
- Conocer y comprender las dificultades de aplicación de la informática en otras disciplinas
- Análisis de requisitos para la construcción de una aplicación AR para educación a nivel de Grado en Arqueología
- Construcción de un prototipo de aplicación basada en AR para simular acciones en el ámbito de la arqueología atendiendo a diferentes entornos reales
- Generación de documentación comprensible que permita la replicación del trabajo

5.1.3 Metodología a desarrollar

- Comparación de librerías de desarrollo de AR
- Selección de librería y dispositivo

- Estudio bibliográfico sobre metodologías de desarrollo de aplicaciones AR - Construcción de la aplicación esqueleto
- Construcción del prototipo de aplicación
- Pruebas de validación
- Documentación

5.1.4 Documentos y formatos de entrega

- Documentación en formato PDF
- Código fuente del prototipo

5.2 Instalación y configuración del sistema

En esta sección se documentará paso por paso el método a seguir para compilar el proyecto e instalarlo en las Microsoft HoloLens 2. Para llevar a cabo este proceso se requiere Visual Studio.

Primero se debe realizar una compilación del proyecto de Unity utilizando el Universal Windows Platform. Si no está seleccionado es posible cambiar la plataforma pulsando el botón *switch platform* en el menú *build settings* de Unity.

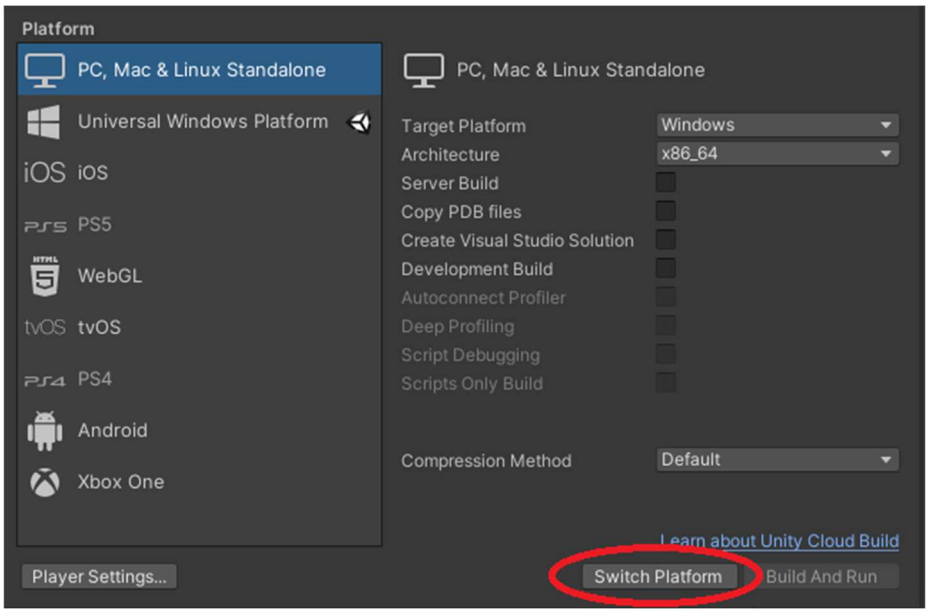


Ilustración 42: Cambio de plataforma en Unity

Una vez compilado, abrimos con Visual Studio el archivo solución con formato *.sln*

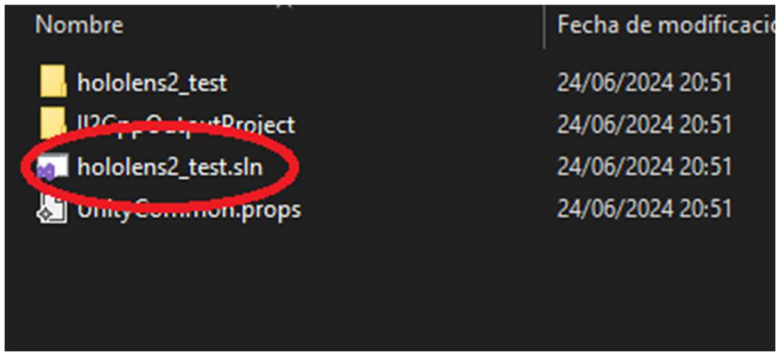


Ilustración 43: Archivo solución en directorio de compilación

Una vez abierto es de suma importancia configurar los siguientes valores de la siguiente manera:

