Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

# Diseño y desarrollo de una aplicación de realidad mixta

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor**: Alfonso Rodríguez Verdera

**Tutor**: Manuel Agustí Melchor

2018-2019

# Agradecimientos

Me gustaría agradecer a mi familia, mi pareja, amigos y tutor.

Gracias a mi familia y amigos por apoyarme y ayudarme durante toda la carrera, vosotros me habéis dado la fuerza que necesitaba.

A mi pareja María, que siempre me ha apoyado y animado en todo este largo proceso de mi vida, si estoy aquí, es por ella.

A mi tutor Manuel Agustí Melchor, por darme la oportunidad de realizar este trabajo y guiarme durante la realización de este proyecto.

# Resumen

La realidad mixta (RM) es la fusión entre la realidad virtual y la realidad aumentada, esta unión permite introducir elementos virtuales en entornos reales, dando una experiencia más real en la interacción entre los objetos digitales y físicos del entorno.

Este proyecto consiste en explorar las opciones y requisitos para realizar aplicaciones dentro del contexto de RM, para ello analizaremos los requisitos hardware y software para desarrollar aplicaciones de RM, también revisaremos los dispositivos de RM existentes hoy en día, así como la instalación y configuración de las gafas de realidad mixta Acer AH101-D8EY proporcionadas por el departamento DISCA de la Universitat Politècnica de València.

Para la realización de este trabajo utilizaremos las herramientas que ofrece Microsoft. Para ello partiremos del entorno que proporciona Microsoft sobre Windows y las librerías de desarrollo disponibles para poder desarrollar una aplicación de RM. En este caso utilizaremos un kit de herramientas para desarrollo de la realidad mixta proporcionado por Microsoft para Unity, el cual nos permite crear escenarios 3D bajo un entorno de realidad mixta.

Finalmente con el fin de comprobar el alcance de las gafas de Acer y explorar las posibilidades que ofrece la RM, desarrollaremos “demos” ambientadas en el universo *Star Wars* donde controlaremos a un Jedi o pilotaremos un caza espacial dónde simularemos la destrucción de la estrella de la muerte.

**Palabras clave:** realidad mixta, realidad virtual, realidad aumentada, Acer AH101-D8EY, Unity, Toolkit mixed reality.

# Abstract

Mixed reality (MR) is the fusion between virtual reality and augmented reality, this union allows virtual elements to be introduced into real environments, giving a more real experience in the interaction between digital and physical objects in the environment.

This project consists of exploring the options and requirements to make applications within the context of MR, for this we will analyze the hardware and software requirements for developing MR applications, we will also review the existing MR devices today, as well as the installation and configuration of Acer AH101-D8EY mixed reality glasses provided by the DISCA department of the Polytechnic University of Valencia.

To carry out this work we will use the tools offered by Microsoft. For this we will start from the environment that Microsoft provides on Windows and the available development libraries to be able to build an MR application. In this case we will use a toolkit for mixed reality development provided by Microsoft for Unity, which allows us to create 3D scenarios under a mixed reality environment.

Finally, in order to check the scope of Acer's glasses and explore the possibilities offered by MR, we will develop “demos” based in the Star Wars universe where we will control a Jedi or pilot a space fighter where collaborating in the destruction of the Death Star.

**Keywords:** mixed reality, virtual reality, augmented reality, Acer AH101-D8EY, Unity, Toolkit mixed reality.

# Resum

La realitat mixta (RM) és la fusió entre la realitat virtual i la realitat augmentada, aquesta unió permet introduir elements virtuals en entorns reals, donant una experiència més real en la interacció entre els objectes digitals i físics de l'entorn.

Aquest projecte consisteix en explorar les opcions i requisits per a realitzar aplicacions dins el context de la RM, per això analitzarem els requisits de programació i maquinari per a desenvolupar aplicacions de RM, també revisarem els dispositius de RM existents hui en dia, així com la instal·lació i configuració de les ulleres de realitat mixta Acer AH101-D8EY proporcionades pel departament DISCA de l'Universitat Politècnica de València.

Per a la realització d'aquest treball utilitzarem les eines que ofereix Microsoft. Per això partirem de l'entorn que proporciona Microsoft sobre Windows i les llibreries de desenvolupament disponibles per poder implementar una aplicació de RM. En concret, utilitzarem un conjunt d'eines per a desenvolupament de la realitat mixta proporcionat per Microsoft per a Unity, el qual ens permet crear escenaris 3D baix un entorn de realitat mixta.

Finalment per tal de comprovar l'abast de les ulleres d'Acer i explorar les possibilitats que ofereix la RM, es construiran "demos" ambientades en l'univers “Star Wars” on controlarem a un Jedi o pilotarem un caça espacial on simularem la destrucció de l'estrela de la mort.

**Paraules clau:** realitat mixta, realitat virtual, realitat augmentada, Acer AH101-D8EY, Unity, Toolkit mixed reality.

# Tabla de contenidos

## Índice

# Índice de diagramas



# Índice de tablas

# Índice de imágenes

# 1. Introducción

Hoy en día vivimos en un mundo donde la tecnología y la información marca nuestro día a día y donde los avances tecnológicos crecen a un ritmo extraordinario en todos los campos. Uno de ellos es la informática y, más concretamente, el desarrollo de aplicaciones de realidad virtual (RV) o de realidad aumentada (RA), donde se pretende llevar al usuario a un mundo alternativo mediante un visor de RV/RA. El desarrollo de esta tecnología de inmersión en una realidad digital ha ido evolucionando con el paso de los años, desde los primeros dispositivos de RV/RA en los años 60 hasta la aparición de la realidad mixta (RM) en los años 90.

## 1.1 Motivación

La realidad mixta, también llamada realidad híbrida, fusiona las realidades aprovechando las ventajas de cada una, con lo que aumentan las posibilidades de uso. En definitiva, la realidad mixta combina elementos digitales sobre un entorno físico, permitiendo al usuario interactuar con ellos.

Actualmente, la realidad mixta es una tecnología que se encuentra en desarrollo, pero es innegable las posibilidades que ofrece al juntar ambas realidades. Por eso empresas como Acer, Samsung o HP han creado sus propios dispositivos facilitando su acceso a cualquier usuario con un ordenador, abriendo así la puerta de esta tecnología al mundo. Otras empresas como Microsoft invierten recursos en su desarrollo incorporando aplicaciones de realidad mixta en sus sistemas operativos como Windows 10.

Tanto la realidad virtual como la realidad aumentada están desarrollándose en muchos ámbitos como el de la medicina, tecnología militar o educación, pero donde más se ven esos avances a nivel de usuario común es en el mundo del entretenimiento. En concreto para los videojuegos, un sector en auge que a través de unas gafas de RV proporciona al usuario una inmersión total dentro de un mundo virtual.

Las perspectivas económicas en cuanto a las tecnologías de RV y RA se sitúan en continuo crecimiento debido a la rápida evolución y desarrollo de esta tecnología que ha permitido, como se ha dicho, una implementación en muchos y diversos sectores para su uso. Algunos de estos motivos son el descenso de los precios de este tipo de tecnología y las grandes posibilidades que ofrece al usuario. En la Figura 1 podemos ver un pronóstico de cómo el mercado de la RV y la RA está en continuo crecimiento. Según un estudio del portal de estadísticas *Statista.com* se prevé un gasto de 20.400.000 millones de dólares para el año 2019.

**Estimación del mercado de la realidad virtual y aumentada**

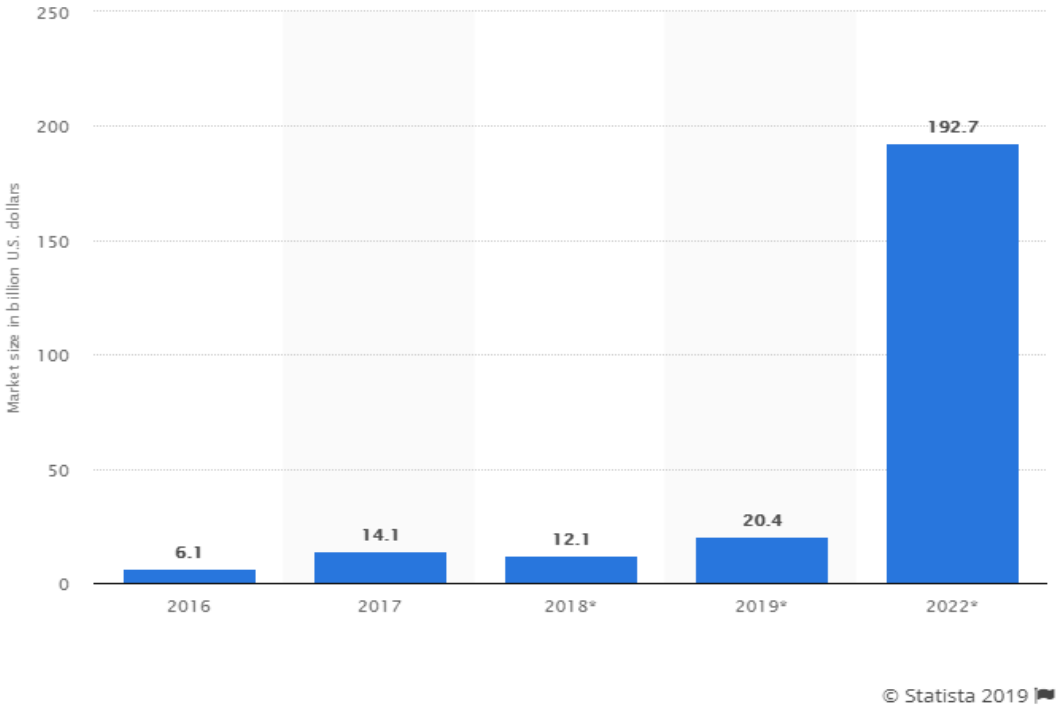



***Figura 1*** *Pronóstico en ventas RV y RA en billones de dolares*[1].

Este auge por la RV/RA no solo está el mundo del entretenimiento, está tecnología puede ayudar a simular situaciones reales que serían demasiado caras de recrear, formando así a nuevos trabajadores de forma más eficiente y económica. Es por esto que cada vez más empresas invierten en estos campos con el fin de mejorar su productividad [1] como es el caso de BMW que ha incorporado la RV en el diseño de sus automóviles o Esteé Lauder creando un espejo virtual que utiliza la realidad aumentada para que los usuarios puedan probar sus maquillajes. También Google está probando nuevas ideas de uso para la RV/RA en diferentes ámbitos como la educación, arquitectura, ingeniería o publicidad.

Teniendo en cuenta las posibilidades que ofrece la realidad mixta, a lo largo de este proyecto se trabajarán los diferentes usos y posibilidades de la misma, los dispositivos que se encuentran en el mercado y se realizarán unos ejemplos prácticos en RM con el fin de proporcionar una guía de las etapas en que se puede dividir el ciclo de desarrollo de una aplicación de RM.

