Universidad Zaragoza
1542

Trabajo Fin de Grado

Diseño y desarrollo de un entorno virtual interactivo 3D para aplicaciones educativas y empresariales

Design and development of an Interactive 3D virtual environment for educational and business applications

Autora

María Isabel Bernal González

Director

Juan Miguel Lujano rojas

Grado en Ingeniería en Diseño Industrial y Desarrollo de Producto

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2024

Diseño y desarrollo de un entorno virtual interactivo 3D para aplicaciones educativas y empresariales

RESUMEN

En este trabajo se ha llevado a cabo el diseño y desarrollo de un prototipo de entorno virtual 3D interactivo, cuyo objetivo principal es ofrecer una experiencia inmersiva tanto en el ámbito educativo como empresarial. El entorno permite a los usuarios interactuar de manera intuitiva con objetos dentro de estancias que se han diseñado para tal fin.

Se seleccionaron herramientas como Sweet Home 3D para la creación de edificios, y Unity para el desarrollo del entorno interactivo. Los edificios creados en Sweet Home 3D fueron importados en Unity, donde se añadió la lógica de la interacción mediante scripts en C#**.** Además, se emplearon personajes de Mixamo y se utilizaron herramientas como Visual Studio Code, junto con el plugin Bito.

El prototipo incluye dos casos de uso: uno educativo y otro empresarial. En el entorno educativo, los usuarios pueden explorar un edificio inspirado en uno de la Universidad de Zaragoza, interactuando con elementos que proporcionan información sobre temas académicos. En el entorno empresarial, el usuario puede visitar una vivienda virtual, interactuando con diferentes objetos obtener información sobre ellos.

Finalmente, se destacan los posibles desarrollos futuros del proyecto, como la implementación de realidad virtual (VR) y la integración de inteligencia artificial para mejorar la interacción en los entornos 3D. El uso de tecnologías como el Universal Render Pipeline (URP) de Unity es fundamental para la compatibilidad con dispositivos VR.

INDICE

1. Introducción

1.1 Objetivos y Justificación

En este trabajo de fin de grado se ha llevado a cabo el diseño y desarrollo de un prototipo de entorno virtual 3D interactivo, con el fin de proporcionar a los usuarios una experiencia inmersiva y enriquecedora. Este entorno está compuesto por diversas estancias, permitiendo a los usuarios interactuar con determinados elementos de manera intuitiva.

El diseño de entornos se basa en una cuidadosa planificación, considerando aspectos como la arquitectura de las estancias, la disposición de los elementos visuales y la usabilidad general. Y en cuanto al desarrollo técnico, se ha buscado utilizar herramientas y lenguajes de programación específicos para la creación de entornos 3D.

El prototipo resultante busca ser válido tanto para fines educativos como empresariales. En el ámbito educativo, los usuarios pueden explorar conceptos y escenarios de manera práctica, lo que facilita el aprendizaje y la comprensión. Por otro lado, en el contexto empresarial, el entorno virtual puede utilizarse para presentaciones, ventas y simulaciones, mejorando la eficiencia y la efectividad de las comunicaciones.

Además, este proyecto está alineado con los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas. Específicamente, se relaciona con el ODS 4 (Educación de Calidad) al proporcionar una herramienta educativa innovadora y accesible. Asimismo, contribuye al ODS 9 (Industria, Innovación e Infraestructura) al explorar tecnologías emergentes y fomentar la colaboración entre el ámbito académico y empresarial.

Para garantizar la adaptabilidad futura, se ha documentado todo el proceso de diseño y desarrollo, incluyendo decisiones clave, códigos fuente y recursos utilizados. Así, cualquier modificación o ampliación posterior será más sencilla y eficiente.

Además, se ha considerado la posibilidad del uso del prototipo con gafas de realidad virtual y la aplicación de inteligencia artificial en futuros trabajos.

1.2 Metodología

Para el desarrollo del entorno virtual 3D, se ha decidido emplear la plataforma Unity [1] debido a su amplia compatibilidad y su rápida curva de aprendizaje; Sweet Home 3D [2] para el diseño arquitectónico, y se ha optado por Mixamo [3] para la obtención de personajes 3D. En apartados posteriores se justifica la elección de cada uno de estas herramientas.

Unity hace uso del lenguaje de programación C# [4]. Para editar código de C#, se ha utilizado Visual Studio Code [5], este editor es totalmente gratuito. Se podría haber utilizado otro editor, pero VS Code presenta ciertas ventajas. Por un lado, la instalación es muy ligera, y tras ser instalado se pueden incluir varios plugins que ayudan la programación autocompletando código. En este caso se ha utilizado el plugin Bito [6]. Existen otros plugins específicos para trabajar con Unity que pueden instalarse en VS Code, pero en este trabajo no se han utilizado. Además, se ha utilizado ChatGPT [7] como asistente en la programación. Así ha sido posible resolver los problemas que han ido surgiendo a lo largo de la realización de este trabajo.

En Unity, para seleccionar el editor externo que se desea utilizar, hay que ir a Edit → Preferences, y ahí se puede seleccionar el editor externo que se desee usar.

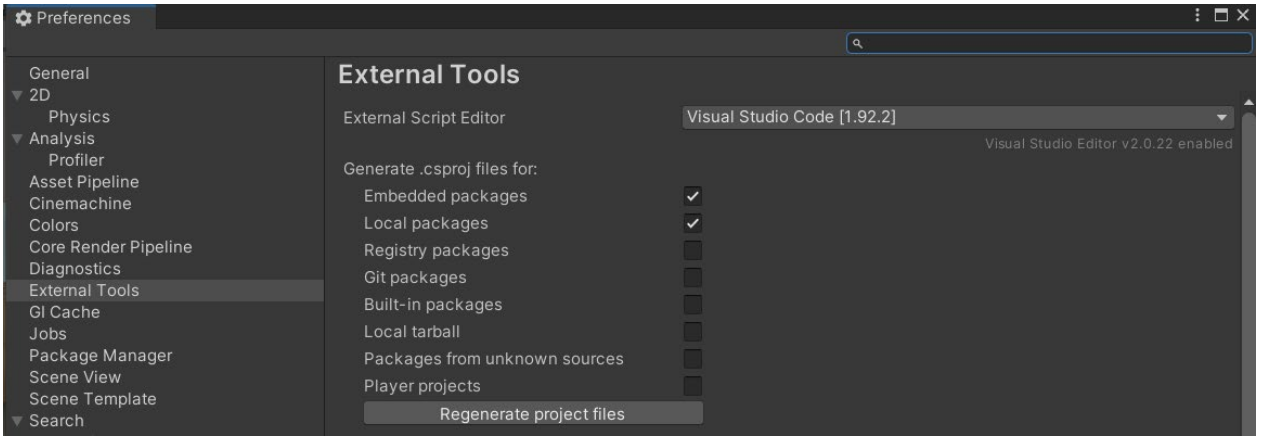


Figura 1. External Tools.

Además, como parte del proceso, se han desarrollado ejemplos con el objetivo de demostrar la aplicabilidad de este trabajo tanto para aplicaciones docentes como empresariales.

Los dos prototipos desarrollados pueden descargarse desde el siguiente link:

https://drive.google.com/file/d/1DRpoAGW8eEXMQOzdym7sUYAue7i0C0Pp/view?usp=drive_link

Los ficheros que se descargan incluyen únicamente aquellos componentes que, por cuestiones de licencia, se pueden distribuir libremente. Se incluye un fichero “LICENSE” donde se indican las condiciones de uso, modificación, etc., de lo ficheros que se encuentran disponibles para su descarga.

A lo largo de este documento se hace referencia a las fuentes utilizadas para poder llevar a cabo los prototipos.

2. Selección de Herramientas

2.1 Selección de herramienta para la creación de edificios y habitaciones

Para seleccionar la herramienta más adecuada para la creación de edificios y habitaciones se ha realizado la siguiente comparativa y estudio de los programas considerados más óptimos para este fin.

Sweet Home 3D [2]

Es una herramienta de diseño de interiores gratuita y fácil de usar, que permite a los usuarios crear planos de planta en 2D y verlos en 3D. Es una herramienta diseñada para ser accesible incluso para aquellos sin conocimientos previos en diseño o modelado 3D. Sus usos más comunes son el diseño de interiores, la planificación de espacios, y la presentación de proyectos de reforma.

Además, dispone de una interfaz intuitiva que permite crear fácilmente las paredes de una casa, agregar puertas, ventanas, y muebles con un simple arrastrar y soltar. Ofrece una visualización 3D interactiva con la que puedes ver el diseño desde cualquier ángulo. E incluye una amplia biblioteca de muebles y elementos decorativos, con la posibilidad de importar más desde otros recursos.

Blender [8]

Es una herramienta de modelado 3D gratuita y de código abierto, conocida por su capacidad para crear desde simples objetos hasta escenas 3D complejas. Es utilizado tanto por aficionados como por profesionales en diversos campos, como la animación, el modelado 3D, y la creación de videojuegos. Tiene una curva de aprendizaje pronunciada debido a la cantidad de funciones que posee y a su complejidad. Sus usos más comunes son el modelado arquitectónico, la animación 3D, los efectos visuales, y el desarrollo de videojuegos.

Además, permite crear modelos 3D detallados de edificios, habitaciones y cualquier otro tipo de objeto. Ofrece motores de renderizado para generar imágenes y animaciones fotorrealista. Incluye herramientas avanzadas para animar objetos y simular fenómenos físicos.

Unity [1]

Es un motor de videojuegos que también se puede utilizar para crear y visualizar edificios y entornos en 3D. Aunque no es una herramienta de modelado en sí, puede utilizarse para ese fin, y también es posible importar modelos desde Blender, Sweet Home 3D u otros programas. Además, ofrece la posibilidad de programar interacciones, física, iluminación, y comportamientos para crear entornos virtuales interactivos.

En cuanto a su dificultad de uso, crear estructuras simples en Unity no es complicado, y su curva de aprendizaje no es pronunciada, pero la integración de scripts y la creación de proyectos más complejos puede requerir conocimientos de programación y diseño. Dada la gran cantidad de material y vídeos disponibles sobre Unity [9], se pueden encontrar soluciones para resolver cualquier desarrollo que se desee abordar.

SketchUp [10]

Es una herramienta popular por su facilidad de uso para modelar edificios y estructuras arquitectónicas. Ofrece una versión gratuita y otra de pago con funcionalidades adicionales. Es ideal para arquitectos y diseñadores que necesitan crear modelos rápidos y precisos sin una curva de aprendizaje muy pronunciada.

Autodesk Revit [11]

Software utilizado por arquitectos, ingenieros y constructores. Es más avanzado que las otras herramientas mencionadas y está diseñado para proyectos grandes y complejos, con un enfoque en la colaboración y la precisión técnica.

Chief Architect [12]

Herramienta profesional para diseñadores y arquitectos que permite crear planos de planta y visualizar construcciones en 3D. Tiene funcionalidades avanzadas para el diseño de interiores y exteriores, pero es más complejo que Sweet Home 3D.

En la Tabla 1 se muestran las características más importantes de las herramientas que se han revisado.

Tabla 1: Tabla resumen características herramientas de creación de edificios. Elaboración propia.

Herramienta	Uso Principal	Dificultad	Tipo de Software	Características Clave	Plataforma	Coste
Sweet Home 3D	Diseño de interiores y planos de planta	Muy baja	Software de diseño de interiores	Interfaz intuitiva, fácil de usar, vista 3D en tiempo real, biblioteca de muebles y objetos	Windows, macOS, Linux	Gratuito
Blender	Modelado 3D y animación	Media-Alta	Software de modelado 3D	Modelado 3D avanzado, renderizado fotorrealista, animación, simulación, extensibilidad por plugins	Windows, macOS, Linux	Gratuito y código abierto
Unity	Creación de videojuegos y simulaciones	Media	Motor de videojuegos	Editor de escenas en 3D, integración con otros modeladores 3D, renderizado en tiempo real, scripting	Windows, macOS, Linux	Gratuito/Pago
SketchUp	Modelado 3D simplificado	Baja-Media	Software de modelado 3D	Interfaz fácil de usar, ideal para arquitectos y diseñadores, biblioteca 3D Warehouse	Windows, macOS	Gratuito/Pago
Autodesk Revit	Modelado de información de construcción (BIM)	Alta	Software BIM	Diseño arquitectónico detallado, enfoque en colaboración, precisión técnica, modelado paramétrico	Windows	Pago (Licencia)
Chief Architect	Diseño arquitectónico y de interiores	Media-Alta	Software de diseño arquitectónico	Herramientas avanzadas para planos de planta y visualización 3D, biblioteca de elementos de construcción	Windows, macOS	Pago (Licencia)

La elección de la herramienta depende del nivel de complejidad del proyecto y de la experiencia del usuario. Sweet Home 3D es ideal si se busca una solución sencilla y directa para el diseño de interiores. Blender es perfecto para quienes necesitan un control total sobre el modelado y renderizado en 3D, aunque posee una curva de aprendizaje más pronunciada. Unity, por su parte, es una opción idónea si se desea crear entornos interactivos, especialmente si se planea integrar esos modelos en un videojuego o una simulación.

Para usuarios que necesitan crear modelos arquitectónicos precisos y detallados, SketchUp o Autodesk Revit podrían ser los más adecuados. Sin embargo, si el objetivo es simplemente visualizar un espacio y experimentar con diferentes disposiciones, Sweet Home 3D podría ser la herramienta más conveniente.