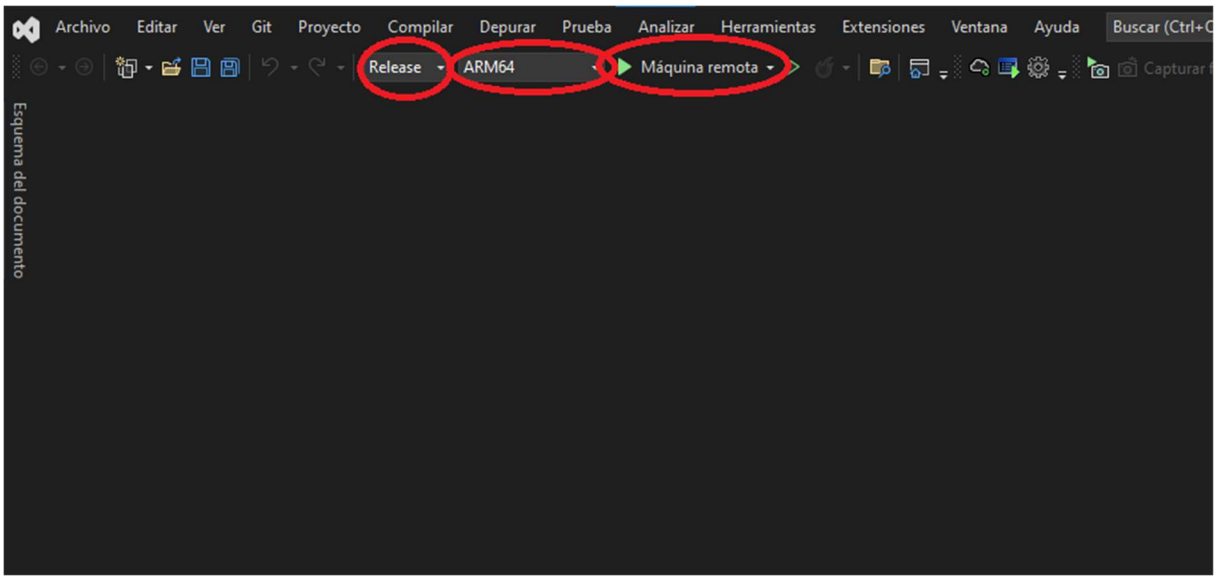


Ilustración 44: Configuración de compilado de proyecto en Visual Studio

- Modo release, ya que de otra manera es posible que haya problemas de rendimiento
- ARM64, ya que es la arquitectura hardware que admiten las HoloLens 2
- Máquina remota, este apartado es condicional. Es posible emparejar el computados con las HoloLens mediante el portal de dispositivos de Windows. En caso contrario se debe conectar las HoloLens físicamente al ordenador mediante un cable. [45]

Una vez esté todo listo simplemente compilamos la solución haciendo clic en compilar proyecto o pulsando Ctrl + Mayus + B.

Habiendo seguido estos pasos la aplicación se habrá compilado y se encontrará en el menú de aplicaciones de las HoloLens una aplicación ejecutable de nuestro proyecto.

5.3 Manuales de usuario

5.3.1 Reposicionamiento de entorno

Es posible reposicionar el entorno realizando un gesto de pinza con la mano derecha o izquierda, simulando que se agarra el entorno como si fuera un objeto físico real. Tras esta interacción, el entorno se bloquea, permitiendo las demás interacciones y previniendo que vuelva a moverse de la posición en la que se ha fijado anteriormente.

Se recomienda para prevenir errores realizar este proceso cuidadosamente ya que si se realiza movimientos bruscos es posible que el gesto se cancele sin previo aviso.