[1] https://www.statista.com/statistics/591181/global-augmented-virtual-reality-market-size/

## 1.2 Objetivos

En el presente proyecto identificamos dos objetivos principales. El primero de ellos, consiste en abordar las características que diferencian el desarrollo de una aplicación "convencional" de una de RM y revisar los componentes software y hardware para el desarrollo de aplicaciones de RM.

En segundo lugar, con el fin de demostrar las posibilidades de uso de las gafas de Acer desarrollaremos una aplicación de realidad mixta mediante el motor gráfico de Unity para un entorno Windows 10. El desarrollo de esta aplicación constará de dos partes: en primer lugar, instalar y configurar el visor de RM de Acer con la plataforma de desarrollo y en segundo lugar, realizar un juego que constará de tres prototipos dónde podamos probar los principales casos de uso de una aplicación de RM.

A continuación se presentan los objetivos secundarios que se plantean:

- Definir la tecnología de Realidad Mixta.
- Investigar sobre las diferentes aplicaciones de la RM.
- Conocer las diferentes plataformas para desarrollar software de RM.
- Realizar una guía para el desarrollo de aplicaciones de RM.
- Desarrollar tres prototipos de RM para Windows 10 utilizando el visor de Acer AH101-D8EY.

### 1.3 Estructura de la memoria

Esta memoria sigue la siguiente estructura por capítulos:

- Capítulo 2 Estado del arte: Se introducen y detallan las herramientas software en el desarrollo de la RM. Además se analizaran y compararán los dispositivos de realidad mixta que hay en el mercado actual.

- Capítulo 3 Instalación y pruebas: Procederemos a la instalación del visor de RM Acer AH101-D8EY, comprobaremos su funcionamiento y analizaremos las funciones que nos ofrece. Además configuraremos el entorno de desarrollo de Unity instalando el kit de desarrollo de realidad mixta (MRTK).

- Capítulo 4 Diseño del proyecto: En este apartado se describen los pasos realizados para diseñar una aplicación de realidad mixta con *Unity*. Veremos los diagramas y casos de uso de nuestra aplicación.

- Capítulo 5 Desarrollo del proyecto: En este capítulo empezaremos a construir la aplicación con el motor *Unity*. Se verán los componentes y *scripts* utilizados en el proyecto.

- Capítulo 6 Conclusiones y trabajos futuros: Comprobaremos si se han cumplido los objetivos y se profundizará sobre los resultados obtenidos y las líneas de trabajos futuros.

# 2. Estado del arte

En este capítulo se profundizará sobre la RM como una evolución de la RV y la RA, ampliando así las posibilidades de esta tecnología. Presentaremos la definición y evolución de la realidad virtual y la realidad aumentada, así como el estado actual en que se encuentran y la llegada de la realidad mixta como una combinación de ambas de cara al futuro.

Antes de definir los diferentes tipos de realidades, tenemos que definir qué es el mundo real y el mundo virtual. El mundo real es aquel que observamos y podemos percibir con nuestros sentidos sin ayuda de ningún dispositivo tecnológico. La RAE la define como: “Lo que es efectivo o tiene valor práctico, en contraposición con lo fantástico e ilusorio.” Por otra parte el mundo virtual es aquel que ayudado por un dispositivo u ordenador altera la realidad generando estímulos para los diferentes sentidos, principalmente visualizando imágenes.

## 2.1 Terminología

El límite entre las tres tipos de realidades mencionadas es difuso, por ello vamos a definirlas, con el fin de entender mejor las diferencias entre ellas. Véase la figura 2.


***Figura 2*** *Realidad virtual, realidad aumentada, realidad mixta* [2].

La Realidad Virtual podemos definirla como la simulación de una realidad alternativa a la real. A. Rowell definió la RV como [2] “una simulación interactiva por computador desde el punto de vista del participante, en la cual se sustituye o se incrementa la información sensorial que recibe”.

[2] http://digitalmedia.com.ec/web/2017/12/14/realidad-aumentada-vs-realidad-virtual-vs-realidad-mixta-una-guia-introductoria/

La realidad aumentada se puede definir como [3] “una combinación de elementos reales y virtuales en 3D interactiva en tiempo real”. Dicho de otra forma la realidad aumentada incorpora información digital sobre un entorno real.

La realidad mixta fue definida por Paul Milgram y Fumio Kishino en 1994 como [4] “cualquier espacio entre los extremos del continuo de la virtualidad, se extiende desde el mundo completamente real hasta el entorno completamente virtual, donde los objetos virtuales interactúan con el mundo físico”.

## 2.2 Historia de la realidad virtual

La RV se remonta a 1950. Su origen surge en la segunda guerra mundial, cuando Estados Unidos buscaba la forma de mejorar y agilizar el entrenamiento de los pilotos a través de simulaciones de vuelo. Con el paso de los años, la tecnología se fue desarrollando hasta que en 1962, Mortin Heilig inventó el *Sensorama* [5] , ver figura 3a, una maquina capaz de mostrar imágenes estereoscópicas tridimensionales en gran angular con sonido y aromas con el objetivo de sumergir al usuario en un mundo alternativo lo más real posible.

Años después, en 1968, Ivan Sutherland fabricó “la espada de Damocles” [5], figura 3b, el primer casco de realidad virtual. El cual consistía en un brazo mecánico anclado en el techo que sostenía un sistema de visualización compuesto por dos pequeñas pantallas estereoscópicas, estas a través de un ordenador generaba imágenes de objetos en tres dimensiones. A su vez, los movimientos del usuario eran detectados por los sensores que permitían al ordenador modificar la orientación de los objetos respecto al usuario.

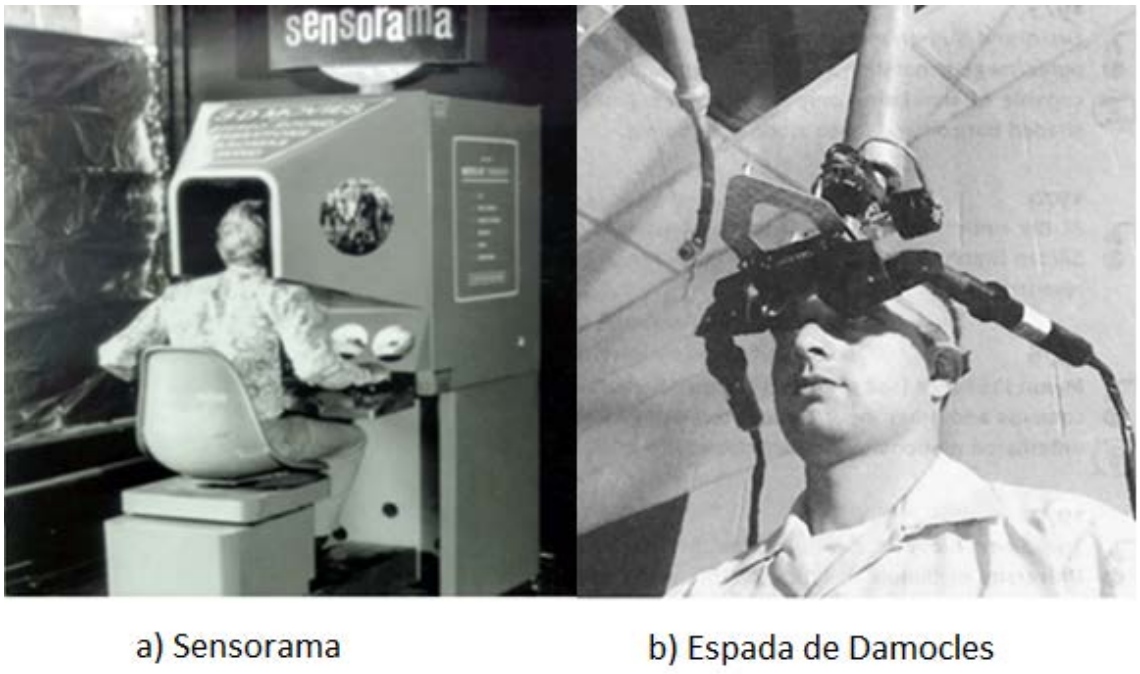



***Figura 3**. Ejemplos de RV. Fuente Wikipedia.org.*

Entre 1970 y 1990 se llevaron a cabo varios proyectos como por ejemplo el proyecto Aspen Movie Map [6], creado en el MIT (Instituto Tecnológico de Massachusetts). Este proyecto consistía en una simulación virtual de la ciudad de Aspen, donde los usuarios podían recorrer las calles de la ciudad simulando un entorno 3D.

Entre los años 1990 y 2000 las empresas de videojuegos, *Nintendo* y *Sega* lanzaron sus propios cascos de realidad virtual [7] para sus respectivas consolas, véase la figura 4. Estas dos empresas, fueron las pioneras en abrir el mundo de la realidad virtual al campo del entretenimiento. De este modo, se dio un salto en el uso de esta tecnología, pasando del uso militar al uso civil.

***Figura 4*** *Primer dispositivo para videojuegos de realidad virtual. Extraída de: mediatrends.es*

Finalmente, desde el año 2000, empresas tecnológicas como Google, HP, Samsung o Microsoft empezaron a desarrollar productos relacionados con la realidad virtual, cada uno con sus propios dispositivos. Un ejemplo de ello es el casco de realidad virtual *Oculus Rift* desarrollado en el año 2016 y que se comercializa sobre diferentes plataformas como Windows, Linux o mac OS.

Hoy en día el uso de la realidad virtual está muy extendido en diferentes campos, entre ellos la industria de los videojuegos, aplicaciones militares, educación, entretenimiento e incluso en medicina. En este último ámbito se puede destacar el desarrollo del proyecto T-Room[3], desarrollado entre los investigadores de los Centros de Desarrollo Cognitivo Red Cenit y de la Universidad Politécnica de Valencia a través del cual pretenden emplear entornos virtuales inmersivos como herramienta de detección de Trastornos del Espectro Autista en niños.

[3] http://www.i3b.upv.es/het/es/proyecto-t-room/

### 2.3 Historia de la realidad aumentada

La realidad aumentada ha ido desarrollándose al mismo tiempo que la realidad virtual. En el año 1990 [8], el investigador de Boeing, Tom Caudell, intentando mejorar los procesos de fabricación de los tableros de configuración de cables, llevó a cabo unas gafas (figura 5) que proyectaban tableros virtuales sobre los tableros reales, ayudando al usuario a realizar su trabajo de manera más eficiente. Es en esta época cuando la realidad aumentada empieza a diferenciarse de la realidad virtual y surge su primera definición: "la inclusión, en tiempo real, de elementos virtuales dentro del universo físico". Tom Caudell.