Finalmente se seleccionó Sweet Home 3D en lugar de usar herramientas más complejas como Blender, o modelar directamente en Unity. Esta decisión se basa en que Sweet Home 3D es muy intuitivo y fácil de usar, lo que permite a usuarios sin experiencia en modelado 3D crear rápidamente diseños de edificios y habitaciones, ya que su interfaz de arrastrar y soltar permite diseñar planos de planta y amueblar espacios fácilmente. Además, es una herramienta gratuita y de código abierto.

Por otro lado, permite crear prototipos de manera eficiente, facilitando así la experimentación con diferentes diseños sin perder tiempo en detalles técnicos complejos, y una vez que el diseño básico está listo se puede exportar a Unity, donde pueden ser añadidos efectos de iluminación y comportamientos interactivos. Esta combinación permite aprovechar las fortalezas de ambas herramientas, la simplicidad de Sweet Home 3D para el diseño arquitectónico inicial, y la potencia de Unity para la creación de entornos interactivos.

2.2 Selección de motor de programación

En cuanto a los motores de programación, se ha realizado un estudio en el que se han revisado varias herramientas.

Unity [1]

Fue lanzado en 2005 por Unity Technologies, con el objetivo de hacer accesible el desarrollo de videojuegos para todo tipo de desarrolladores, incluidos aquellos con poca experiencia técnica. Originalmente solo funcionaba en macOS, pero se expandió rápidamente a otras plataformas.

Permite crear juegos tanto en 2D como en 3D, con herramientas físicas, animación y efectos especiales. También permite la creación de experiencias en realidad virtual (VR) y aumentada (AR). Y cuenta con una tienda de recursos llamada Asset Store [13], donde se pueden descargar elementos para los proyectos.

- Sistemas Operativos: Windows, macOS, Linux.
- Plataformas: PC, consolas (como PlayStation y Xbox), dispositivos móviles (iOS, Android), web, y dispositivos de realidad virtual y aumentada.
- Coste: Tiene una versión gratuita, pero para proyectos más grandes o comerciales existen licencias con funcionalidades adicionales.

Godot Engine [14]

Creado en 2014, como un proyecto de código abierto, por Juan Linietsky y Ariel Manzur. Su desarrollo fue impulsado por la necesidad de una herramienta de creación de juegos que fuera accesible y flexible.

Permite crear juegos en 2D y 3D. Además, utiliza un lenguaje de programación propio llamado GDScript, aunque también puede utilizar C# y C++.

- Sistemas Operativos: Windows, macOS, Linux.
- Plataformas: PC, consolas, dispositivos móviles y web.
- Coste: Gratuito y de código abierto.

Unreal Engine [15]

Desarrollado por Epic Games y lanzado por primera vez en 1998. Permite crear juegos 3D de alta calidad, e incluye un sistema de físicas avanzado y un motor de renderizado que produce gráficos de gran realismo. También permite desarrollar proyectos VR y AR.

- Sistemas Operativos: Windows, macOS, Linux.
- Plataformas: PC, consolas, dispositivos móviles y VR/AR.
- Coste: Gratuito para proyectos pequeños, pero Epic Games cobra un porcentaje de los ingresos si el juego genera más de una cierta cantidad de dinero.

Game Maker Studio [16]

Creado por Mark Overmars en 1999 y posteriormente adquirido por YoYo Games. Permite a los usuarios crear juegos sin necesidad de aprender a programar, aunque también ofrece opciones avanzadas para programadores.

Es especialmente adecuado para la creación de juegos en 2D, con una interfaz sencilla que permite diseñar juegos mediante arrastrar y soltar elementos. También tiene soporte para 3D, pero más limitado.

- Sistemas Operativos: Windows, macOS.
- Plataformas PC, dispositivos móviles, consolas y web.
- Coste: Ofrece una versión gratuita, pero las versiones de pago permiten exportar juegos a diferentes plataformas y ofrecen más funcionalidades.

Cocos Creator [17]

Es parte del ecosistema Cocos, desarrollado por Chukong Technologies en 2016. Está basado en el motor Cocos2d-x, ampliamente utilizado para juegos móviles. Adecuado para crear juegos en 2D, aunque también permite 3D. Es popular en el desarrollo de juegos móviles debido a su rendimiento y eficiencia. Incluye un entorno visual y es compatible con JavaScript y TypeScript.

- Sistemas Operativos: Windows, macOS.
- Plataformas: Móviles, web y PC.
- Coste: Gratuito y de código abierto, con opciones de pago.

Flax Engine [18]

Es un motor relativamente nuevo, lanzado en 2020 por Flax Labs. Se diseñó para ser ligero y rápido, adecuado para desarrolladores que buscan alto rendimiento en sus proyectos.

Es conocido por su rendimiento y su capacidad para manejar gráficos 3D avanzados, y permite programar en C#. Es adecuado para juegos tanto en 2D como en 3D.

- Sistemas Operativos: Windows.
- Plataformas: PC, consolas, y dispositivos móviles.
- Coste: Con licencia gratuita para proyectos pequeños y opciones de pago para comerciales.

GDevelop [19]

Motor de código abierto creado por Florian Rival.

Especialmente diseñado para juegos 2D, permite crear juegos usando un sistema de eventos visuales que no requiere conocimientos de programación.

- Sistemas Operativos: Windows, macOS, Linux.
- Plataformas: PC, web y dispositivos móviles.
- Coste: Gratuito y de código abierto.

Stride Engine [20]

Anteriormente conocido como Xenko, fue desarrollado por Silicon Studio y lanzado en 2014. Es un motor de código abierto centrado en ofrecer un entorno de desarrollo profesional para juegos en 3D.

Ofrece un motor de renderizado potente y soporte completo para gráficos 3D, utiliza C# como lenguaje de programación principal, y es adecuado para juegos que requieren gráficos detallados y efectos visuales avanzados.

- Sistemas Operativos: Windows.
- Plataformas: PC y consolas.
- Coste: Código abierto y gratuito.

Open 3D Engine (O3DE) [21]

Es un motor de código abierto desarrollado por la Fundación Linux y Amazon Web Services (AWS), lanzado en 2021.

Permite crear experiencias en 3D de alta calidad, especialmente adecuado para proyectos grandes y complejos.

- Sistemas Operativos: Windows, Linux.
- Plataformas: PC, consolas, dispositivos móviles.
- Coste: Gratuito y de código abierto.

GB Studio [22]

Fue lanzado por Chris Maltby en 2019. Permite a los usuarios crear juegos en 2D con gráficos retro, simulando el estilo de los juegos de la consola portátil Game Boy.

- Sistemas Operativos: Windows, macOS, Linux.
- Plataformas: Emuladores de Game Boy, web y PC.
- Coste: Gratuito.

En la Tabla 2 se muestran las características más importantes de los entornos de programación que se han revisado.

Unity es el que se ha seleccionado y utilizado para llevar a cabo este trabajo. Posee una curva de aprendizaje poco pronunciada y posee todas las características que se necesitan para alcanzar los objetivos que se plantearon inicialmente. Además, posee una gran comunidad de usuarios y una tienda de recursos (Asset Store) donde se pueden encontrar elementos ya creados, lo que puede acelerar el desarrollo. Puede exportar proyectos a diferentes plataformas, como PC, consolas, y dispositivos móviles. Y es adecuado tanto para pequeños proyectos como para desarrollos más grandes y complejos, lo que permite comenzar con algo sencillo y expandirlo según sea necesario.

Tabla 2: Tabla resumen características motores de programación. Elaboración propia.

Motor	2D/3D	Lenguaje de Programación	Coste
Unity	2D/3D	C#, Visual Scripting	Gratuito/Pago
Godot Engine	2D/3D	GDScript, C#, C++	Gratuito
Unreal Engine	3D	C++, Blueprint	Gratuito/Pago
Game Maker Studio	2D/3D	GML (Game Maker Language)	Gratuito/Pago
Cocos Creator	2D/3D	JavaScript, TypeScript	Gratuito
Flax Engine	2D/3D	C#	Gratuito/Pago
GDevelop	2D	Sin programación (eventos)	Gratuito
Stride Engine	3D	C#	Gratuito
Open 3D Engine	2D/3D	C++, Lua, Python	Gratuito
GB Studio	2D	Sin programación (visual)	Gratuito

3. Desarrollo de Prototipos

3.1 Creación de edificios

Con el programa Sweet Home 3D se han creado dos edificios diferentes, uno para cada uno de los dos casos de aplicación práctica (uso docente y empresarial). Por un lado, para el uso docente, a partir de un plano del edificio Torres Quevedo de la Universidad de Zaragoza (Campus Río Ebro), se ha creado un edificio basado en él, pero en una versión muy simplificada. Dispone de dos plantas, conectadas por dos escaleras, y una serie de estancias: varias aulas, salas de estudio, un comedor y baños en cada planta. Este edificio tiene el fin de que, al ser posteriormente exportado a Unity, se pueda interactuar con elementos que proporcionen al usuario información de manera que se pueda tener una experiencia didáctica inmersiva. En las Figura 2, 3 y 4 se muestran varias imágenes del edificio.

Figura 2. Exterior edificio.

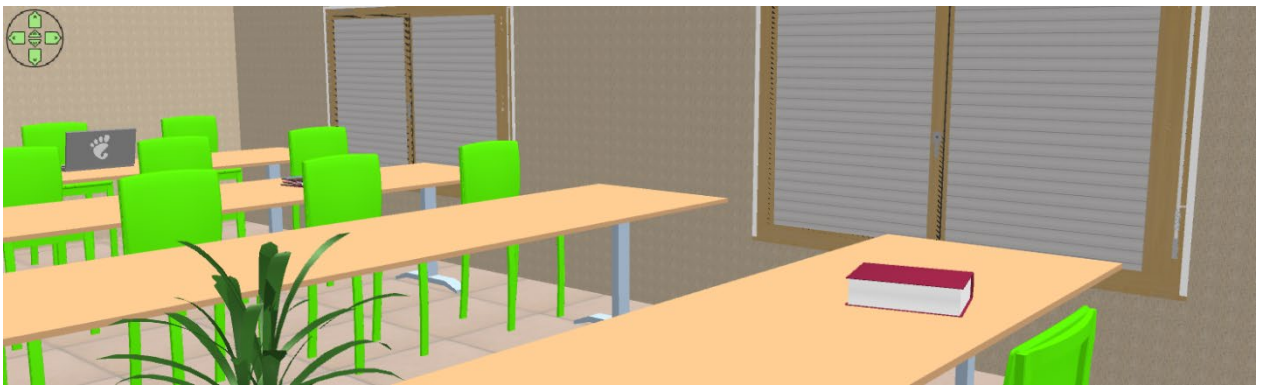


Figura 3. Aula de clases.

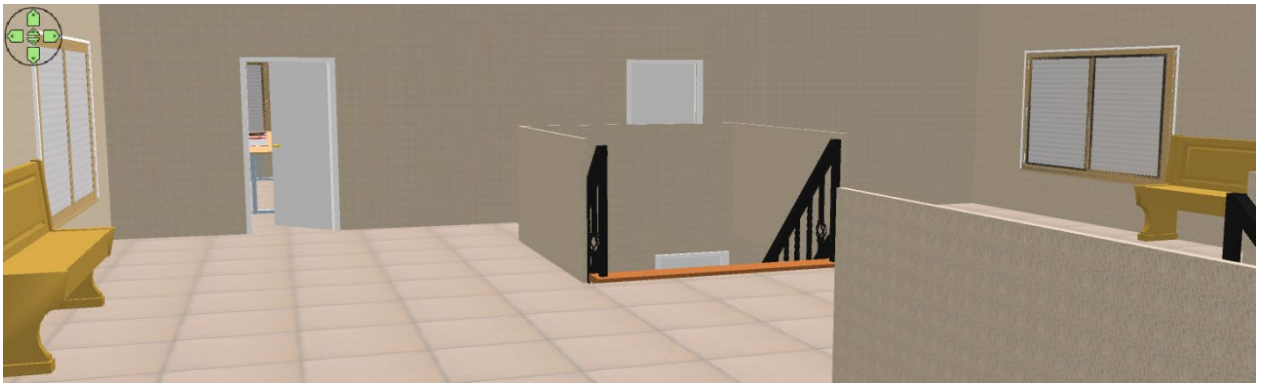


Figura 4. Interior edificio.

Por otra parte, se ha creado un segundo edifico. Se trata de una vivienda amueblada, y está destinado al caso de uso empresarial. Este edificio posee una única planta, con un amplio salón-comedor, un dormitorio principal con baño, otros dos dormitorios, un segundo baño y una cocina. El fin de este edificio es que, tras ser exportado a Unity, se pueda interactuar con los elementos de la casa, obteniendo información sobre estos, por lo que es un ejemplo de uso empresarial de este trabajo.

En las Figura 5, 6 y 7 se muestran varias imágenes de la vivienda.

Figura 5. Vista exterior.

Figura 6. Salón.

Figura 7. Comedor.

En el Anexo I se explica, en detalle, cómo se han creado con Sweet Home 3D estos dos edificios.

3.2 Importación de edificios creados en Sweet Home 3D a Unity

Se va a mostrar en este apartado el procedimiento que permite importar en Unity edificios creados con Sweet Home 3D [23]. En primer lugar, hay que exportar el edificio que hayamos realizado en Sweet Home 3D. Para ello, debemos ir a Vista 3D, seleccionar vista aérea y también seleccionar mostrar todos los niveles. A continuación, hay que hacer clic en Exportar a formato OBJ… (Figura 8).