5.3.2 Manipulación de objetos

Ya sea con la mano izquierda o con la mano derecha es posible manipular los objetos holográficos presentados en la escena:

- **Rotación:** es posible rotar un objeto una vez agarrado realizando movimientos de rotación con las manos
- **Escalado:** es posible escalar el objeto agarrándolo con ambas manos y arrastrándolo en direcciones opuestas, o acercando las manos mientras se sostiene el objeto.
- **Traslación:** simplemente agarrando el objeto es posible desplazarlo moviendo la mano mientras esté agarrado.

5.3.3 Colocación de muros

Se pueden colocar muros de dos maneras:

1. Interactuando con el menú de mano e instanciando un muro pulsando el primer botón del menú. Después manipular este muro para adecuarlo a las preferencias del usuario.
2. Utilizando el dibujado de muros de la siguiente manera:
    - Apuntar con el punto de la mano derecha al entorno.
    - Realizar un gesto de agarre con la mano izquierda, juntando el dedo índice y el pulgar
    - Utilizar la mano derecha para dibujar la línea donde se situará el muro en el escenario.
    - Relajar el gesto de la mano izquierda una vez la línea se encuentre según las preferencias del usuario
    - Una vez instanciado el muro, se puede escalar usando los agarres localizados en las esquinas del muro.

5.3.4 Menú de mano

Es posible acceder al menú de mano realizando un movimiento de supinación con la muñeca. Se podrá comprobar que el menú sigue a la muñeca allá donde se mueva. Es posible bloquear la posición del menú agarrándolo y soltándolo en cualquier lugar del entorno. A través de este menú se pueden activar las demás interfaces del prototipo además de poderse instanciar los muros.

5.3.5 Dibujos

Para acceder a las funcionalidades de dibujo del prototipo debe activarse la interfaz usando el menú de mano. Una vez el menú esté activo se puede configurar la funcionalidad de dibujo mediante los botones de la interfaz de dibujo:

1. Dibujar: activa o desactiva la capacidad de dibujo
2. modo: avanza mediante el uso de una lista circular entre las opciones de modo de dibujo
3. deshacer: deshace el último dibujo realizado

4. limpiar: elimina de la escena todos los dibujos realizados

5.3.5.1 Modos de dibujo

- **Fingerprint:** uniendo el pulgar y el dedo índice de la mano derecha se instancia un dibujo que siguen los movimientos de la mano mientras el gesto se mantenga, simulando un lápiz.
- **Mesh:** realizando el mismo gesto definido en el modo anterior se instanciará el dibujo donde impacte el puntero de la mano derecha. Este dibujo solo se realiza sobre la malla del entorno simulado
- **Polyline:** al realizar el mismo gesto definido para los otros modos de dibujo, se añade un nuevo vértice a la polilínea creada. Se ha introducido un gesto adicional para este modo de dibujo, que consiste en unir el dedo corazón con el dedo pulgar. Su función es cambiar a otra polilínea diferente a la anterior, evitando así la adición interminable de vértices a una única línea, siendo este proceso controlado por el usuario.

6 DEFINICIONES Y ABREVIATURAS

- **Framework**: El framework es un término que proviene del inglés y significa «marco de trabajo» o «estructura». En el ámbito de la programación, un framework es un conjunto de herramientas y librerías que se utilizan para desarrollar aplicaciones más fácilmente y de manera más eficiente.
- **Shooter**: género de videojuegos que engloba un amplio número de subgéneros que tienen la característica común de permitir controlar un personaje que, por norma general, dispone de un arma (mayoritariamente de fuego) que puede ser disparada a voluntad
- ***in-game****:* Del inglés in game (dentro del juego). Se refiere a una acción que sucede dentro de la escena.