***Figura 5*** *Tablero de configuración cables. Fuente: informit.com.*

A pesar de que T. Caudell fue el primero en definir la realidad aumentada, fue L.B. Rosenberg, en los años 80, quien fabricó el primer sistema basado en esta realidad. Como suele ser habitual en este tipo de aplicaciones, fue en el ámbito militar. Se trataba de un dispositivo que asistía al usuario sobre cómo realizar ciertas tareas, como un guía virtual. Más tarde, en 1994, los investigadores de la Universidad de Columbia Steven Freiner, Blair MacIntyre y Dorée Seligmann inventaron un dispositivo llamado KARMA, capaz de proyectar imágenes en 3D para dar instrucciones al usuario de como recargar una impresora.

Durante los siguientes años la realidad aumentada comenzó a coger impulso y se empezaron a desarrollar nuevos sistemas gracias a la mejora de las capacidades computacionales de los ordenadores y tarjetas gráficas. Pero no es hasta 2012, cuando *Google* lanzó al mercado las *Google Glass* [9] figura 6, unas gafas que añaden información adicional sobre la realidad que se está observando. A partir de entonces, empresas tecnológicas empezaron a desarrollar sus propios productos de realidad aumentada.

***Figura 6*** *Gafas de RA Google Glass.*

Otro logro de esta tecnología fue en el año 2016. *Niantic* lanza al mercado Pokemon Go, un juego de realidad aumentada para dispositivos móviles, en el que a través de la cámara se mezclan elementos de un entorno real con otros virtuales. El juego tuvo una gran acogida y repercusión en la población reimpulsando de este modo la realidad aumentada como nunca antes se había visto.

Actualmente, se está realizando un gran esfuerzo en el desarrollo de plataformas de realidad aumentada para potenciar la inclusión en distintos ámbitos. Uno de ellos es el de la educación. Un ejemplo de ello es el proyecto Onirix[4], una plataforma donde se pueden desarrollar aplicaciones de RA con modelos 3D. Por medio de estas aplicaciones se ayuda a los profesores a explicar conceptos abstractos a los alumnos de manera interactiva a través de un dispositivo móvil.

## 2.4 Realidad mixta

La realidad mixta, también llamada realidad hibrida es el resultado de combinar la realidad virtual con la realidad aumentada. Esta fusión de tecnologías permite al usuario una inmersión de la realidad mayor, mejorando la experiencia y aumentando, de ese modo, sus posibilidades de uso ya que permite abarcar un espectro más amplio.

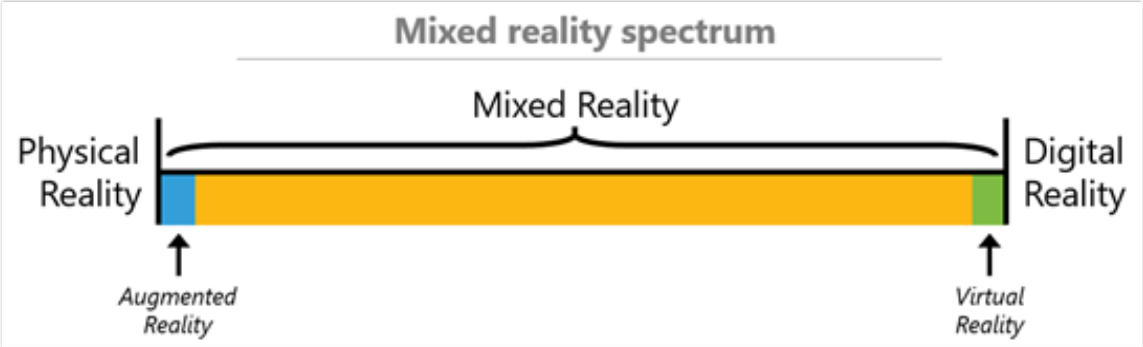



***Figura 7**: Espectro de la realidad mixta. Fuente: docs.microsoft.com.*

Como podemos observar en la figura 7, el espectro de realidad mixta [10] abarca tanto a la realidad virtual como a la realidad aumentada. A medida que nos acercamos hacia la RV accedemos a un mundo virtual, donde encontraremos más cantidad de elementos generados por ordenador (elementos virtuales), mientras que si nos desplazamos hacia la RA, nos aproximaremos más a la realidad física.

La realidad mixta incluye ambas tecnologías por lo que engloba mayores prestaciones. Esta realidad permite una experiencia más envolvente con una interacción persona-computador que mejora la percepción de una realidad. Esta mejora de la percepción es debido a la cantidad de sensores que se han ido incorporando en los dispositivos de realidad mixta y la mejora del hardware que permite un mayor rendimiento y calidad visual.

El sistema de funcionamiento de la RM consiste principalmente en introducir objetos virtuales sobre el mundo real o sobre un entorno virtual de forma instantánea. Estos objetos pueden ser reales, los cuales se escanean en imágenes 3D para posteriormente introducirlas en el mundo virtual. Por otra parte si estos objetos son virtuales, el sistema se encargará de leer estos objetos e incorporarlos en el mundo real situándolos en una escena según la posición establecida.

[4] https://www.onirix.com/es/

Para conseguir una experiencia de realidad mixta se necesitan dispositivos específicos que incorporen una serie de sensores (cámaras, giroscopios, sensores de proximidad…) con los que se pueda capturar la información del mundo real e introducirlos en el mundo virtual. Es en este punto donde la tecnología está desarrollándose con el fin de conseguir dispositivos más inmersivos que abarquen el máximo espectro de la realidad.

### 2.5 Diferencia entre realidad virtual, realidad aumentada y realidad mixta

Tanto en la realidad virtual como en la realidad aumentada se utilizan dispositivos similares, lo cual puede llegar a complicar la diferenciación entre ellas. Sin embargo, sus utilidades y funcionamiento son diferentes. La realidad virtual es completamente inmersiva, es decir, el usuario interactúa con un mundo completamente virtual, mientras que en la realidad aumentada se aplican elementos virtuales sobre un mundo físico real.

En cuanto a la recepción de información percibida por el usuario en la realidad virtual nos encontramos con una distorsión de los sentidos mayor que en la realidad aumentada, debido a que nos sumergimos en un entorno totalmente virtual lo cual puede producir una sensación mayor de mareos. Además de esto, los dispositivos necesarios para estas realidades son distintos según el tipo de realidad que experimentemos, en los dispositivos de realidad aumentada los usuarios no necesitan colocarse ningún dispositivo en la cabeza, basta con una pantalla que proyecte dicha información. No es así en la realidad virtual, que necesita de un dispositivo en la cabeza para trasmitir esta sensación de inmersión hacia un mundo virtual.

Otra diferencia entre ambas tecnologías es su nivel de requisitos hardware a nivel computacional. En este aspecto, la realidad virtual tiene unos requisitos más altos, ya que debe procesar muchos más datos en tiempo real que la realidad aumentada.

Finalmente vamos a resumir en la Tabla 1 las principales características de estas tres realidades.

| *REALIDAD VIRTUAL* | *REALIDAD AUMENTADA* | *REALIDAD MIXTA* |
|---|---|---|
| **Inmersión completa:** escenario, objetos, gráficos virtuales. | **Inmersión mixta:** Añade elementos virtuales sobre un entorno físico real. | **Inmersión mixta:** combina elementos reales y virtuales. |
| *Necesita de un headset en la cabeza del usuario.* | Uso de pantallas o headsets. | Necesita de un headset en la cabeza del usuario. |
| *Requerimientos hardware altos.* | Requerimientos hardware medios. | Requerimientos hardware altos. |

***Tabla 1*** *Características de las diferentes realidades.*[5]

[5] https://www.aniwaa.com/guide/vr-ar/ultimate-vr-ar-mr-guide/

## 2.6 Libreras para el desarrollo de la realidad mixta

- **MRTK**

Se trata de un kit de herramientas de código abierto [11] con una comunidad de desarrolladores apoyada por Microsoft, permite el desarrollo de aplicaciones de realidad mixta en Unity para Windows 10. Este kit proporciona una serie de prototipos y modelos prefabricados listos para usarse ayudando a los desarrolladores a crear contenido de forma más rápida. Además es compatible con la mayoría de dispositivos de realidad mixta como son: Acer Windows *Mixed Reality*, HP Windows *Mixed Reality*, Lenovo Explorer, Asus Windows *Mixed Reality* o HoloLens.

- **OpenHMD**

OpenHMD [12] proporciona una API[6] gratuita, de código abierto que permite desarrollar software multiplataforma (Android, OpenBSD, Linux, OS X, Windows) para dispositivos de realidad mixta. OpenHMD ofrece un plugin para el motor de desarrollo Unity que permite controlar el seguimiento de cámara y su rotación, por contra no admite el seguimiento de posición del usuario por lo que se limita su funcionamiento al seguimiento de la cabeza. Además tiene una gran compatibilidad con los dispositivos de RM como: Oculus Rift, HTC Vive, Sony PSVR, Deepoon E2.

- **Build Wagon**

Build Wagon [13] es un entorno de desarrollo basado en web para uso exclusivo de las Hololens, permite escribir código en *Javascript* guardado en la nube para múltiples desarrollos. Además tiene un emulador[7] online que permite simular tus proyectos. Build Wagon tiene dos versiones, una gratis en la que se limita el desarrollo a un proyecto y a 1 GB de espacio y la versión plus con proyectos ilimitados y 2GB de espacio en la nube por 35 dólares al mes.

- **MixCast**

MixCast [14] es un SDK diseñado para desarrollar aplicaciones y videos de realidad mixta desde una realidad virtual. Tiene compatibilidad con los motores de desarrollo principales como son Unity y Unreal permitiendo un desarrollo más rápido. El software de MixCast crea una salida de vídeo del mundo real en el cual agrega elementos virtuales. Está disponible en dos versiones, una de prueba limitada y una de pago de 50 dólares al año.

[6] http://openhmd.net/doxygen/0.1.0/openhmd_8h.html
[7] https://buildwagon.com/holoemulator.html

## 2.7 Entornos de desarrollo de realidad mixta

En este apartado vamos a ver los principales entornos de desarrollo compatibles para desarrollar aplicaciones de realidad mixta.

### 2.7.1 Unity

Unity es un motor gráfico para el desarrollo de videojuegos que permite trabajar tanto en 2D como en 3D [15] , siendo de las plataformas más usadas actualmente. Creado en el 2005 por Unity Technologies, permite desarrollar para múltiples plataformas como son Windows, Android, Linux o IOS.

Además ofrece distintos modelos de versión, para un uso gratuito y un modo más completo para empresas. Unity dispone de un editor visual donde poder controlar y manipular los objetos, así como la posibilidad de importar modelos y texturas.