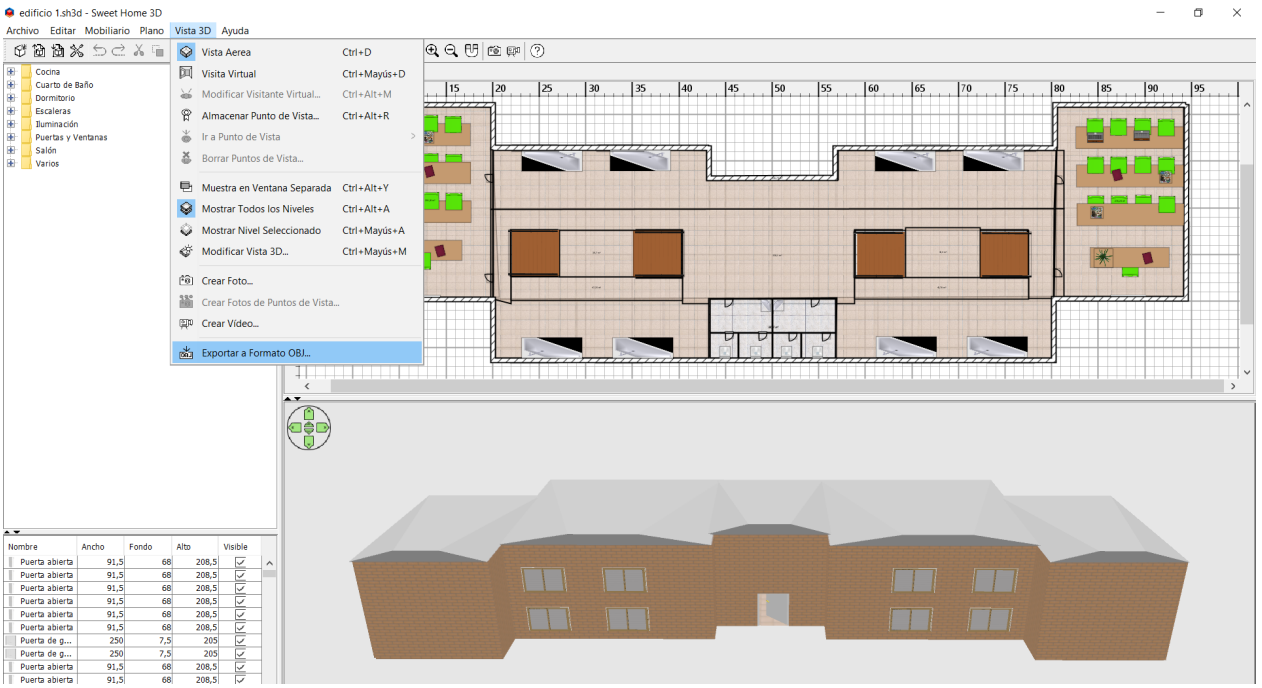


Figura 8. Selección de opciones en el menú para la exportación en Sweet Home 3D.

Debemos seleccionar la ubicación de la exportación, y para ello es importante seleccionar una carpeta que no contenga ningún fichero, ya que en esa carpeta se van a guardar todos los ficheros necesarios para poder posteriormente importar el edificio en otros programas (Figura 9).

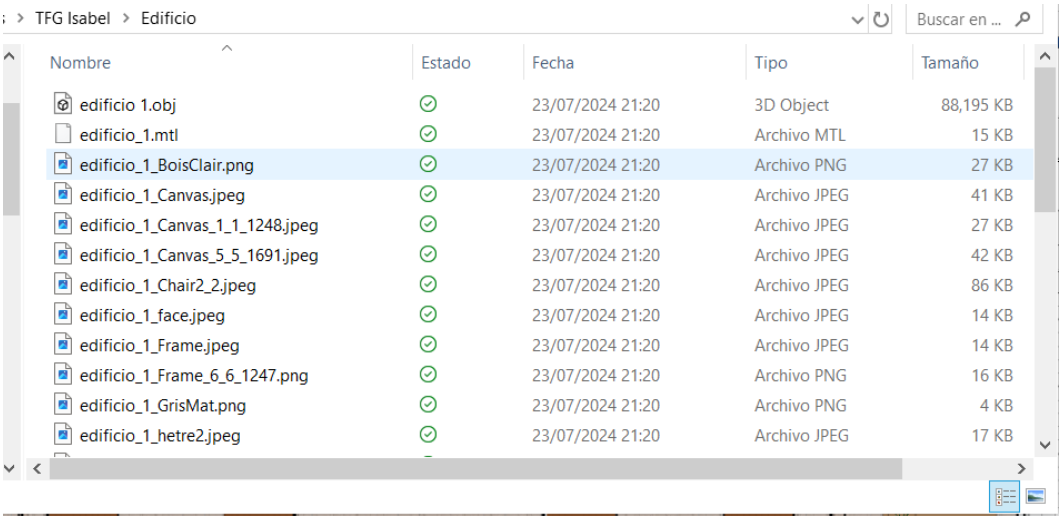


Figura 9. Ficheros generados tras la exportación en Sweet Home 3D.

Tras esto ya se puede abrir Unity para proceder a importar el edificio. En primer lugar, creamos un proyecto nuevo, que en este caso será del tipo Universal Render Pipeline (URP) por la mayor compatibilidad que ofrece con diferentes plataformas respecto de otros tipos de proyectos (Figura 10).

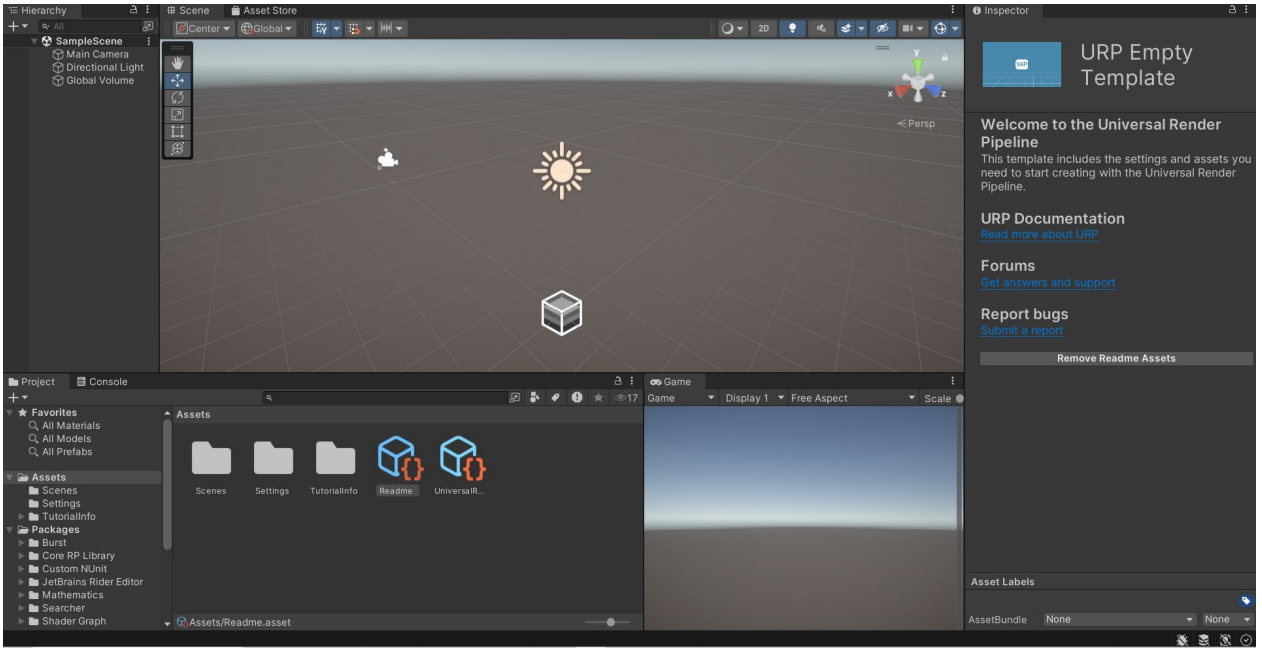


Figura 10. Proyecto URP creado en Unity.

A continuación, vamos a arrastrar la carpeta donde teníamos los ficheros generados al exportar el edifico desde Sweet Home, y llevamos esa carpeta a la de Assets. En este caso esa carpeta se llama Edificio.

Ahora colocamos un terreno (GameObject --> 3D Object --> terrain). Así podremos colocar encima del terreno el edificio.

Como por defecto el tamaño del terreno es de 1000x1000, podemos centrarlo colocando x=-500 e y=-500 en el Inspector del Terrain (Figura 11).

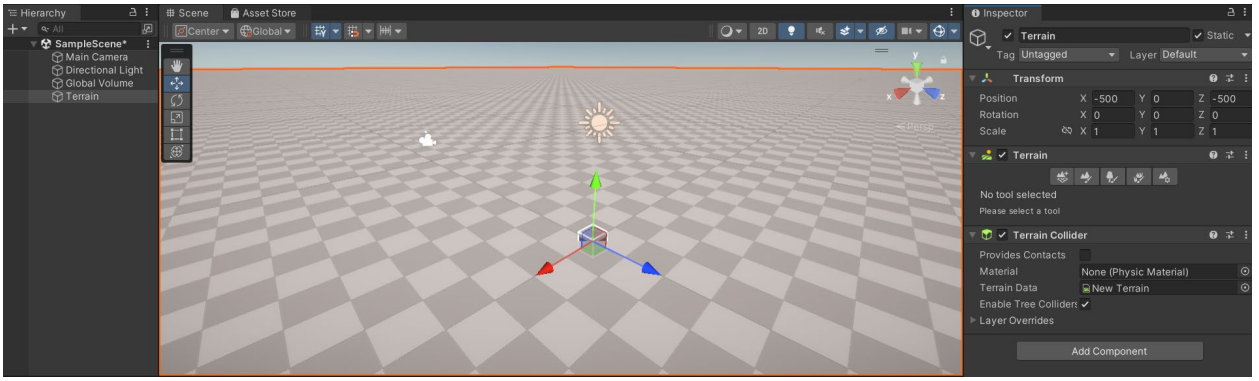


Figura 11. Creación y centrado del terreno.

Para colocar el edificio, abrimos la carpeta “Edificio”, y tendremos que arrastras uno de los ficheros que se encuentra en el interior hasta la escena, pero antes vamos a seleccionar ese fichero, y en el Inspector vamos a cambiar su escala, ya que el edificio tiene unas dimensiones muy grandes, y es necesario reducirlas. En este caso ponemos un Scale Factor de 0.01 (Figura 12).

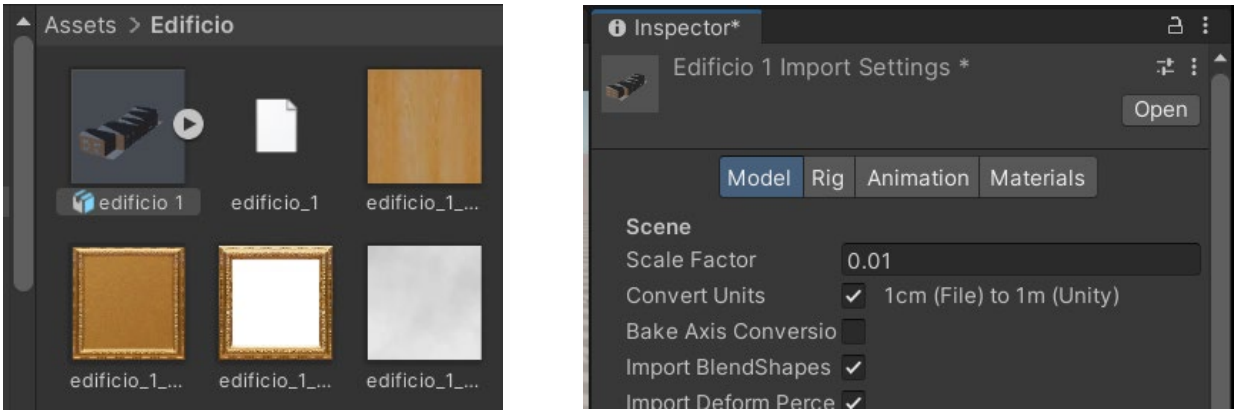


Figura 12. Cambio de la escala del edificio.

Ese fichero que hemos seleccionado (Edificio1.obj) lo tenemos que arrastrar hasta la escena. Posteriormente deberemos mover el edificio y la cámara, hasta que se pueda ver correctamente el edificio desde el punto de vista deseado (Figura 13).