- **Open-source: t**ipo de software, que se distribuye con una licencia de código abierto que permite al usuario, si tiene las capacidades necesarias, poder utilizar el código fuente para modificarlo y realizar cambios y mejoras.
- **Joint:** del inglés (articulación). En el contexto de este TFG se usa para referirse a la representación dentro de la simulación de las articulaciones de la mano.
- **Shaders:** programas especializados que se ejecutan en la GPU y se utilizan para controlar el renderizado de gráficos. Permiten realizar efectos visuales avanzados como iluminación, sombras, y texturas, mejorando la apariencia y el realismo de los objetos en aplicaciones gráficas y videojuegos-
- **Assets:** son recursos que se utilizan en la creación de juegos y aplicaciones. Estos pueden incluir modelos 3D, texturas, sonidos, scripts, animaciones y otros archivos multimedia que se importan al proyecto y se integran en la escena para construir el contenido interactivo y visual
- **Feedback:** información proporcionada a una persona o sistema sobre su desempeño o comportamiento, con el objetivo de mejorar y ajustar futuras acciones. Traducido como retroalimentación.

7 BIBLIOGRAFÍA

[1] «Rockcontent,» . Available: https://rockcontent.com/es/blog/realidad-aumentada/.

[2] «Microsoft.com,» . Available: https://www.microsoft.com/es-es/hololens/buy.

[3] «Museo Arqueológico de Alicante,» . Available: https://www.marqalicante.com/Exposiciones/es/EXVOTOS-IBERICOS-DE-BRONCE-E11.html.

[4] «www.juntadeandalucia.es,» . Available: https://www.juntadeandalucia.es/cultura/enclaves/enclave-arqueologico-puente-tablas#.

[5] «Wikipedia, la enciclopedia libre,» . Available: https://es.wikipedia.org/wiki/L._Frank_Baum.

[6] «History of information,» . Available: https://historyofinformation.com/detail.php?id=4233.

[7] «Wikipedia, la enciclopedia libre,» . Available: https://es.wikipedia.org/wiki/Morton_Heilig.

[8] D. Calvo Puente, «Linkedin,» . Available: https://es.linkedin.com/pulse/realidad-aumentada-su-origen-y-evolucion-deborah-calvo-puente#main-content.

[9] «Wikipedia, la enciclopedia libre,» . Available: https://es.wikipedia.org/wiki/Jaron_Lanier.

[10] . Available: https://www.magicleap.com/es-es.

[11] C. Cubillos Figueroa y R. Alfaro Arancibia, Marzo 2014. . Available: http://opac.pucv.cl/pucv_txt/txt-4500/UCE4968_01.pdf.

[12] «BIMCV,» Proyectos CEIB-CS, . Available: https://bimcv.cipf.es/bimcv-projects/asistencia-quirurgica-con-realidad-aumentada/.

[13] H. Gwo-Jen , W. Po-Han , C. Chi-Chang y T. Nien-Ting , «www.tandfonline.com,» . Available: https://www.tandfonline.com/doi/full/10.1080/10494820.2015.1057747.

[14] A. F. Bulagang y A. Baharum, «www.researchgate.net,» . Available: https://www.researchgate.net/publication/322206541_A_Framework_for_Developing_Mobile-Augmented_Reality_in_Higher_Learning_Education.

[15] . A. Murat y A. Gökçe, «www.sciencedirect.com,» . Available: https://www.sciencedirect.com/science/article/abs/pii/S1747938X16300616?via%3Dihub.

[16] «Sketchfab,» . Available: https://sketchfab.com/feed.

[17] «Wikipedia, la enciclopedia libre,» . Available: https://es.wikipedia.org/wiki/Unreal_Engine.

[18] «UT-Hub,» . Available: https://www.ut-hub.com/nanite-unreal-engine-5/.

[19] «EpicGames,» . Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine?application_version=5.0.

[20] . Available: https://assetstore.unity.com/assets-for-pros.

[21] . Available: https://es.wikipedia.org/wiki/Microsoft_Visual_Studio.

[22] . Available: https://es.wikipedia.org/wiki/NetBeans#.

[23] «GitHub,» . Available: https://github.com/microsoft/MixedRealityToolkit-Unity.