Este motor gráfico, utiliza *PhysX* para los cálculos físicos, el cual está desarrollado por Nvidia[8] por lo que ofrece una gran compatibilidad con las tarjetas gráficas existentes, Además utiliza OpenGL[9] y ofrece soporte tanto a la realidad virtual como a la realidad aumentada.

Dispone de una tienda para descargar recursos, plantillas, modelos, etc, creados por la comunidad para ayudar en el desarrollo de sus aplicaciones. También tiene una gran cantidad de documentación para ayudar a futuros desarrolladores.

Unity se integra con Visual Studio facilitando así la creación de scripts con el lenguaje de programación C-Sharp (C#), que es un lenguaje orientado a objetos y desarrollado por Microsoft.

### 2.7.2 Unreal engine

Creado en 1998, Unreal Engine es un motor gráfico para videojuegos creado por Epic Games [16]. Este motor escrito en C++ es compatible con plataformas tales como: Microsoft Windows, macOS, Linux, SteamOS, iOS, Android, PlayStation 4, PlayStation VR. Por su parte también ofrece varias herramientas para la construcción de entornos desde su tienda.

Unreal Engine se encarga de renderizar los gráficos en 2D y 3D así como del sonido, animación, memoria e inteligencia artificial, proporcionando una calidad gráfica muy alta. Además permite desarrollar juegos en realidad virtual con una inmersión muy detallada. Desde sus primeras versiones tiene compatibilidad para OpenGL y Direct3D.

[8] https://www.nvidia.com/es-es/
[9] https://es.wikipedia.org/wiki/OpenGL

En cuanto a la licencia de uso. Desde 2015 se lanzó la versión de Unreal Engine 4 disponible de forma gratuita con el único requisito de pagar un 5% de los primeros 3000 dólares ganados con esta herramienta.

### 2.7.3 CryEngine

Cry Engine es un motor gráfico [17] creado por la empresa Crytek en 2002 para las tarjetas gráficas de Nvidia. Debido a su éxito se desarrolló como motor gráfico para videojuegos con alta calidad en tiempo real. Se trata de un sistema multiplataforma compatible con la mayoría de plataformas de videojuegos (PlayStation 3, Xbox 360 y Wii U). Tiene soporte tanto para Windows como para Linux. También a través de las últimas actualizaciones soporta de forma nativa DirectX 12, Vulkan y RV.

En cuanto a su desarrollo, usa *scripts* escritos en C# y C++. Utiliza inteligencia artificial para mejorar la calidad gráfica y utiliza audio 3D. Al igual que los otros motores gráficos, *CryEngine* dispone de su propia tienda donde los usuarios pueden descargar materiales, sonidos y objetos 3D creados por la comunidad. Esta comunidad creada desde 2016 proporciona documentación y formación a los desarrolladores.

Otro aspecto importante para todo desarrollador es su licencia de uso, la cual a partir de la versión *CryEngine* 5, fue liberada al público con un sistema "paga lo que quieras" por lo que puedes usarlo pagando lo que consideres oportuno.

| ***Motor de desarrollo*** | ***Plataformas compatibles*** | ***Realidad mixta*** | ***Desarrollo en 3D*** | ***Lenguajes de programación*** | ***Precio*** |
|---|---|---|---|---|---|
| *Unity* | Windows, OS X y Linux, PLAYSTATION, XBOX, IOS, ANDROID. | SI | SI | UnityScript , C# y Boo | Versión gratuita<br><br>Versión de pago |
| *CryEngine* | Windows, OS X, Linux, PLAYSTATION, XBOX, IOS, ANDROID. | SI | SI | C++, Lua, C#. | Versión gratuita |
| *Unreal Engine* | Windows, OS X y Linux, PLAYSTATION, XBOX, IOS, ANDROID. | SI | SI | C++ | Versión gratuita |

***Tabla 2*** *Resumen motores gráficos.*

Como conclusión de este apartado, la tabla 2 muestra un resumen de los diferentes entornos de desarrollo para la realidad mixta.

Para el desarrollo de este proyecto se ha elegido la plataforma de desarrollo Unity ya que se trata de un software que ofrece una versión gratuita para el desarrollo de aplicaciones, además de disponer de una amplia documentación libre [18] y recursos ya construidos listos para usar por parte de su comunidad de desarrolladores. Otro aspecto importante es la versatilidad que permiten los *scripts* en C#, el cual es clave ya que es un lenguaje que se ha estudiado en algunas asignaturas de la carrera. Finalmente otro punto clave para desarrollar a través de Unity es su integración con el MRTK de Microsoft, esto permite desarrollar proyectos de realidad mixta de forma más rápida y eficaz.

## 2.9 Dispositivos de realidad mixta

En el mundo de la realidad mixta existen diferentes dispositivos con los que interactuar con el mundo virtual. Estos dispositivos permiten al usuario extender la percepción de la realidad entre el espacio real y virtual, aumentando las capacidades de la realidad virtual y aumentada.

Dentro de la realidad mixta podemos distinguir dos tipos de dispositivos [19]. Véase la tabla 3:

- Los **dispositivos inmersivos** o envolventes capaces de crear entornos virtuales de 360 grados, se caracterizan por tener una pantalla totalmente opaca y utilizar sensores como cámaras y giroscopios para detectar la posición del usuario y adaptarla al entorno virtual.

- Los **dispositivos holográficos,** los cuales permiten ver el mundo real junto con contenido virtual. Este contenido virtual se representa por medio de hologramas creados de manera digital, el usuario podrá interactuar con ellos mediante gestos o comandos de voz recogidos por los sensores del dispositivo.

| ***Dispositivo*** | ***Pantalla*** | ***Cámaras*** | ***Entorno*** | ***Mapeo espacial*** | ***Espacio de seguimiento*** |
|---|---|---|---|---|---|
| *Holográfico* | Pantalla transparente. | Ver el entorno real. | Hologramas virtuales sobre un entorno real. | Permite un mapeo espacial. | Espacio de uso no limitado físicamente. |
| *Inmersivo* | Pantalla opaca. | Detectar posición del usuario. | Inmersión total en un mundo virtual. | No permiten mapeo espacial. | Espacio de uso limitado a través de cables. |

***Tabla 3*** *Diferencias dispositivos holograficos vs inmersivo.*

Debido a la gran versatilidad de estos dispositivos empresas como Microsoft, Samsung, HP o Acer han presentado sus dispositivos cada uno con unas especificaciones y precios diferentes con el fin de llegar a todos los públicos. A continuación vamos a describir algunos de los dispositivos más representativos de realidad mixta actuales.

### 2.9.1 Dispositivos inmersivos

#### HP Windows Mixed Reality[10]

***Figura 8*** *HP Windows Mixed Reality.*

Especificaciones del dispositivo:

- Dos pantallas de cristal líquido de alta resolución a 1440 x 1440.
- Tamaño de pantalla diagonal LCD de 2,89 "(x2).
- Pantalla con bisagras en la parte delantera con dos cámaras frontales.
- Hasta 105 grados de campo de visión horizontal.
- Frecuencia de actualización de la pantalla hasta 90 Hz.
- Salida de audio incorporada y soporte de micrófono a través de jack de 3.5mm.
- Cable individual de hasta 4m con salida HDMI 2.0 (pantalla) y USB 3.0 (datos) para conectividad.
- Precio: 403 €.

[10] https://store.hp.com/SpainStore/Merch/Offer.aspx?p=c-auriculares-realidad-mixta

**Samsung hmd odyssey+**[11]


***Figura 9*** *Samsung hmd odyssey+.*

Especificaciones del dispositivo:

- Dos pantallas Amoled de resolución 2880 x 1600 (1440 x 1600 por ojo).
- Tamaño de pantalla diagonal de 3,5"(x2).
- Pantalla con bisagras en la parte delantera.
- Hasta 110 grados de campo de visión horizontal.
- Frecuencia de actualización de la pantalla de 60 a 90 Hz.
- Salida de audio incorporada y doble micrófono.
- Sensores: 6 cámaras DOF, IPD sensor, G-sensor, giroscopio, sensor de proximidad.
- Cable individual 4 metros con HDMI 2.0 (pantalla) y USB 3.0 (datos) para conectividad.
- Peso: 644g.
- Precio: 446 €

[11] https://www.samsung.com/us/computing/hmd/windows-mixed-reality/hmd-odyssey-windows-mixed-reality-headset-xe800zba-hc1us/

**ASUS Windows Mixed Reality Headset**[12]

***Figura 10*** *Gafas Asus.*

Especificaciones del dispositivo:

- Dos pantallas de cristal líquido de alta resolución a 2880 x 1440.
- Tamaño de pantalla diagonal de 2,89 "(x2).
- Pantalla con bisagras en la parte delantera.
- Sensores de seguimiento: acelerómetro, giroscopio, sensor de proximidad y magnetómetro.
- Hasta 105 grados de campo de visión horizontal.
- Frecuencia de actualización de la pantalla hasta 90 Hz.
- Salida de audio incorporada y soporte de micrófono a través de jack de 3.5mm.
- Cable de 4 metros con salida HDMI 2.0 (pantalla) y USB 3.0 (datos) para conectividad.
- Peso: 400g.
- Precio: 357 €[13].

[12] https://www.asus.com/us/Headset/ASUS-Windows-Mixed-Reality-Headset-HC102/specifications/

[13] https://www.microsoft.com/en-us/p/asus-windows-mixed-reality-headset-with-motion-controllers/8ngvlhl26hzg?activetab=pivot:overviewtab

**Acer mixed reality**[14]


***Figura 11*** *Gafas Acer.*

Especificaciones del dispositivo:

- Tamaño de pantalla: LCD 2.89" x 2.
- Densidad de pixel: 706 ppi.
- Resolución: 1440 x 1440 para cada ojo.
- Refresco: 90Hz usando HDMI 2.0, 60Hz usando HDMI 1.4.
- Sensores de seguimiento: acelerómetro, giroscopio, sensor de proximidad y magnetómetro.
- Cámara de seguimiento: Inside Out B+W VGA Camera.
- Conectividad: 1 cable 4 metros HDMI 1.4/2.0, 1 x USB 3.0, bluetooth 4.0.
- Campo de visión: 100 grados.
- Peso: 440g.
- Precio: 360 €.

[14] https://www.coolmod.com/acer-windows-mixed-reality-headset-gafas-realidad-virtual-precio

### 2.9.2 Dispositivos holográficos

**Microsoft HoloLens**[15]

***Figura 12*** *Gafas Microsoft HoloLens.*

Especificaciones del dispositivo:

- Lentes transparentes holográficos.
- Resolución holográfica: 2,3 M puntos de luz.
- Interfaz de gestos, voz y mirada.
- Procesador Intel Atom x5-Z8100 (64 bit) 4 núcleos de 1GHz.
- Sensores: acelerómetro, giroscopio, magnetómetro, 4 Cámaras de seguimiento, 1 cámara de profundidad y 1 cámara de foto 2MP y vídeo HD a 30 FPS, 4 micrófonos y altavoces integrados.
- Conectividad: Wi-Fi 802.11ac, Micro USB 2.0, Bluetooth 4.1 LE.
- Batería de 16.500 mWh hasta 3h de uso activo.
- Memoria Flash de 64 GB y 2 GB de RAM.
- Campo de visión: hasta 120 grados.
- Precio: 5.489 €[16].