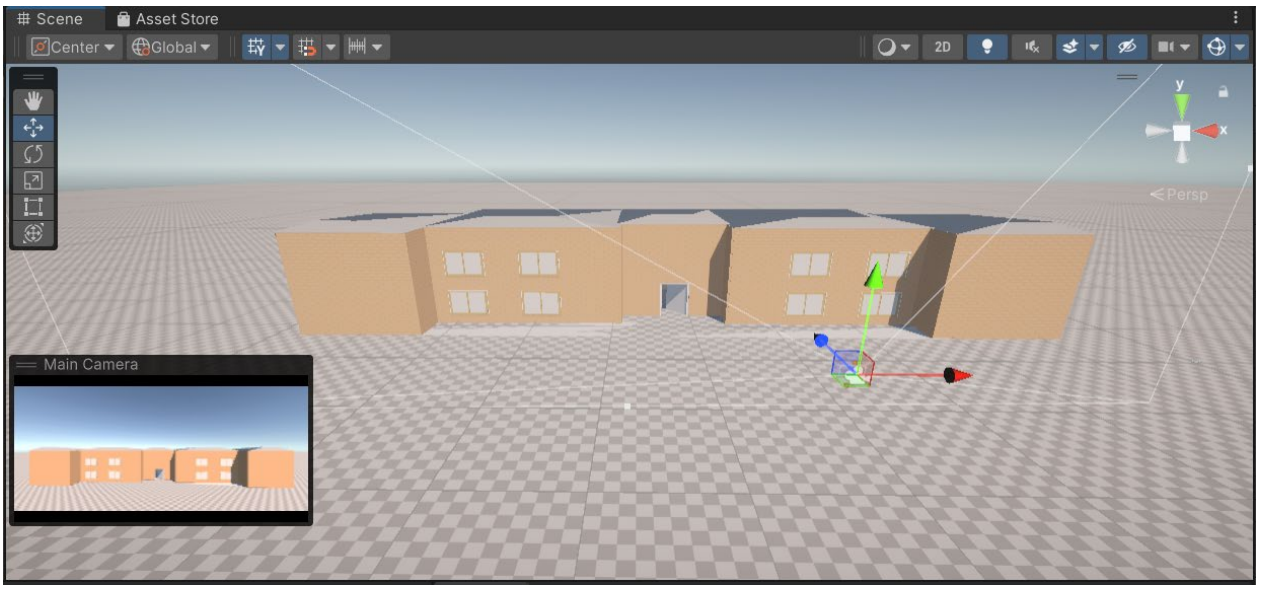


Figura 13. Colocación del edificio y la cámara.

Además, con el fin de que la parte baja del edificio no interfiera con el terreno, vamos a elevar un poco el edificio, tal y como se muestra en la Figura 14, donde la hemos elevado colocando 0.01 en el eje y, para hacer esto tiene que estar seleccionado el edificio.

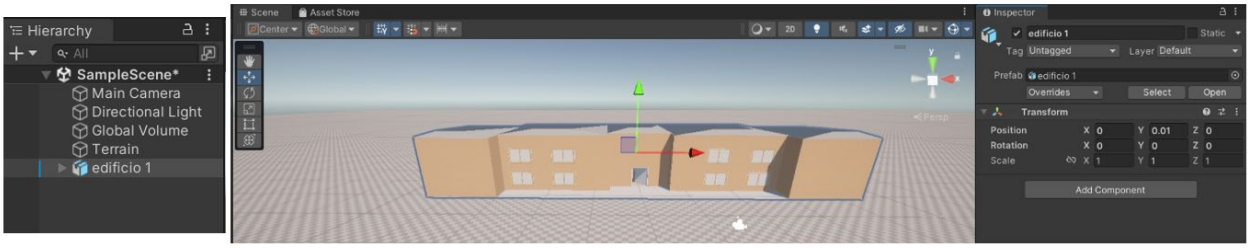


Figura 14. Elevación del edificio.

A continuación, y teniendo el edificio seleccionado, vamos a añadir componentes (Add Component). En primer lugar, añadiremos el Rigidbody, Para ello tenemos que hacer clic en “Add Component”, y buscar RigidBody, y cuando lo tengamos añadido, marcar la casilla Is Kinematic (Figura 15).

Un Rigidbody en Unity es un componente que permite que un objeto sea afectado por la física, como la gravedad, las colisiones, y otras fuerzas físicas. Sin un Rigidbody, el objeto no reaccionará físicamente.

Cuando marcas la opción "Is Kinematic" en un Rigidbody, le estás indicando a Unity que el objeto será controlado por ti (a través de scripts) y no por el motor de física de Unity. Esto significa que:

- El objeto no será afectado por las fuerzas físicas (como la gravedad o las colisiones).
- El objeto no interactuará físicamente con otros objetos a menos que se programe específicamente.

Al marcar "Is Kinematic", te aseguras de que el edificio permanezca en su lugar sin ser afectado por colisiones o gravedad. Si no marcas "Is Kinematic", el edificio podría reaccionar a las colisiones de una manera no deseada, por ejemplo, cayendo o desplazándose cuando otros objetos chocan con él.

Además, se reduce la carga de cálculo del motor de física de Unity, mejorando el rendimiento general.

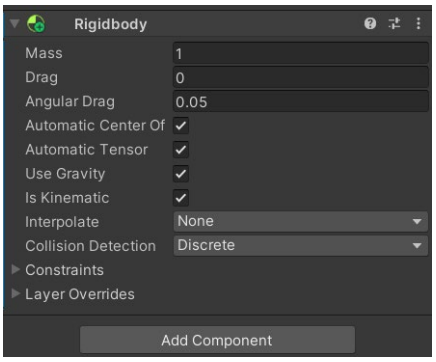


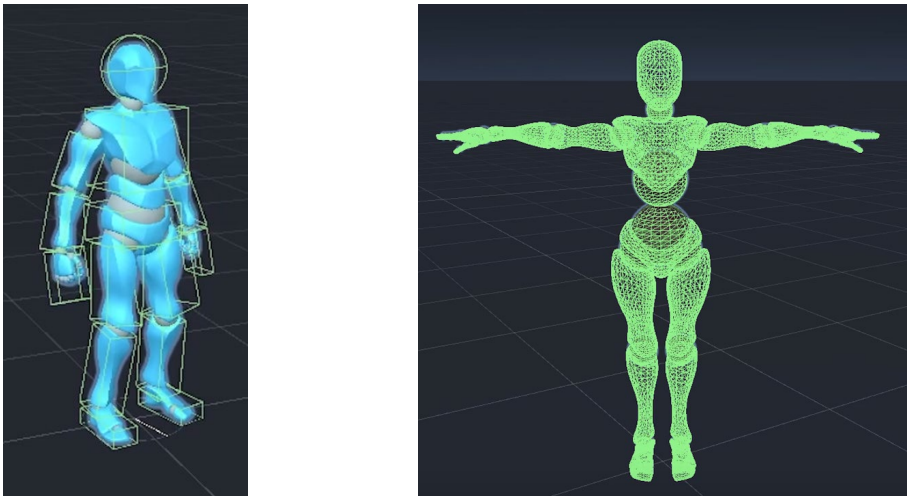
Figura 15. Opciones de Rigidbody para el edificio.

Por último, para poder entrar con un personaje en el edificio e interaccionar con todos los objetos que se encuentran en el interior, ya que Unity ha importado todos los elementos, es necesario seleccionar todos los elementos que se encuentran dentro del edificio y tenemos que aplicarles un colisionador. Si no se hace esto el personaje atravesará los distintos objetos, y no es ese el resultado deseado.

En Unity, los colisionadores son componentes que se añaden a los objetos para gestionar las colisiones y la detección de contacto. Dos tipos comunes de colisionadores son el Box Collider y el Mesh Collider.

- Box Collider**:** Adecuado para colisiones simples. Se usa cuando la forma del objeto se puede aproximar bien a una caja y no se necesita una precisión alta en la colisión.
- Mesh Collider**:** Ofrece colisiones precisas para formas complejas pero consumiendo más recursos computacionales.

En la figura 16 se muestra la diferencia que existe entre usar un tipo de Collider u otro.



Box Collider Mesh Collider

Figura 16. Diferencia entre Box Collider y Mesh Collider. Fuente: [24]

Aunque en este caso, para los diferentes elementos que forman el edificio, podría ser adecuado utilizar el Box Collider, realmente el correcto es el Mesh Collider, ya que en otro caso quedarían bloqueadas las puertas a causa de cómo crea Sweet Home 3D las paredes y puertas. Para aplicar Mesh Collider, es necesario seleccionar todos los elementos que forman el edificio, y añadir en el Inspector un Mesh Collider. Recordemos que anteriormente hemos aplicado Rigid Body al Edificio.

Finalmente, podemos aplicar una textura al terreno, siguiendo los siguientes pasos [25]:

Paso 1: Selecciona el Terrain en la jerarquía (hierachy) de la escena.

Paso 2: En el Inspector, dentro del componente Terrain, despliega la sección Paint Terrain.

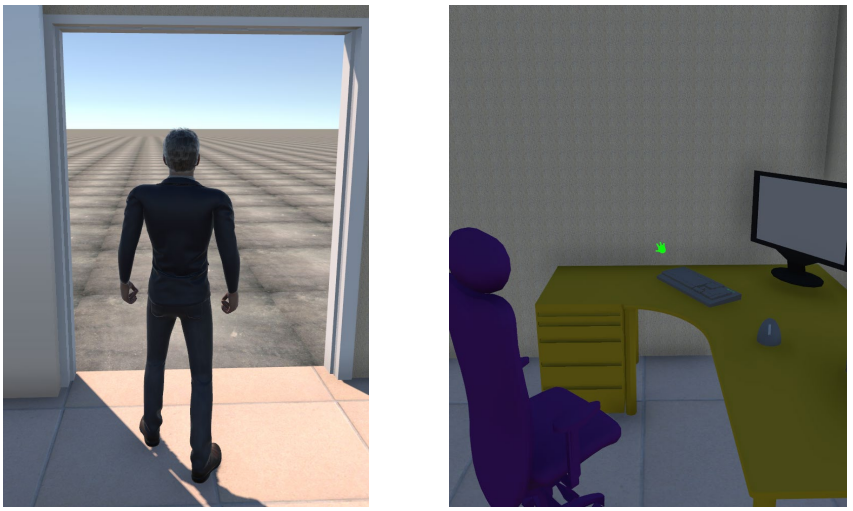
Paso 3: En el panel de Terrain, selecciona Edit Terrain Layers y luego elige Create Layer.

Paso 4: Aparecerá un cuadro de diálogo para seleccionar una textura o material. En este caso se ha seleccionado para el terreno “Floor Cement.jpg”

Paso 5: La textura seleccionada debería aplicarse automáticamente a todo el terreno.

3.3 Control del personaje en tercera y primera persona

En este apartado se muestra cómo poder pasar de tercera a primera persona, y viceversa [26]. En tercera persona el personaje aparece en pantalla, mientras que en primera persona el personaje no aparece (Figura 17). Es lo que se suele denominar como cambio de perspectiva.



Tercera persona Primera persona

Figura 17. Diferencia entre primera y tercera persona.

Lo primero que hay que hacer es acceder a la Asset Store de Unity [13], o bien desde la página web o desde el menú del propio programa que te redirige a la misma página.

En este trabajo se ha utilizado el “Starter Assets -ThirdPerson” [27] (Figura 18), que está disponible en el Asset Store de Unity, para controlar el personaje. Se ha utilizado este Asset porque facilita enormemente la implementación del control del personaje. Aunque en principio este Asset está pensado para controlar en tercera persona al personaje, se ha logrado que también sirva para el control en primera persona, siendo posible cambiar de primera persona a tercera, y viceversa, pulsando la tecla Q del teclado.

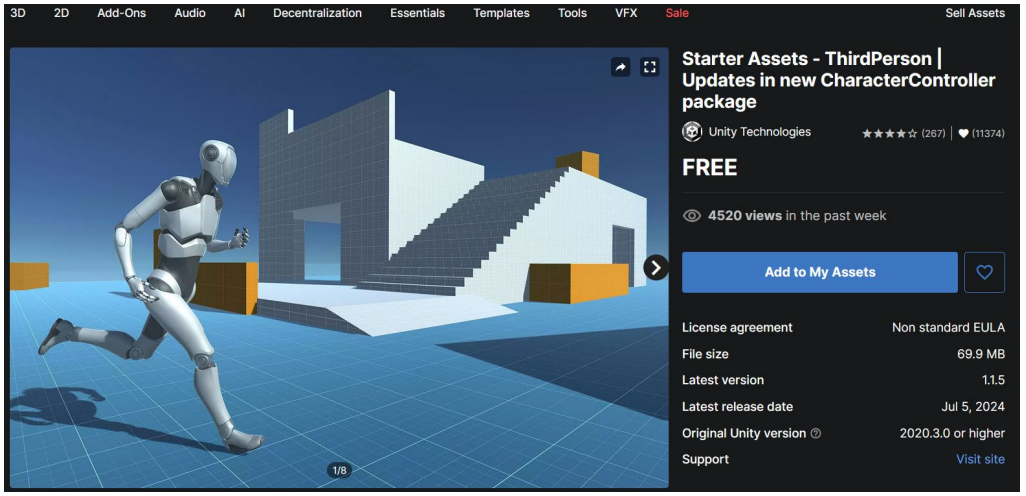


Figura 18. Starter Assets – Third Person. Fuente: [27]

A continuación, se debe seleccionar la opción de “Add to My Assest” (Figura 18), y en la siguiente pestaña (Figura 19) hay que clicar en “Open in Unity”.

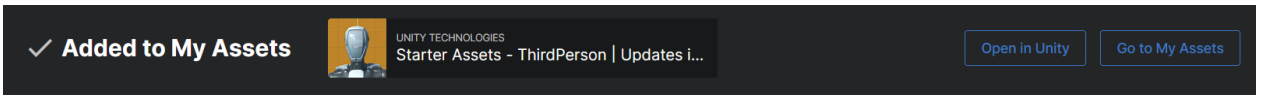


Figura 19. Added to My Assets. Fuente: [27]

Se abrirá el Package Manager (Figura 20), y hay que hace clic sobre “Download”.