[24] «XR Today,» . Available: https://www.xrtoday.com/mixed-reality/what-is-mrtk-unity-mixed-reality-toolkit/.

[25] «NormalMap-Online,» . Available: https://cpetry.github.io/NormalMap-Online/.

[26] «flaticon,» . Available: https://www.flaticon.es/.

[27] «Valtx,» . Available: https://www.valtx.pe/blog/metodologias-para-el-desarrollo-de-software-que-son-y-para-que-sirven.

[28] I. Cornejo, «SlideShare,» 18 Septiembre 2017. . Available: https://es.slideshare.net/IvnCornejo/modelo-de-ciclo-de-vida-de-prototipado-evolutivo-79872674.

[29] «Navarro Abogados,» . Available: https://navarroabogados.net/que-es-una-iguala/.

[30] «Talent,» . Available: https://es.talent.com/.

[31] «mediterraneoantiguo.com,» . Available: https://mediterraneoantiguo.com/2017/05/27/el-recinto-de-la-puerta-del-sol-de-puente-tablas-un-santuario-oracular/.

[32] «docs.unity3d,» . Available: https://docs.unity3d.com/es/2018.4/Manual/CollidersOverview.html.

[33] «learn.microsoft.com,» . Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/hand-tracking?view=mrtkunity-2022-05.

[34] «learn.microsoft.com,» . Available: establece el objeto de referencia para realizar el seguimiento (por ejemplo, la transformación de la cámara principal, el rayo de mano, etc.), maneja la recopilación de componentes del solucionador y ejecuta su actualización en el orden adecuado..

[35] «learn.microsoft.com,» . Available: https://learn.microsoft.com/es-es/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/solvers/solver?view=mrtkunity-2022-05#radialview.

[36] «docs.unity3d,» . Available: https://docs.unity3d.com/es/2018.4/Manual/class-TrailRenderer.html.

[37] «learn.microsoft.com,» . Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/bounds-control?view=mrtkunity-2022-05.

[38] «learn.microsoft.com,» . Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/constraint-manager?view=mrtkunity-2022-05.

[39] «docs.unity3d,» . Available: https://docs.unity3d.com/es/2018.4/Manual/class-LineRenderer.html.

[40] «docs.unity3d,» . Available: https://docs.unity3d.com/Manual/class-MeshFilter.html.

[41] «docs.unity3d,» . Available: https://docs.unity3d.com/Manual/class-MeshRenderer.html.

[42] «docs.unity3d,» . Available: https://docs.unity3d.com/es/2018.4/Manual/CollidersOverview.html.

[43] M. Agudo Villanueva, «mediterraneoantiguo.com/,» . Available: https://mediterraneoantiguo.com/2017/05/27/el-recinto-de-la-puerta-del-sol-de-puente-tablas-un-santuario-oracular/.

[44] A. Ruiz, «autopista.es,» . Available: https://www.autopista.es/planeta2030/cafe-con-arturo-ruiz-arqueologo-yacimiento-ibero-puente-tablas-jaen_235022_102.html.

[45] «learn.microsoft.com,» . Available: https://learn.microsoft.com/es-es/windows/uwp/debug-test-perf/device-portal-desktop.

[46] R. Pressman, Ingeniería del Software, McGraw-Hill, 2010.

[47] «Wikipedia, la enciclopedia libre,» . Available: https://es.wikipedia.org/wiki/Jaron_Lanier.

[48] R. Pressman, «Ingeniería del Software: Un enfoque práctico,» . Available: https://www.javier8a.com/itc/bd1/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF.

[49] V. D. J. ORTEGA HERNANDEZ, 14 Febrero 2019. . Available: https://reunir.unir.net/bitstream/handle/123456789/8280/ORTEGA%20HERNANDEZ%2C%20VICTOR%20DE%20JESUS.pdf?sequence=1&isAllowed=y#:~:text=La%20estimaci%C3%B3n%20de%20esfuerzos%20es,que%20afectar%C3%A1%20directamente%20sus%20finanzas.