[15] https://docs.microsoft.com/es-es/windows/mixed-reality/hololens-hardware-details
[16] https://www.microsoft.com/es-es/p/microsoft-hololens-commercial-suite/944xgcf64z5b?cid=msft_web_collection&activetab=pivot:techspecstab

**Microsoft Hololens 2**[17]

***Figura 13*** *Microsoft Hololens 2. Fuente: wearable-technologies.com.*

Especificaciones del dispositivo:

- Lentes transparentes holográficos.
- Resolución holográfica: 2,5 M puntos de luz.
- Interfaz de gestos, voz y mirada.
- Procesador Qualcomm Snapdragon 850.
- Sensores: acelerómetro, giroscopio, magnetómetro, 4 Cámaras de seguimiento, 1 cámara de profundidad y 1 cámara de foto 8MP y vídeo1080p a 30 FPS, 5 micrófonos y altavoces integrados.
- Conectividad: Wi-Fi 802.11ac, USB tipo C, Bluetooth 5.0.
- Seguimiento 6DoF, mapeo espacial y captura de hologramas.
- Memoria Flash de 64 GB y 4 GB de RAM.
- Campo de visión: hasta 120 grados.
- Peso 566g.
- Precio: 3.500 $ para empresas.

[17] https://www.microsoft.com/es-es/hololens/hardware

Después de analizar los dispositivos de realidad mixta del mercado, podemos extraer las principales características de cada uno para ver cuál de ellos tiene una mejor relación calidad precio. En la tabla 4 realizaremos una comparativa contemplando las especificaciones de cada dispositivo [20].

| Headset | Resolución | Campo visión | Sensores | Pantalla | Precio |
|---|---|---|---|---|---|
| *HP Windows Mixed Reality* | 1440 x 1440 | Hasta 105 grados | acelerómetro, giroscopio, proximidad, magnetómetro | LCD 2,89" (x2) | 403€[18] |
| *Samsung hmd odyssey+* | 1440 x 1600 | Hasta 110 grados | MR Camera, IPD sensor, G-sensor, giroscopio, proximidad | AMOLED 3,5" (x2) | 446€ |
| *ASUS Windows Mixed Reality* | 1440 x 1440 | Hasta 105 grados | Giroscopio, acelerómetro proximidad, magnetómetro | LCD 2,89" (x2) | 357€ |
| *Acer mixted reality* | 1440 x 1440 | Hasta 100 grados | acelerómetro, giroscopio, proximidad, magnetómetro | LCD 2,89" (x2) | 360€ |

***Tabla 4*** *Comparativa dispositivos de realidad mixta.*

En la Tabla 4 podemos ver que el dispositivo más económico son las gafas de "ASUS Windows Mixed Reality", las cuales disponen de unas características similares al resto. Por contra, tenemos las gafas de "Samsung hmd odyssey+", que destacan por su pantalla Amoled de 3,5 pulgadas en cada lente y su resolución superior de 1440x1600.

Por otra parte las gafas de *Microsoft HoloLens* se distancian del resto por sus especificaciones, apostando por un contenido holográfico donde los usuarios pueden interactuar con los elementos mediante gestos.

Finalmente, nosotros utilizaremos las gafas proporcionadas para este proyecto, que son las "Acer mixed reality" y analizaremos que resultados ofrecen.

[18] https://www.pccomponentes.com/gafas-realidad-virtual-hp-vr1000-100nn-windows-mixed-reality-headset

# 3. Instalación y pruebas

Para la realización de este proyecto se dispone de las gafas de realidad mixta Acer AH101-D8EY proporcionadas por el Departamento de Informática de Sistemas y Computadores (DISCA) de la Universitat Politècnica de València. En este aparatado procederemos a su instalación y ver los posibles problemas que puedan surgir en su instalación.

## 3.1 Características de las gafas de realidad mixta Acer AH101-D8EY

En este apartado vamos a analizar más detalladamente los componentes y características de estas gafas de Acer. Las de Acer fueron las primeras gafas presentadas para la Windows *Mixed Reality* en marzo del 2017. Su precio es de 360 euros en Amazon.

### 3.1.1 *Headset*


***Figura 14**: Gafas Acer AH101-D8EY.*

El visor de Acer, Figura 14, tiene unas dimensiones físicas de 195,8 x 94,8 x 106,59 mm, con un peso de 350 g. En la parte frontal cuenta con dos cámaras para realizar el seguimiento y posicionamiento del usuario. Esto es ayudado por los leds que integran los controladores de movimiento. En cuanto a los sensores que llevan las gafas Acer, podemos encontrar un giroscopio, sensor de proximidad, acelerómetro y magnetómetro. El cable de datos y vídeo es de 4 metros y se conecta a través de un conector HDMI y un USB 3.0, estos a su vez vienen acompañadas de un conector Jack 3.5 para la salida de sonido.

***Figura 15*** *Lentes LCD Acer.*

Las gafas Acer disponen de dos pantallas LCD de 2.89 pulgadas y una resolución de 1440x1440 pixeles cada una. Véase la figura 15. Estas lentes ofrecen una resolución combinada de 2880x1440, lo cual es más que suficiente para una visualización óptima de los contenidos. Además su frecuencia de refresco es de 90 Hz. Otro aspecto importante a destacar es su ángulo de visión de 100º, que permite una inmersión total dentro de las aplicaciones de realidad virtual.

### 3.1.2 *Controladores de movimiento*


***Figura 16*** *Controladores de movimiento. Fuente [21].*

Los controladores de movimiento de Acer, Figura 16, funcionan a través de bluetooth 4.0 y requieren de dos pilas AA para su funcionamiento. Estos sirven para marcan el posicionamiento a través de los leds luminosos que rodean a cada controlador, estos leds, son detectados por las cámaras del visor, las cuales permiten hacer el seguimiento y detectar los límites establecidos.

Con estos mandos, que a su vez hacen función de joystick, podemos movernos y seleccionar objetos libremente por cualquier lugar. Además dispone de botones de acceso directo al menú y configuraciones.

### 3.2 Características del computador

Microsoft define dos tipos de equipos[19] que se muestran en la Tabla 5. Estos son los requisitos del equipo recomendado y los de un equipo "avanzado".

| | **Windows Mixed Reality** | **Windows Mixed Reality Ultra** |
|---|---|---|
| Sistema operativo | Windows 10 Fall Creators Update o posterior | Windows 10 Fall Creators Update o posterior |
| Procesador | Intel® Core™ i5 7200U (7.ª generación móvil), doble núcleo con tecnología Intel® Hyper-Threading habilitada o superior<br>AMD Ryzen 5 1400 3.4 Ghz (escritorio), | Intel® Core™ i5 4590 (4.ª de generación de escritorio)<br>AMD Ryzen 5 1400 3.4 Ghz |
| RAM | Doble canal DDR3 de 8 GB | DDR3 de 8 GB o superior |
| Espacio en disco | 10 GB | 10 GB |
| Tarjeta gráfica | GPU compatibles:<br>Tarjeta gráfica integrada Intel HD 620 o GPU integrada compatible con DX12 superior<br>GPU discreta NVIDIA MX150<br>GPU discreta Nvidia GeForce GTX 1050<br>GPU discreta Nvidia 965M<br>AMD Radeon RX 460/560. | GPU compatibles:<br>NVidia Geforce GTX 1060 o GPU discreta compatible con DX12 superior<br>AMD Radeon RX 470/570 o GPU discreta compatible con DX12 superior |
| Controlador de gráficos | Modelo de controladores de pantalla de Windows (WDDM) 2.2 | Modelo de controladores de pantalla de Windows (WDDM) 2.2 |
| Puerto de pantalla gráfica | HDMI 1.4 o DisplayPort 1.2 | HDMI 2.0 o DisplayPort 1.2 |
| Monitor | Monitor externo o integrado VGA (800x600) | Monitor externo o integrado VGA (800x600) |
| Tipo de USB | USB 3.0 de tipo A o tipo C | USB 3.0 de tipo A o tipo C |
| Tipo de Bluetooth | Bluetooth 4.0 | Bluetooth 4.0 |

***Tabla 5*** *Requisitos hardware.*

[19] https://support.microsoft.com/es-es/help/4039260/windows-10-mixed-reality-pc-hardware-guidelines

### Descripción de los equipos utilizados

Con el fin de comprobar su correcto funcionamiento y realizar una guía de configuración más amplia, la instalación se ha realizado en diferentes equipos:

- Ordenador Intel(R) core i5-7600, CPU 3.50GHz 16 GB, ram Windows 10 enterprise 1803 versión 17134.48. NVIDIA GeForce GT 710.
- Ordenador Intel(R) core i5-7600, CPU 3.50GHz 16 GB, ram Windows 10 enterprise 1809 versión 17763.134. NVIDIA GeForce GT 710.
- Portatil msi Windows 10 Pro 1803 version 17134.407. Intel(R) core i7-6700HQ, CPU 2.60GHz 8 GB ram, NVIDIA GeForce GTX 1060.

En el proceso de instalación se han probado tres equipos con configuraciones distintas, en dos de ellos ha dado problema y no hemos podido instalar el equipo. Los dos primeros equipos no cumplen con las especificaciones requeridas en cuanto a la tarjeta gráfica y no nos permite finalizar el proceso de instalación. Finalmente en el tercer equipo después de actualizar a la última versión de Windows y completar todas las actualizaciones de *Windows Update* la instalación se realizó con éxito. En el siguiente punto veremos el proceso de instalación de nuestro dispositivo Acer.

## 3.3 Instalación

La instalación de las gafas Acer requiere partir de la versión Windows 10 *Creators Update* actualizada como punto de partida. El proceso de instalación consta de los siguientes pasos [22]:

- Conectamos las gafas Acer a un puerto USB 3.0 y HDMI. Hay que asegurarse de conectar el puerto HDMI de la tarjeta gráfica y no el de la placa base.
- Se nos abrirá automáticamente la aplicación de realidad mixta de Microsoft (ya instalada por defecto en las versiones Windows 10 Creators Update), en caso de no abrirse se puede acceder desde el menú inicio de Windows**. Inicio> Portal de Realidad Mixta.** Vease Figura 17. Una vez la aplicación ha sido iniciada, se procederá a la descarga e instalación del programa.