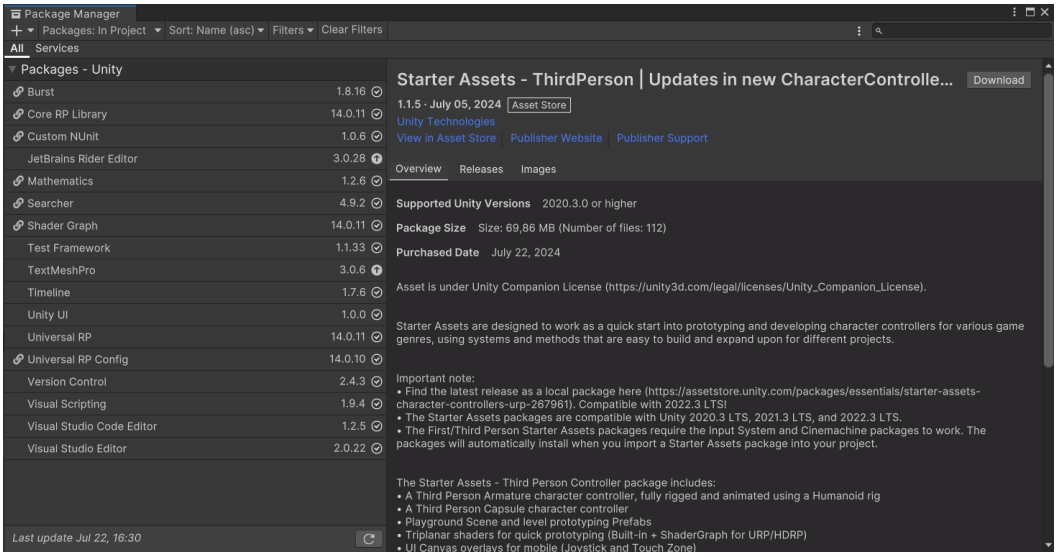


Figura 20. Package Manager.

Y a continuación se debe hacer clic sobre “import”.

Es posible que pida instalar algunos paquetes necesarios para que funcione correctamente el Asset, en ese caso (Figura 21) hay que hacer clic en “Install/Upgrade”.

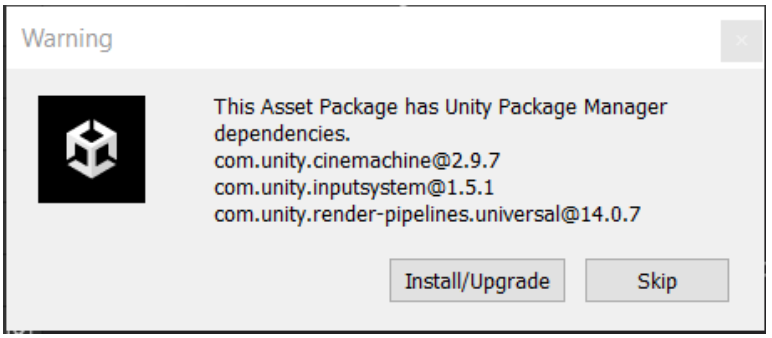


Figura 21. Instalación de paquetes adicionales. Elaboración propia.

Finalmente hay que importar el Asset, haciendo clic en “Import”.

Tras la instalación se reiniciará el editor de Unity. Y ya se podrá usar el Asset.

Si todo ha ido bien, deberá haber aparecido una nueva carpeta dentro de la carpeta Assets con el nombre “StarterAssets”, (Figura 22). Si no está esa carpeta, se debe repetir el procedimiento anterior de instalación, ya que nos habremos saltado algún paso.

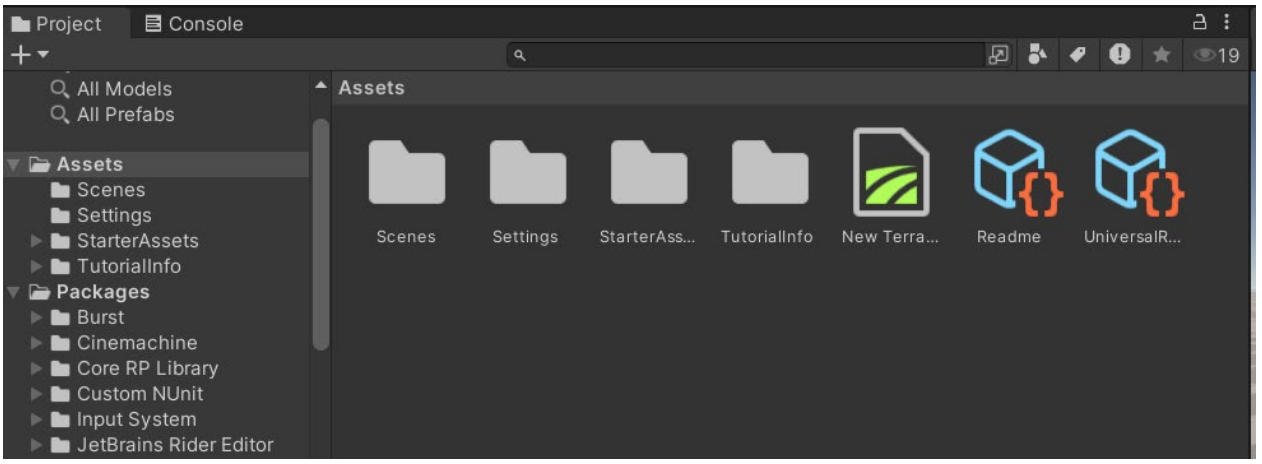


Figura 22. Carpeta Assets.

En el Anexo II se detalla el procedimiento que se ha seguido en este trabajo para controlar al personaje, tanto en tercera como en primera persona.

3.4 Interacción con objetos

En este apartado se va a describir cómo podemos interactuar con objetos usando un RayCast. Para ello, al objeto con el que se quiere interactuar hay que añadirle un tag y un layer. En Unity, los conceptos de tag y layer son herramientas fundamentales para organizar y gestionar los objetos dentro de una escena.

Un tag es una cadena de texto que puedes asignar a uno o varios objetos en Unity. Los tags son útiles para identificar y agrupar objetos que comparten características similares, facilitando la gestión y las interacciones en el juego. Aquí hay algunos usos comunes:

Identificación: Permiten identificar fácilmente un grupo de objetos en la escena.

Interacción: Puedes escribir código que detecte colisiones o interacciones con objetos específicos basados en su tag.

Un layer (capa) es una forma de agrupar objetos para la gestión de visibilidad, colisiones y renderización.

Usar tags y layers facilita la escritura de código más limpio y eficiente. En lugar de comparar nombres o

propiedades de objetos, se pueden utilizar tags y layers para identificar y gestionar grupos de objetos de manera más directa y organizada. Además, configurar correctamente las capas para las colisiones y la renderización puede reducir significativamente la carga computacional, mejorando el rendimiento.

Por lo tanto, los tags y layers son herramientas muy útiles en Unity para organizar y gestionar la interacción y el comportamiento de los objetos, haciendo que el desarrollo sea más eficiente y el rendimiento sea mejor.

En el Anexo III se detalla la metodología aplicada en este trabajo para interactuar con objetos, mostrando un ejemplo de cómo puede interaccionarse con un cubo mediante scripts de lenguaje C#.

En lugar de utilizar tags y layers, otra posibilidad para interaccionar con objetos es utilizar un trigger. En este trabajo también se ha aplicado esta técnica. En concreto se ha utilizado para abrir automáticamente la puerta corredera principal de uno de los dos casos. La puerta se abre automáticamente cuando el personaje se acerca suficientemente a ella, simulando una situación de la vida real en la que un sensor detecta a una persona que se acerca a una puerta, y ésta se abre automáticamente, y tras el paso de la persona se cierra. En el Anexo IV se explica cómo se puede llevar a cabo la interacción con una puerta mediante un trigger [28], tal y como se ha realizado en este trabajo.

3.5 Cambio del skin del personaje

En este apartado se indica cómo puede cambiarse el Skin del personaje en Unity [29], sustituyéndolo por uno de los que ya existe en Mixamo [3]. Esta es una página web que pertenece a Adobe, y desde donde es posible descargar skins en varios formatos, siendo uno de ellos compatible con Unity. Además, aunque no es el objeto de este documento ni de este trabajo, también se pueden descargar animaciones concretas que se deseen aplicar a nuestro personaje, pero en nuestro caso es suficiente con el movimiento de andar y correr que por defecto posee al utilizar el Starter Asset - ThidPerson. También sería posible utilizar skins personalizados, que pueden crearse previamente con Blender, o de forma más sencilla utilizando la web de Ready Player Me [30]. En este trabajo nos hemos limitado a utilizar skins disponibles en la página de Mixamo.

En primer lugar, debemos ir a la página de Mixamo, y para poder acceder a los skins deberemos logearnos, para lo que necesitaremos una cuenta gratuita de Adobe. Posteriormente ya podremos acceder a las skins.

Dentro de la página debemos seleccionar la pestaña “Characters” (Figura 23).

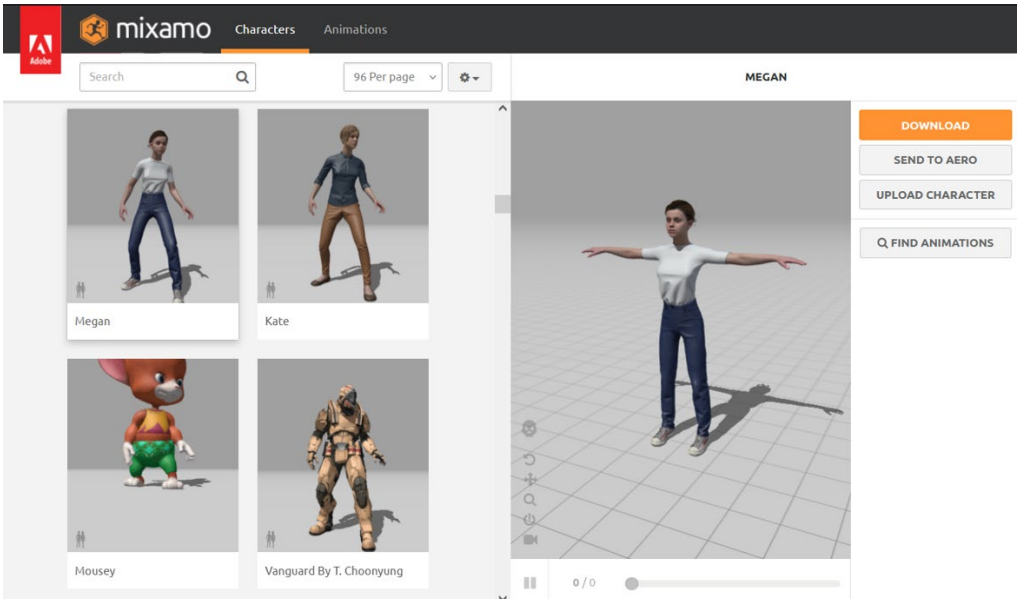


Figura 23. Personajes disponibles en Mixamo.

Tendremos varios skins entre los que podremos elegir el que nos parezca más adecuado para nuestro proyecto. En la Figura 23 se ha seleccionado el de Megan. A continuación haremos clic con el botón izquierdo del ratón en “Download”, y en el cuadro de diálogo que aparecerá en pantalla seleccionaremos el Formato FBX for Unity(.fbx), dejaremos Pose por defecto en T-Pose, y descargaremos el fichero .fbx haciendo clic sobre “Download”. El fichero descargado tiene un nombre con números y letras, pero podemos renombrarlo, por ejemplo, a Megan.fbx. Podemos descargar varios skins para realizar pruebas en nuestro proyecto. En este caso se han descargado tres, que son lo que se muestran en la Figura 24.

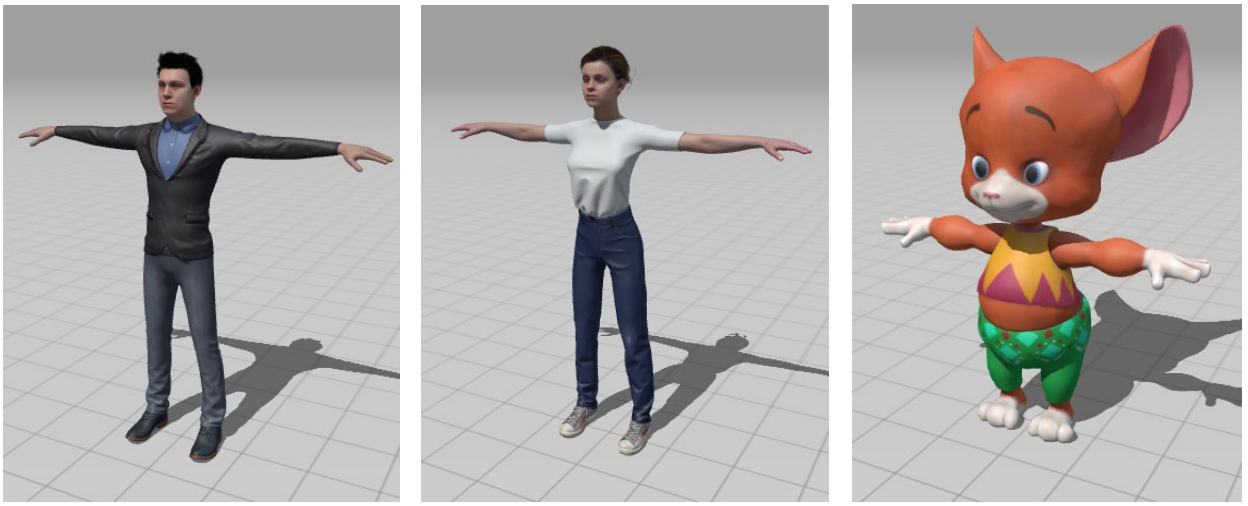


Figura 24. Skins descargados.