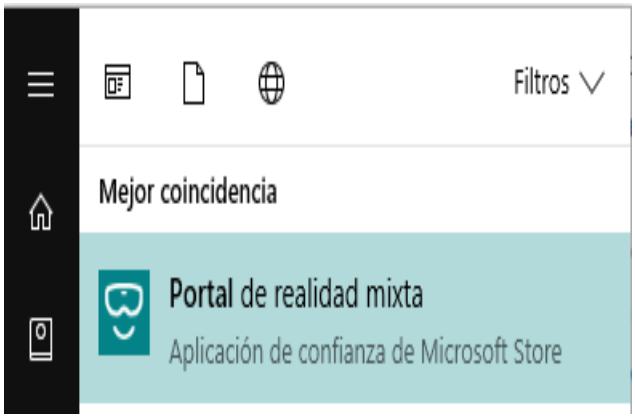



***Figura 17*** *Aplicación Portal realidad mixta.*

- La aplicación comprobará el hardware de la máquina para verificar la compatibilidad con la realidad mixta (verde: cumple los requisitos; amarillo: podría haber problemas; rojo: no cumple los requisitos). Figura 18.


***Figura 18*** *Comprobación de requisitos hardware.*

- Seguidamente se procederá a la configuración de los controladores de movimiento. Para ello se debe apretar sobre el botón de Windows durante dos segundos y después pulsar el botón de emparejamiento otros dos segundos.

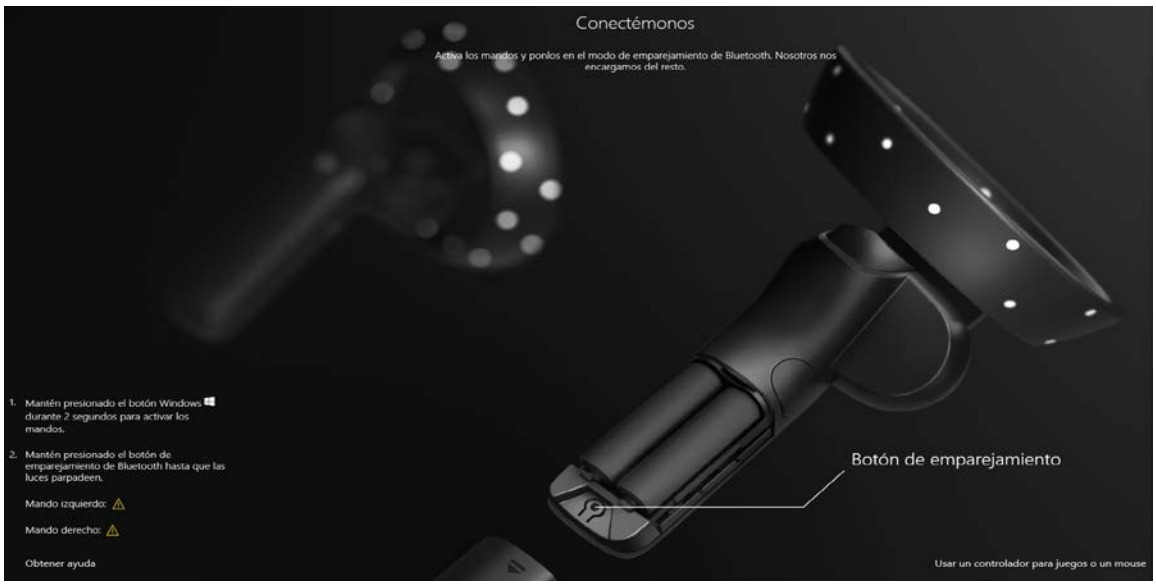



***Figura 19*** *Configuración mandos.*

- En este apartado tenemos dos opciones: la primera consiste en no crear límites, con lo que deberemos usar las gafas sentados (este método puede reducir la experiencia de usuario al usar la realidad mixta). La segunda opción es crear un límite, para ello, se debe recorrer un cuadrado alrededor de la habitación de 1.5 x 2 metros como mínimo.


***Figura 20*** *Configuración de límites.*

- Procedemos a la configuración de los limites (área en la que te puedes mover usando las gafas de realidad mixta; este límite es necesario ya que ayuda a evitar tropezar con obstáculos físicos). Ver figura 21.

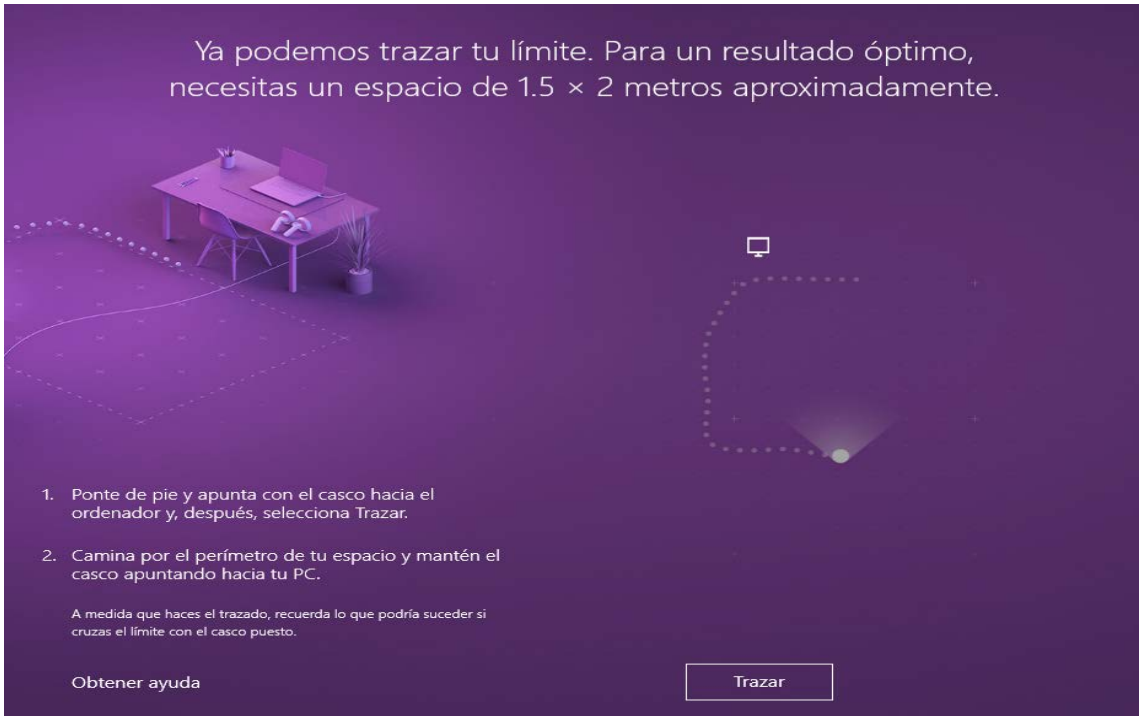



***Figura 21*** *Aplicar límites.*

- Opcionalmente podemos configurar comandos por voz a través del asistente de Microsoft Cortana.

- Una vez finalizada la configuración de los dispositivos se iniciaría un tutorial donde aprenderemos a utilizar los controles y a movernos por el entorno virtual.


***Figura 22*** *Tutorial portal realidad mixta.*

## 3.4 Aplicaciones de realidad mixta:

Microsoft Windows 10 ofrece una serie de aplicaciones preinstaladas para utilizar la realidad mixta. En este apartado vamos a analizar algunas de estas aplicaciones.

### 3.4.1 Portal de realidad mixta

*Windows Mixed Reality* cuenta con una interfaz propia, llamada portal de realidad mixta de Windows [23] que está disponible desde la versión de *Windows* 10 *creators update*. Una vez conectadas las gafas y colocadas en la cabeza, se nos abrirá automáticamente la aplicación de Windows que simula una casa desde la cual se puede acceder a todo el contenido de realidad mixta, véase figura 23.

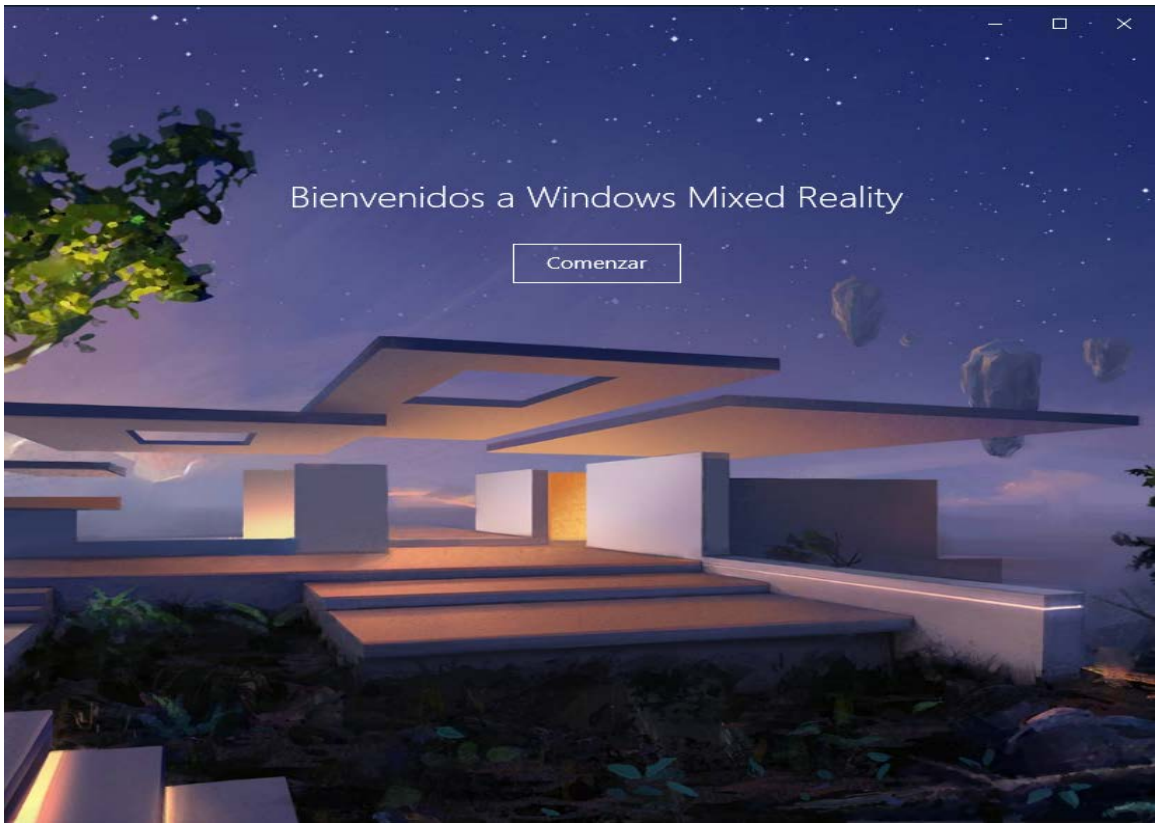



***Figura 23*** *Aplicación de la casa de realidad mixta.*

Desde esta interfaz es posible acceder a las aplicaciones multimedia más comunes como Skype, visualizar fotos, vídeos, ver películas, escuchar música, jugar e incluso navegar por Internet. Todas estas aplicaciones las podemos gestionar, ajustando su tamaño, cambiarlas de habitación, etc.


***Figura 24*** *Interior casa realidad mixta.*

Para poder moverte por las diferentes habitaciones de la casa hay dos opciones: utilizar los mandos, previamente configurados, con los que deberemos posicionarnos primero mirando hacia la dirección que queremos ir y seguidamente apuntar con el haz de luz manteniendo pulsado el joystick hacia esa dirección. La otra opción es a través de la combinación WASD y el ratón.

***Figura 25*** *Habitación casa realidad mixta.*

Otro aspecto a destacar es el nivel de decoración, el cual se puede ajustar a nuestro gusto, añadiendo o quitando elementos 3D, los cuales podemos importar o comprar. Además desde el portal de realidad mixta se puede acceder a la tienda de Microsoft donde descargar más contenido virtual.

### 3.4.2 Hologramas

Hologramas es una aplicación para el portal de realidad mixta de Microsoft. Dispone de una amplia galería de hologramas en 3D, ver figura 26. Estos hologramas se pueden editar, añadir textos, ajustar el tamaño y mover sobre un escenario virtual (en este caso la casa de realidad mixta). Además muchos de ellos tienen movimientos y sonidos establecidos, estos a su vez se pueden manipular a través de comandos de voz mediante el asistente de Windows Cortana.

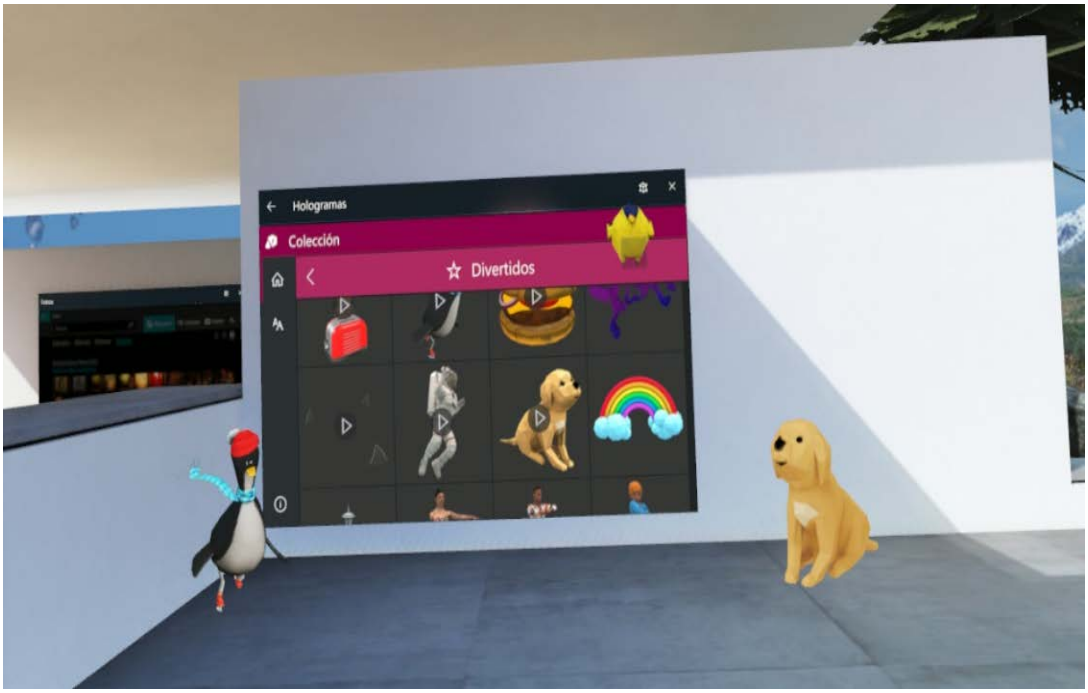



***Figura 26*** *Aplicación Hologramas.*

### 3.4.3 Visor 3D

Se trata de una aplicación de Microsoft que permite a los usuarios modelar cualquier objeto 3D y usarlo a través de la cámara del ordenador mezclando así elementos virtuales sobre la realidad. Dispone de una amplia galería de modelos 3D con movimiento y sonido.

### 3.5 Problemas y soluciones

Una vez configuradas las gafas de realidad mixta, procedemos a ver algunos puntos que pueden causar problemas [24]:

- Hay que asegurarse de conectar el cable HDMI a la placa de la tarjeta gráfica y no a la placa base del ordenador, de esta manera nos aseguramos de utilizar la tarjeta gráfica dedicada.

- Los controladores de movimiento usan Bluetooth para conectarse a tu PC. Si el PC no tiene Bluetooth, puedes comprar un adaptador de micrófono Bluetooth USB 4.0 y conectarlo al PC.

- Antes de configurar *Windows mixed reality*, en caso de utilizar una red corporativa, el administrador debe habilitarlo para tu organización.

- Es necesario tener una versión Windows 10 *Fall Creators Update* o posterior.

- El equipo debe tener las últimas actualizaciones de Windows.

- Son necesarios unos auriculares y micrófono para conectarlos a las gafas con el fin de proporcionar el audio para la inmersión dentro del mundo virtual.

### 3.6 Configuración de la instalación de Unity

Instalaremos los siguientes componentes de Unity, necesarios para desarrollar aplicaciones de realidad mixta mostrados en la Figura 27:

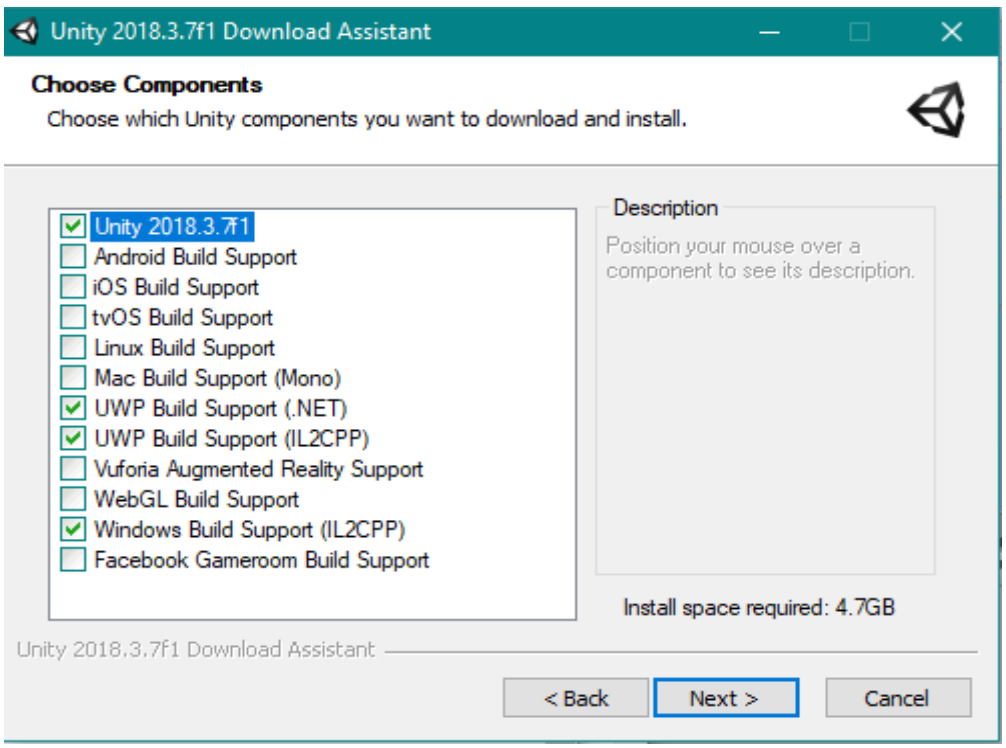



***Figura 27*** *Componentes Unity.*

- ***UWP Build Support* (IL2CPP)**

El componente IL2CPP [20] permite mejorar el rendimiento de aplicaciones UWP reduciendo el tamaño de sus aplicaciones. IL2CPP convierte el código de la API de scripts de Unity en código C++ estándar utilizando un compilador nativo.

- **Configurar las gafas de realidad mixta en Unity**

Para poder usar las gafas de realidad mixta, debemos configurar previamente en unity las configuraciones del jugador. **Edit > Project Settings > Player.**

Debemos activar en XR Settings la opción de Virtual Reality Supported y añadir el SDK Windows Mixed Reality instalando el plugin UnitySetup-Metro-Support-for-Editor. Ver figura 28.

Una vez configurado este apartado, al ejecutar cualquier simulación creada en Unity, se abrirá la aplicación de portal de realidad mixta, donde se podrá visualizar la aplicación creada en Unity.

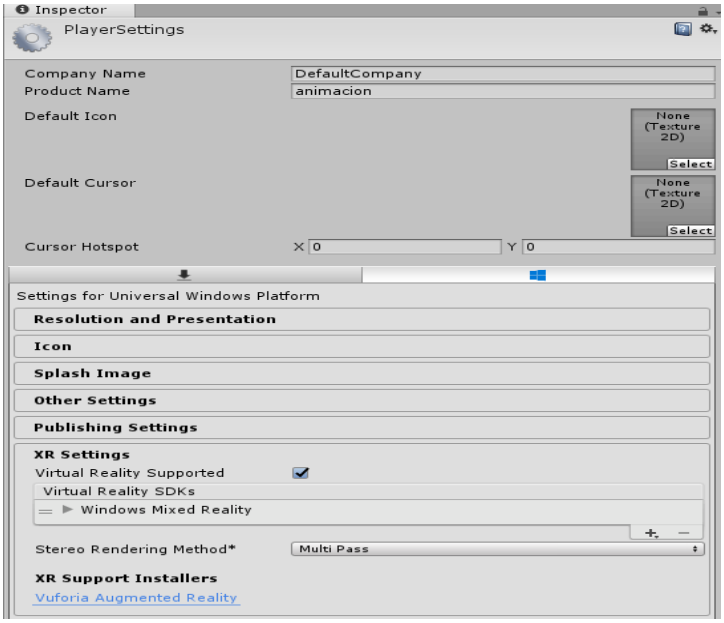



***Figura 28*** *Configuración Unity.*

- ***Build settings***

En este punto vamos a generar una solución UWP para una aplicación de realidad mixta. Para ello vamos a File > Build Settings. Seleccionaremos la opción de plataforma Universal Windows Plataform y pulsaremos el botón "*switch Plataform*" y dejaremos los siguientes parámetros según la figura 29, finalmente pulsaremos sobre el botón Build y seleccionaremos la carpeta donde guardar la solución.