En el anexo V se detalla el procedimiento que se ha seguido en este trabajo para poder cambiar el skin del personaje por uno descargado desde Mixamo [29].

3.6 Creación de un ejecutable

En este apartado se detalla, en cinco pasos, cómo crear un fichero ejecutable en Unity [9], de manera que el proyecto realizado pueda ser ejecutado sin necesidad de disponer del código fuente ni tener instalado Unity.

Paso 1: Abrir las Configuraciones de Construcción (Build Settings):

En File → Build Settings... en la barra de menú superior, aparecerá la pestaña que se muestra en la Figura 25.

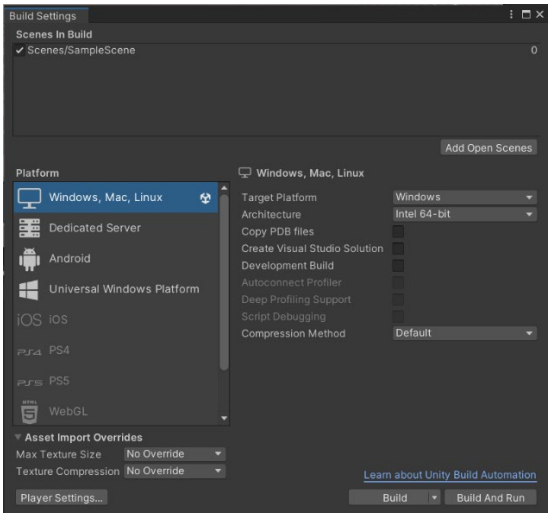


Figura 25. Build Setting.

Paso 2: Configurar la Plataforma:

En Build Settings, asegúrate de que PC, Mac & Linux Standalone esté seleccionado.

En la sección Platform, selecciona Windows, Architecture debe estar en x86_64 para sistemas de 64 bits (que es lo más común).

Si es necesario, también puedes seleccionar x86 para 32 bits.

Paso 3: Añadir Escenas a la Build:

En la ventana Build Settings, verás una lista llamada Scenes in Build.

Asegúrate de que las escenas que deseas incluir en la build están marcadas. Si falta alguna, usa el botón Add Open Scenes para añadir la escena actualmente abierta.

Paso 4: Configurar Opciones Adicionales (Player Settings):

Haz clic en Player Settings en la parte inferior de la ventana de Build Settings (Figura 26). Allí puedes configurar opciones adicionales como:

- Product Name: Nombre del proyecto/juego.
- Icono: Icono personalizado para tu ejecutable.
- Resolución por defecto, pantalla completa, etc.

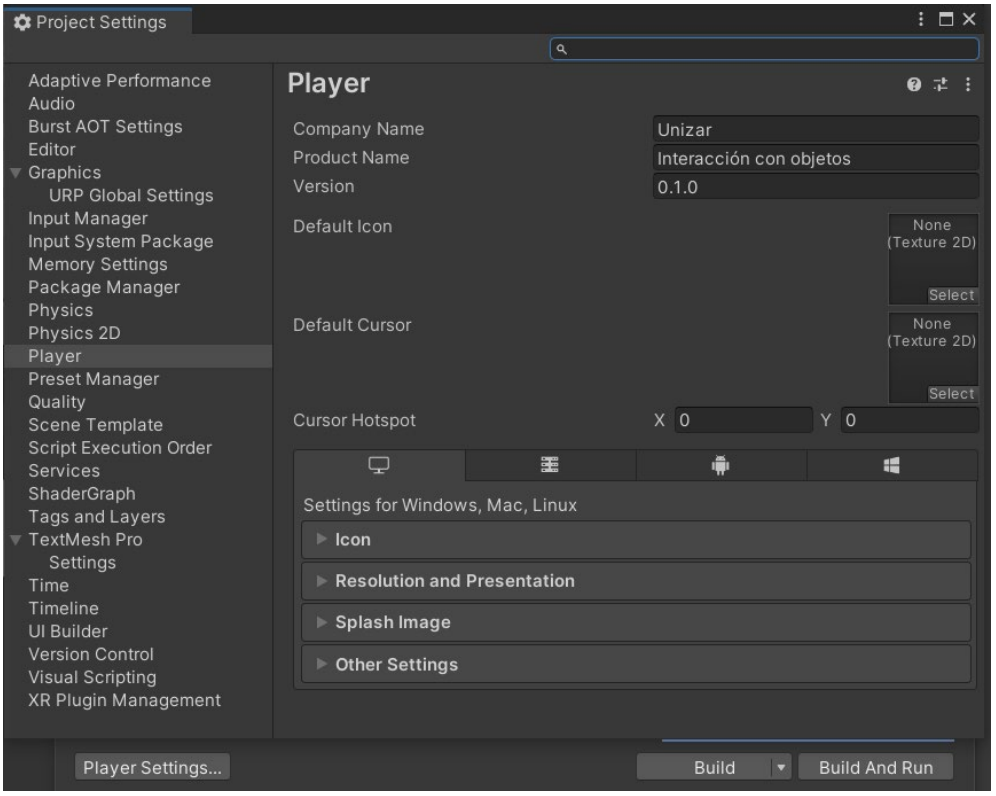


Figura 26. Proyect Settings.

Paso 5: Crear el Ejecutable:

Una vez que todo esté configurado, haz clic en Build o Build and Run.

Selecciona la ubicación donde quieres guardar el ejecutable, y Unity comenzará a compilar el proyecto.

Una vez terminado, encontrarás un archivo .exe junto con una carpeta de datos ([Your Project Name]_Data) en la ubicación seleccionada.

3.7 Finalización del programa

Se ha creado un script para poder salir de la ejecución del programa mediante la tecla ESC del teclado.

El script se denomina ExitOnEsc.cs, y se encuentra disponible en el anexo VI.

4. Casos desarrollados

4.1 Uso Docente

Para el caso destinado a uso docente, se ha utilizado uno de los edificios creados con Sweet Home 3D, que es una versión simplificada del edificio Torres Quevedo del Campus Río Ebro de la Universidad de Zaragoza. El usuario, a través del manejo de un personaje, puede entrar a este edificio y explorarlo. Tal y como se ha comentado anteriormente la entrada al edificio se realiza mediante una puerta que automáticamente se abre cuando el personaje se acerca a ella, tal y como se muestra en la Figura 27.

Figura 27. Entrada al edificio docente.

En el interior del edificio se han colocado algunos objetos típicos de un edificio docente, como, por ejemplo, una máquina expendedora de comida, tal y como se muestra en la Figura 28.

Figura 28. Máquina expendedora.

También se encuentran, en el interior del edificio varias aulas y salas de estudio (Figura 29)

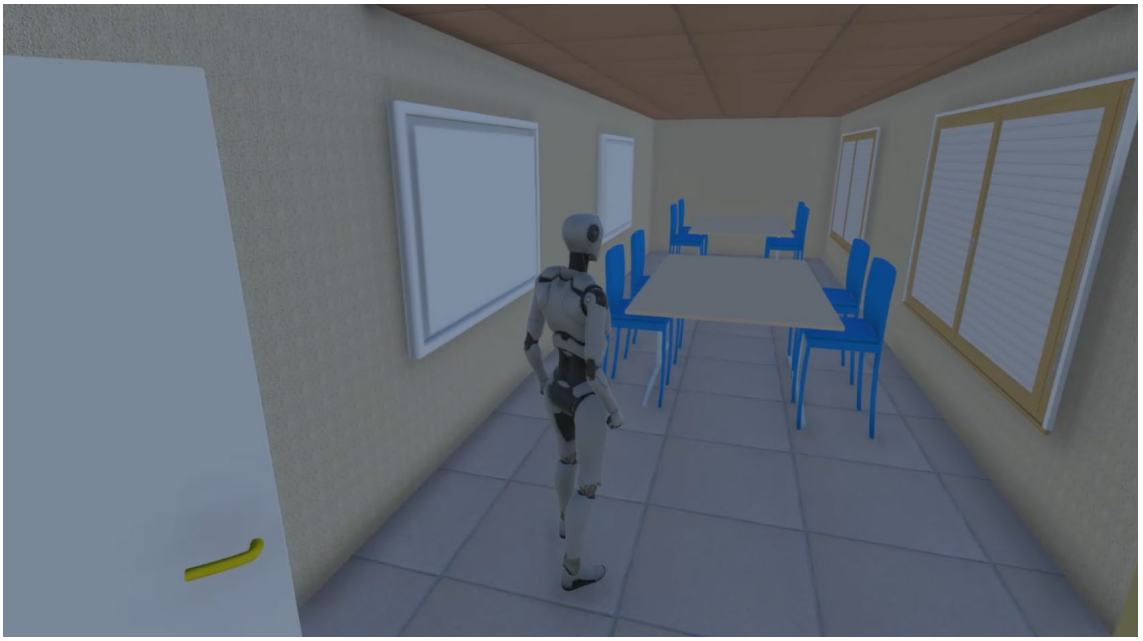


Figura 29. Aula del edificio docente.

Y, por supuesto, era necesario colocar baños en el edificio (Figura 30).

Figura 30. Baños edifico docente.

En una de las aulas, de la segunda planta, hay un cartel al lado de la puerta que indica que en su interior se puede encontrar información sobre Bauhaus, que es un tema relacionado con la historia del diseño y de gran relevancia (Figura 31).

Figura 31. Interior edificio docente entrada aula Bauhaus.

Dentro de este aula el usuario se encuentra con objetos en forma de paneles en cuya superficie se indican los títulos de diferentes temas sobre la historia de Bauhaus. Estos son los objetos con los que se puede interactuar (Figuras 32 y 33)

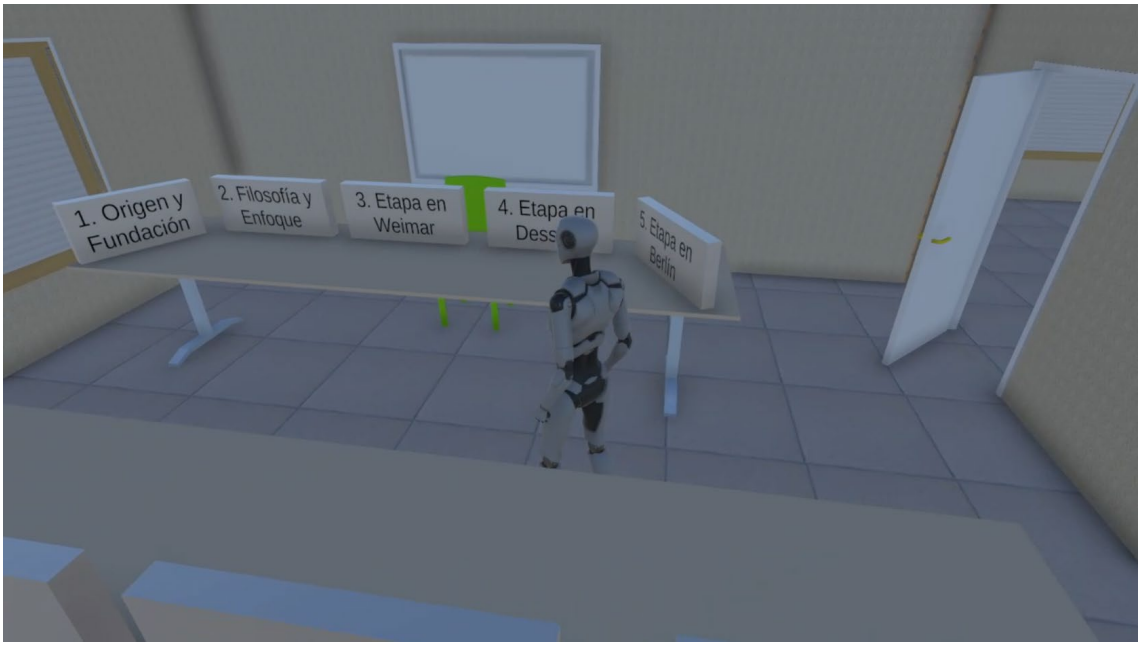


Figura 32. Aula Bauhaus temario 1.

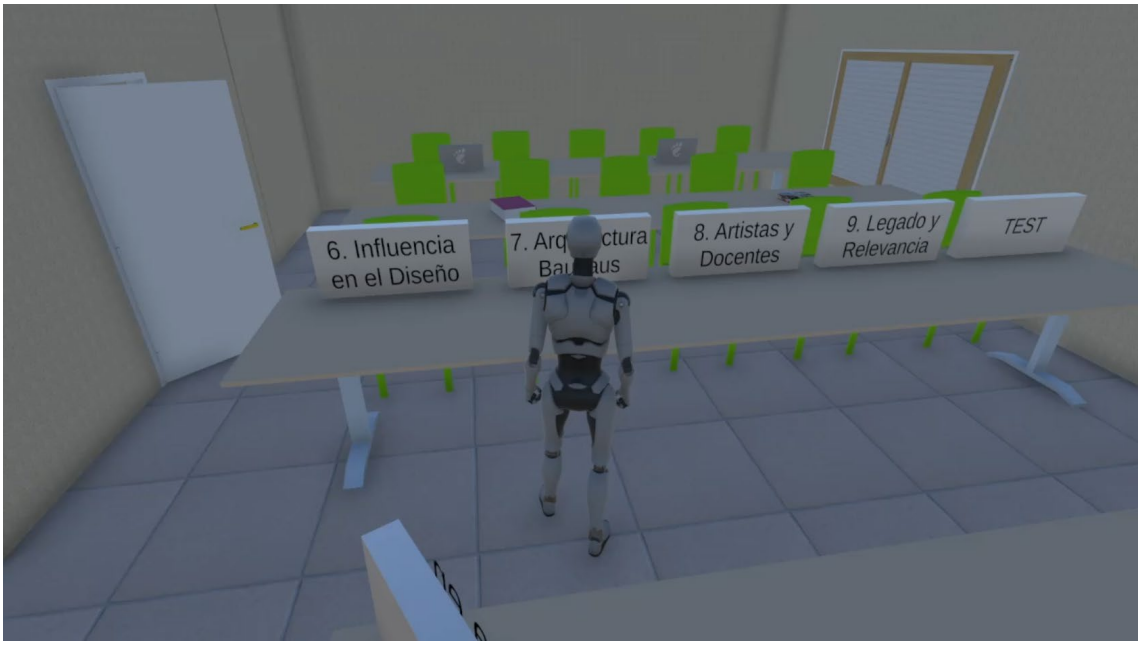


Figura 33. Aula Bauhaus temario 2.

Llevando al personaje cerca de estos objetos, aparece en pantalla el mensaje “Pulsa la tecla E”, y además cambia el color del objeto con el que se puede interactuar, para que así sea más intuitivo para el usuario el uso de la herramienta (Figura 34).

Figura 34. Selección de temario.

Si el usuario pulsa la tecla E, se muestra en pantalla la diapositiva del apartado correspondiente, proporcionando información sobre ese apartado (Figura 35). Estas diapositivas se encuentran en el anexo VII.

Figura 35. Diapositiva temario.

Finalmente, se puede acceder a un test en el que hay que responder a varias preguntas sobre la información proporcionada en los objetos con los que se ha interactuado (Figura 36).

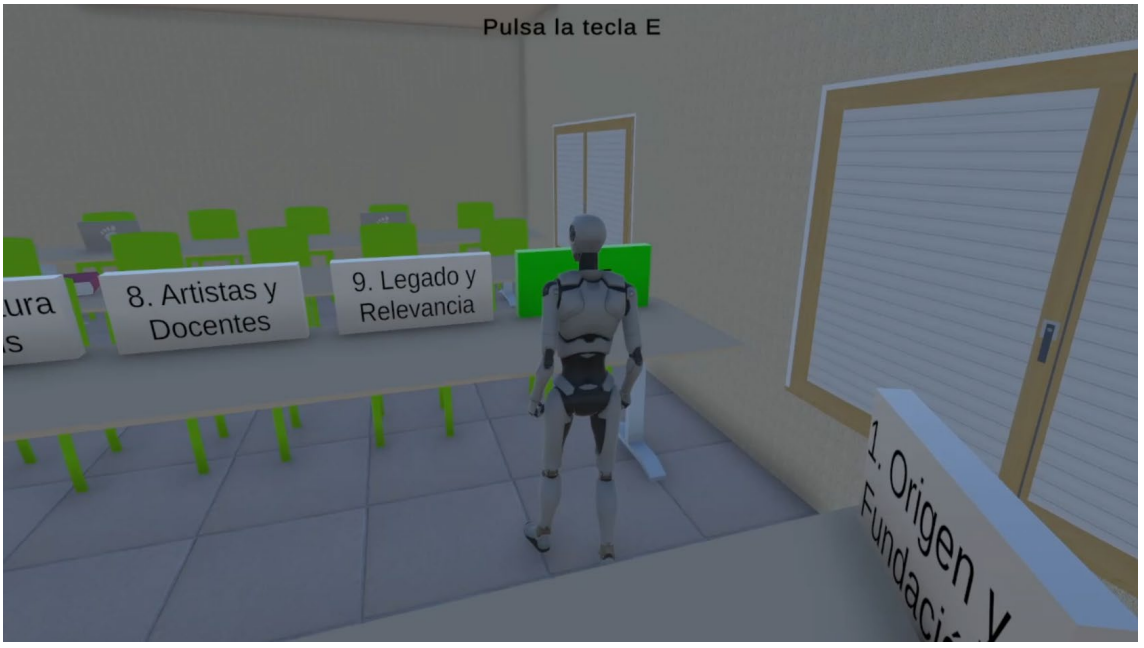


Figura 36. Selección de test.

Durante la realización del test se muestran, una tras otra, imágenes de diapositivas con una pregunta y cuatro posibles respuestas (Figura 37). Cuando el usuario selecciona una de las posibles respuestas, se muestra en pantalla la que es correcta, y una vez respondidas todas las preguntas se muestra el número de aciertos que se han obtenido. Todas las diapositivas con las preguntas y posibles respuestas se encuentran disponibles en el Anexo VII.

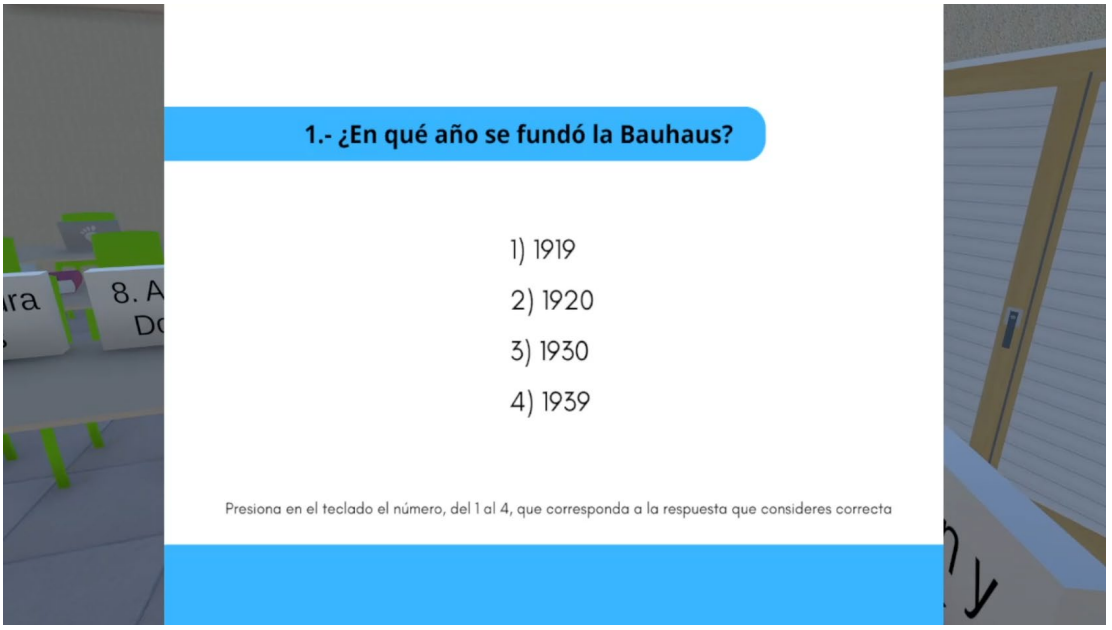


Figura 37. Diapositiva test.

De esta manera se le da a este proyecto un fin educativo. Este caso que se ha desarrollado es un ejemplo concreto, pero se puede adaptar esta metodología de aulas virtuales a diferentes temáticas, haciendo que cada una de las aulas se destine a un tema diferente. Además, se podría mostrar información en otros formatos, como, por ejemplo, mediante vídeos o audios.

Así, se logra que sea más amigable el proceso de aprendizaje, al integrar un método innovador utilizando una estructura que se asemeja a los videojuegos, lo que permite captar mejor la atención de los usuarios, superando a otros métodos más convencionales de enseñanza. Además, fomenta la autonomía del usuario en el proceso de aprendizaje. Especialmente podría ser utilizado en las escuelas para motivar a los niños a aprender de una forma más interactiva y llamativa para ellos.

4.2 Uso Empresarial

El caso para el uso empresarial consiste en una casa modelada y amueblada, en la cual el usuario puede entrar e inspeccionarla a través del manejo de un personaje (Figuras 38, 39).

Figura 38. Entrada a la casa.

Figura 39. Salón.

Dentro de la vivienda se encuentran diferentes objetos, con los que se va a poder interactuar, de modo que si el personaje se acerca, por ejemplo, a la televisión del salón, aparece en pantalla el mensaje “Pulsa la letra E” (Figura 40).

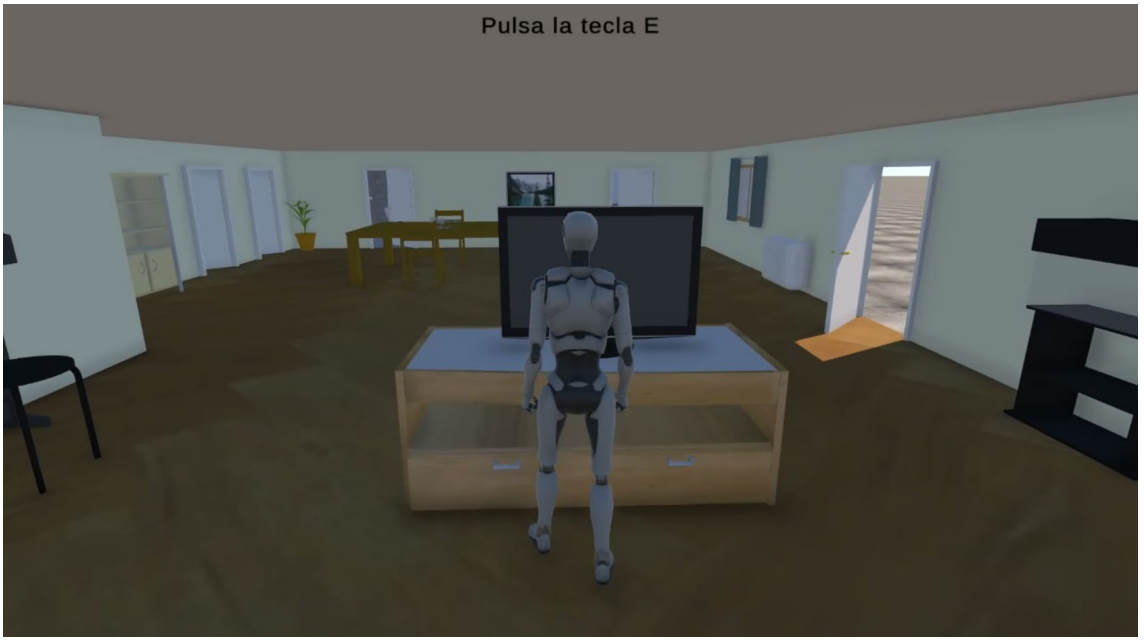


Figura 40. Interacción con un objeto.

Al pulsar la letra E para interaccionar con el objeto aparecerá en pantalla una diapositiva con información sobre el producto con el que se ha interactuado, de manera que el usuario recibe información sobre un objeto de la casa (Figura 41). Todas las diapositivas con información sobre productos se encuentran disponibles en el Anexo VIII.

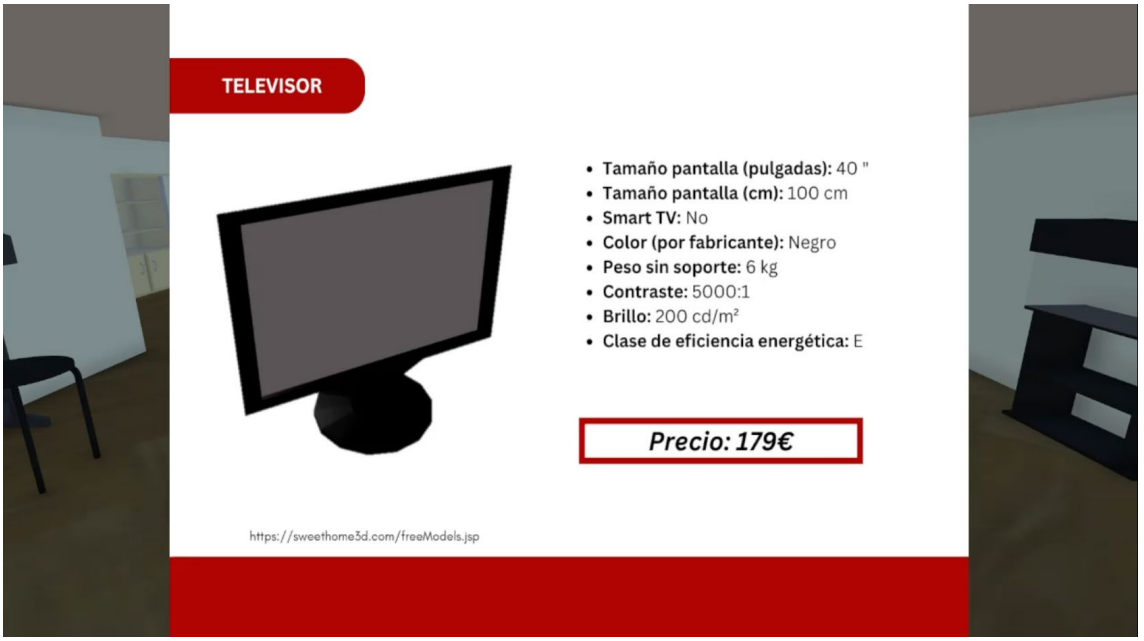


Figura 41. Información de la televisión.

Tal y como se ha comentado anteriormente, en este trabajo es posible cambiar de tercera a primera persona, y viceversa. En la Figura 42 se muestra cómo se puede interaccionar con un objeto en primera persona, y obtener información sobre él (Figura 43).

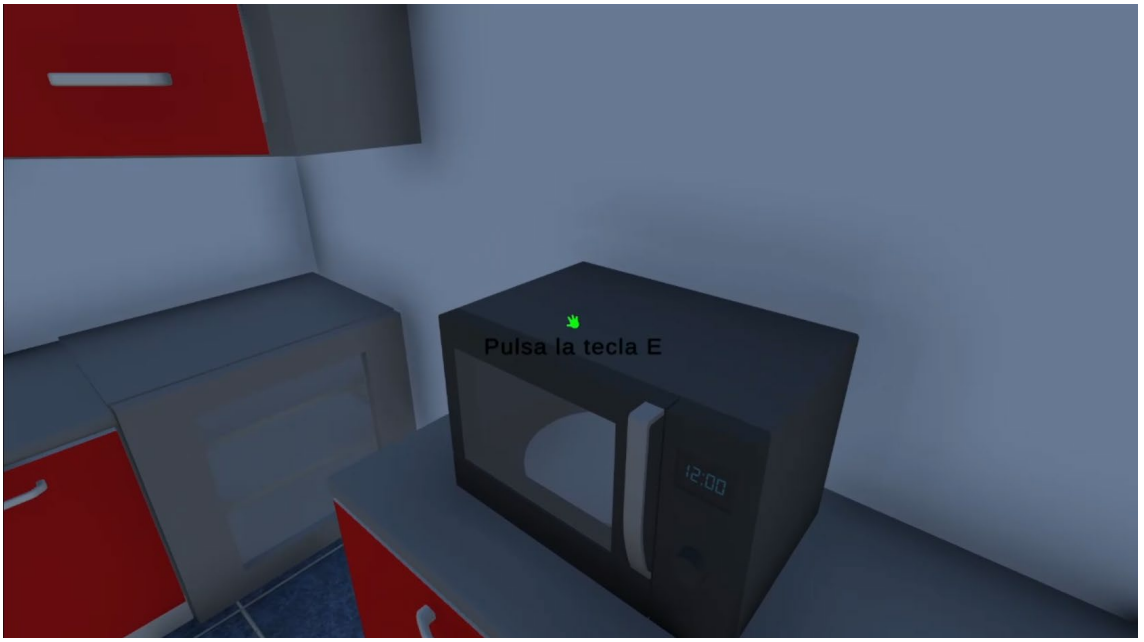


Figura 42. Interacción en primera persona.

Figura 43. Información del microondas.

De esta forma se tiene un entorno virtual que puede ser usado para fines comerciales y empresariales. La idea que se ha desarrollado en este trabajo es una muestra de cómo una empresa podría presentar de manera virtual e innovadora una gama de productos, por medio de un entorno en el que los usuarios pudieran interactuar con un modelo virtual de éstos, y así de esta forma obtener información.

5. Conclusiones

Finalmente, con este proyecto se han conseguido los objetivos propuestos en un inicio, se han diseñado y desarrollado dos prototipos de entorno virtual 3D interactivos, que proporcionan a los usuarios una experiencia inmersiva y enriquecedora, donde estos pueden encontrar diversas estancias que permiten una interacción intuitiva con diferentes elementos. Cada uno está destinado a un fin diferente, pero ambos siguen los mismos patrones de interacción, lo que ha llevado a obtener como resultado el caso docente y el empresarial.

El docente está destinado a ser utilizado en casos académicos de forma que, al permitir la enseñanza de una manera más innovadora y atractiva para los estudiantes, a través de la interacción del usuario con elementos que le proporcionan información como en un juego, busca fomentar su aprendizaje por medio de una estructura que se asemeja a los videojuegos. Siendo posible adaptar esta metodología a cualquier nivel de enseñanza, como por ejemplo a colegios, para motivar a los niños a aprender de una forma más interactiva y llamativa para ellos.

Por otro lado, el caso empresarial permitiría a empresas implementar este trabajo en presentaciones virtuales de sus productos, por medio de un entorno 3D en el cual se expongan versiones virtuales de una gama de productos que quieran presentar al mercado de una manera innovadora y llamativa. Posibilitando la interacción de sus potenciales clientes con estos productos, y obtener así información importante, como el precio, características, etc.

Para llevar a cabo este trabajo se han empleado diferentes herramientas, cuya elección ha requerido una investigación previa sobre las posibilidades que cada uno ofrecía, teniendo en cuenta sus características, cuáles serían los más adecuados para llevar a cabo el proyecto, y la compatibilidad para exportar elementos creados de uno a otro.

Tras una exhaustiva revisión, se llegó a la conclusión de que la herramienta más adecuada para el modelado de edificios y para la creación de habitaciones era Sweet Home 3D. Este programa permite la creación de prototipos de manera eficiente, facilitando así la experimentación con diferentes diseños sin perder tiempo en detalles técnicos complejos, y una vez que el diseño básico está listo se puede exportar en formatos como OBJ o DAE, compatibles con Unity, donde pueden ser añadidos efectos de iluminación, y comportamientos interactivos.

Para elegir el motor de programación se ha seguido el mismo procedimiento de elección que con la herramienta de creación de edificios. Finalmente se llegó a la conclusión de que la mejor opción era Unity, debido a su versatilidad, ya que permite crear tanto juegos como aplicaciones en 3D, y sus herramientas

para manejar gráficos, físicas y animaciones son muy potentes y accesibles. Además, posee una gran comunidad de usuarios y una tienda de recursos (Asset Store) donde se pueden encontrar elementos ya creados, lo que puede acelerar el desarrollo. Puede exportar proyectos a diferentes plataformas como PC, consolas, y dispositivos móviles, y si se desea expandir a realidad virtual o aumentada ofrece soporte completo para estas tecnologías. Es relativamente fácil de aprender comparado con otros motores. Y es adecuado tanto para pequeños proyectos como para desarrollos más grandes y complejos, lo que permite comenzar con algo sencillo y expandirlo según sea necesario.

Esta combinación permite aprovechar las fortalezas de ambas herramientas, la simplicidad y accesibilidad de Sweet Home 3D para el diseño arquitectónico inicial, y la potencia de Unity para la creación de experiencias interactivas y detalladas.

En cuanto a la programación de los scripts, se han utilizado herramientas y lenguajes de programación específicos para la creación de entornos 3D. El lenguaje de programación ha sido C#, y el editor Visual Studio Code, junto con el plugin Bito, que ayuda a autocompletar el código. Además, se ha utilizado ChatGPT como asistente de programación, para resolver los problemas que han ido surgiendo a lo largo de la realización de este trabajo.

Todo lo mencionado en los párrafos anteriores ha supuesto que el desarrollo de este proyecto haya sido una experiencia de gran valor, tanto a nivel personal como profesional, debido a que me ha permitido perfeccionar mis habilidades de investigación y gestión del tiempo. He tenido que identificar y seleccionar las fuentes más relevantes y fiables. Además, el proceso de aprendizaje ha sido muy enriquecedor debido a que he tenido que adquirir nuevos conocimientos y competencias, como el uso de los programas utilizados para el desarrollo del proyecto. Por otro lado, este trabajo no solo me ha proporcionado una comprensión más profunda sobre entornos virtuales 3D e interacción con objetos, sino que también ha fortalecido mi capacidad para resolver problemas y trabajar de manera autónoma.

6. Trabajos futuros

La futura implementación de realidad virtual (VR) [31] en este proyecto se ve altamente favorecida por el uso del Universal Render Pipeline (URP) [32]. Gracias a su excelente compatibilidad con una amplia gama de plataformas, como dispositivos móviles y sistemas de VR, el trabajo desarrollado puede adaptarse fácilmente a dispositivos como Meta Quest, SteamVR, o PlayStation VR, permitiendo una transición fluida entre plataformas sin necesidad de realizar cambios significativos en el código o los gráficos. URP está diseñado para ofrecer un elevado rendimiento, esencial en experiencias de VR, gestionando tasas de refresco altas como los 90 Hz, asegurando una experiencia fluida y cómoda para el usuario. Además, su eficiencia en el uso de recursos es crucial para dispositivos con hardware limitado, como móviles y gafas de VR, garantizando un funcionamiento óptimo en plataformas menos potentes. Por estas razones, URP es una opción ideal para futuros desarrollos de realidad virtual en el proyecto, sin comprometer la calidad visual y facilitando el trabajo a desarrolladores con poca experiencia en gráficos avanzados [33].

Por otro lado, aunque en este trabajo no se ha aplicado inteligencia artificial, en futuros proyectos se podría explorar su implementación utilizando el ML-Agents Toolkit de Unity para crear entornos 3D interactivos más avanzados. Esta herramienta permite entrenar agentes inteligentes, utilizando técnicas de aprendizaje. El proceso de entrenamiento y optimización de estos agentes requiere el uso de Python y bibliotecas específicas de aprendizaje automático. La API de ML-Agents en Python se comunica con Unity, lo que facilita el control del entorno y el entrenamiento de los agentes utilizando algoritmos de machine learning [34]. Por ejemplo, se podrían crear agentes inteligentes que interactúen con los usuarios en tiempo real, proporcionando asistencia personalizada o simulando comportamientos humanos en escenarios de entrenamiento. Esto no solo mejoraría la experiencia del usuario, sino que también abriría nuevas posibilidades en campos como la educación, donde los agentes podrían actuar como tutores virtuales, respondiendo preguntas y guiando a los estudiantes a través de contenidos específicos. La integración de estos agentes inteligentes permitiría crear experiencias más inmersivas y adaptativas, elevando el potencial de los entornos 3D interactivos a nuevos niveles.

7. Referencias Bibliográficas

[1] Unity Technologies, Unity, (2024). https://unity.com/es (accessed May 31, 2024).

[2] eTeks, Sweet Home 3D, (2024). https://www.sweethome3d.com.

[3] Adobe Systems Incorporated, Mixamo, (2024). https://www.mixamo.com.

[4] Microsoft, C# Documentation, (2024). https://learn.microsoft.com/en-us/dotnet/csharp/.

[5] Microsoft, Visual Studio Code, (2024). https://code.visualstudio.com/.

[6] Bito, Bito.ai, (2024). https://bito.ai/.

[7] OpenAI, ChatGPT, (2024). https://chat.openai.com/.

[8] Blender Foundation, Blender, (2024). https://www.blender.org.

[9] Unity Technologies, Unity Documentation, (2024). https://docs.unity.com/.

[10] Trimble Inc., SketchUp, (2024). https://www.sketchup.com.

[11] Autodesk Inc., Autodesk Revit, (2024). https://www.autodesk.com/products/revit.

[12] I. Chief Architect, Chief Architect, (2024). https://www.chiefarchitect.com.

[13] Unity Technologies, Unity Asset Store, (2024). https://assetstore.unity.com/?srsltid=AfmBOopwtPDFyz_E9Vyupp7WH2RsuPA6Pj6wmLp2_j_TwFgqXv7sVtcO.

[14] Godot Engine, Godot Engine, (2024). https://godotengine.org.

[15] Epic Games, Unreal Engine, (2024). https://www.unrealengine.com.

[16] YoYo Games, GameMaker, (2024). https://gamemaker.io.

[17] Cocos, Cocos, (2024). https://www.cocos.com/.

[18] Flax Engine, Flax Engine, (2024). https://flaxengine.com.

[19] GDevelop, GDevelop, (2024). https://gdevelop.io.

[20] Stride 3D, Stride 3D, (2024). https://www.stride3d.net.

[21] Open 3D Engine, Open 3D Engine (O3DE), (2024). https://www.o3de.org.

[22] GB Studio, GB Studio, (2024). https://www.gbstudio.dev.

[23] MindFly - Laberintos, EXPORTAR CASA DE SWEET HOME 3D A UNITY [2021], (2021). https://www.youtube.com/watch?v=B7CgVOJXeww.

[24] Kostas, COLLIDERS y TRIGGERS en UNITY, (2021). https://www.youtube.com/watch?v=R67Xi4CH-tA.

[25] Unity Technologies, Applying Textures to Terrain, (2017). https://docs.unity3d.com/2017.1/Documentation/Manual/terrain-Textures.html.

[26] Alpha Creative, COMO CAMBIAR PRIMERA Y TERCERA PERSONA EN UNITY | Cambio de Perspectiva, (2023). https://www.youtube.com/watch?v=_CnLk3jH4qQ.

[27] Unity Technologies, Starter Assets - Third Person Character Controller, (2024). https://assetstore.unity.com/packages/essentials/starter-assets-thirdperson-updates-in-new-charactercontroller-pa-196526.

[28] Contenido AP, Unity - Basico - Animar, abrir puerta con trigger, (2022). https://www.youtube.com/watch?v=6QP7nc4Ksy4.

[29] Contenido AP, Unity - Cambiar personaje en tercera persona de Starter Assets por otro de Mixamo, (2023). https://www.youtube.com/watch?v=3oKfsoXR9uA&t=300s.

[30] Ready Player Me, Ready Player Me, (2024). https://readyplayer.me/es.

[31] Unity Technologies, Soluciones de VR en Unity, (2024). https://unity.com/es/solutions/vr.

[32] Unity Technologies, Universal Render Pipeline, (2024). https://unity.com/es/srp/universal-render-pipeline.

[33] Unity Technologies, Crear experiencias de realidad virtual y mixta en Unity, (2024). https://unity.com/es/resources/create-virtual-mixed-reality-experiences-unity.

[34] ARM Holdings, ARM Community, (2024). https://community.arm.com.