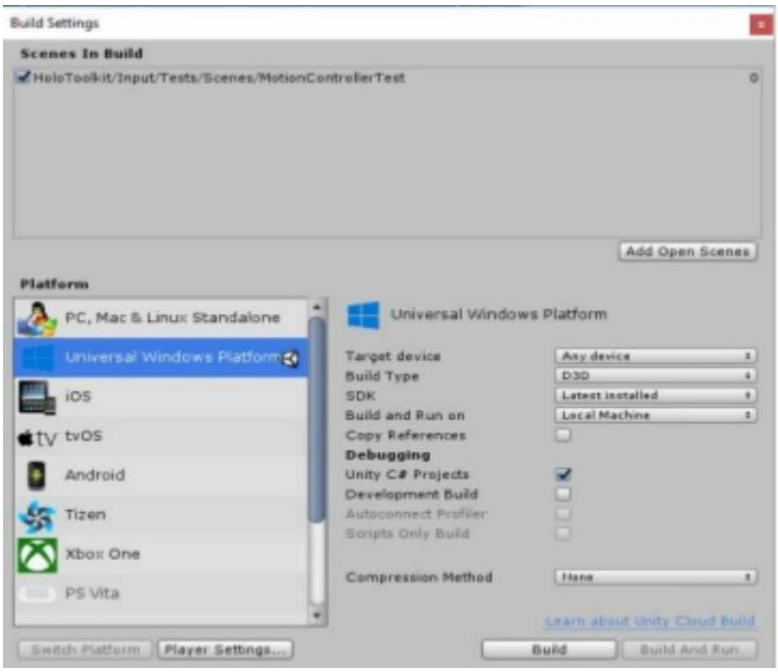



***Figura 29*** *Configuración solución en Unity.*

[20] https://docs.unity3d.com/Manual/IL2CPP.html

### 3.7 Importando MRTK a Unity

Nosotros utilizaremos el motor Unity, por lo que será necesario importar estas herramientas mediante el menú Assets > Import Package > Custom Package. Seleccionaremos los paquetes *foundation* y *examples* descargados de la web de microsoft MRTK [21], utilizaremos para este trabajo las versiones disponibles: "Microsoft.MixedReality.Toolkit.Unity.Foundation-v2.0.0-RC1" y "Microsoft.MixedReality.Toolkit.Unity.Examples-v2.0.0-RC1".

- El paquete **Foundation** (figura 30) contiene las funcionalidades básicas para el desarrollo de la realidad mixta, entre ellas se encuentran las interfaces, clases, entradas de audio, de control de interfaz de usuario, dependencias del sistema etc.

- En cuanto al paquete **Examples**, este contiene los activos y las escenas principales para empezar a utilizar la realidad mixta de una manera más rápida, debido a que dispone de elementos creados listos para utilizarse.

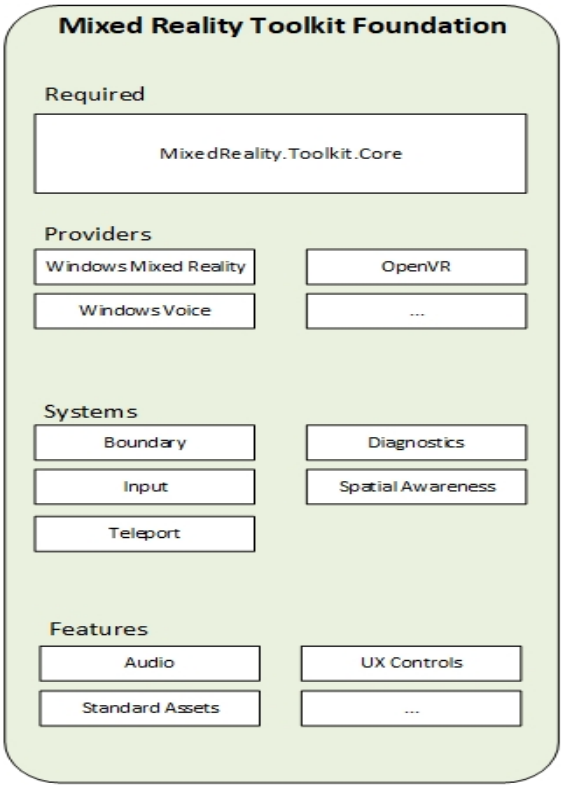



***Figura 30*** *Paquetes de Foundation.*
*Fuente: docs.microsoft.com*

Después de importar las librerías necesarias en Unity aparecerá un nuevo menú (figura 31) dónde podremos configurar ajustes como: los dispositivos de entrada, la cámara, límites, traslación del usuario y estadísticas de rendimiento.

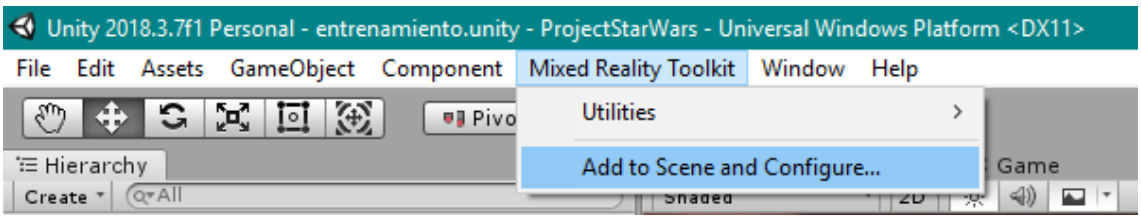



***Figura 31*** *Menú Mixed Reality Toolkit de Unity*

[21] https://github.com/microsoft/MixedRealityToolkit-Unity/releases/tag/v2.0.0-RC2.1

### 3.8 Configurar los controladores de movimiento en Unity

Después de haber añadido el *Toolkit* de realidad mixta podemos acceder a los mandos del dispositivo y añadir las funcionalidades adecuadas. Para ello debemos ir a al menu *Edit* > *Project Settings* > *Input* y seleccionaremos el identificador adecuado del controlador de movimiento al que queremos acceder (AXIS_X) donde X será el botón del controlador [25].

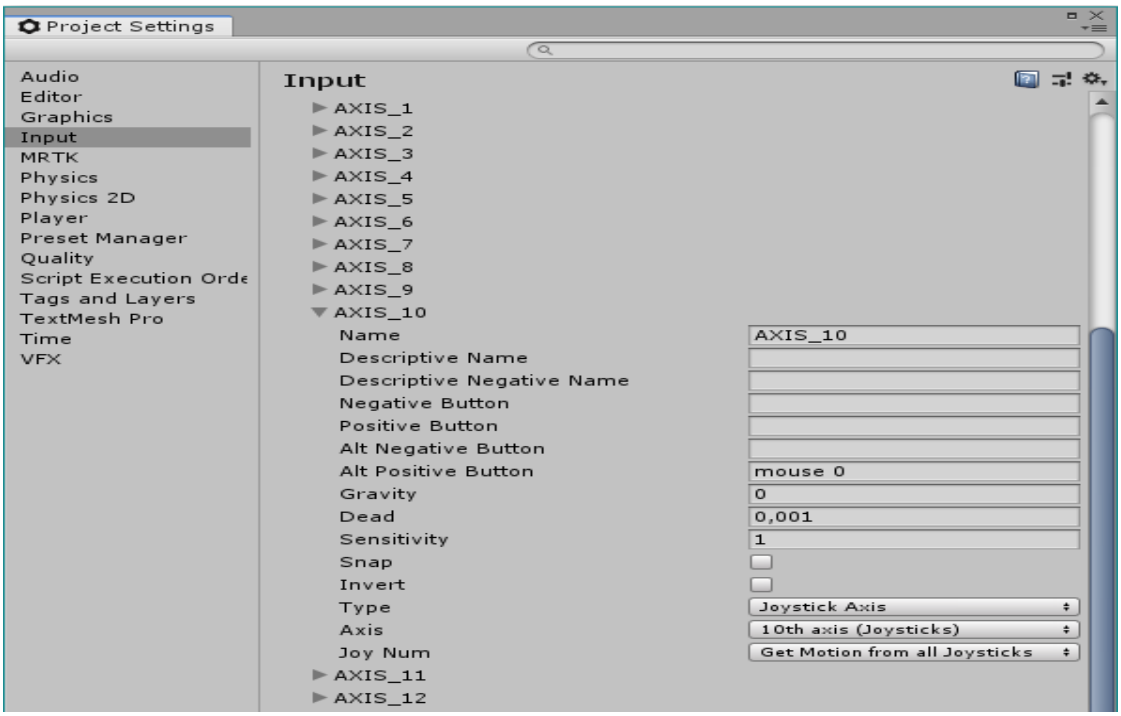



***Figura 32*** *Configurar sistemas de entrada en unity.*

Los controladores de movimientos tienen asignados diferentes funciones para cada botón. Estas funcionalidades estarán presentes en todas las escenas [26]:

- **Boton *Select*:** Cada vez que pulsamos este botón se disparará un rayo láser hacia nosotros.
- **Boton *Grasp*:** este botón está configurado, mediante *script*, para reanudar el juego una vez finalizado.
- ***Home*:** cuando se pulsa este botón se vuelve a la casa de realidad mixta, saliendo de la aplicación.
- ***Thumbstick*:** mediante este joystick nos podemos mover por la aplicación

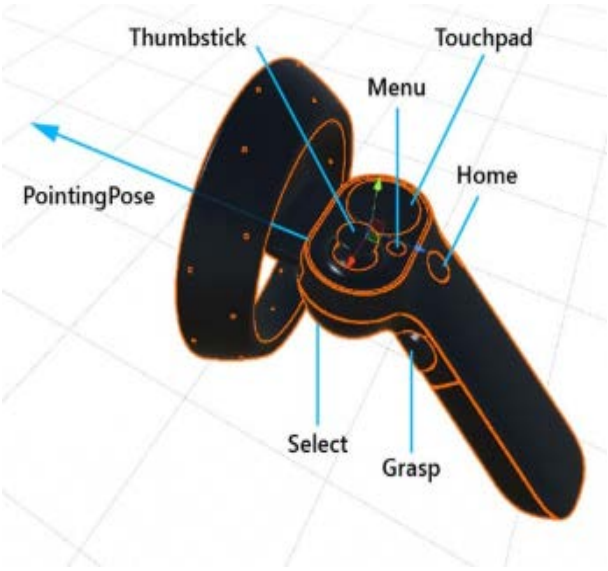



***Figura 33*** *Botones motion controllers.*

Una vez configurado los botones de los controladores a los que vamos a acceder, tenemos que programar su comportamiento a través de scripts aplicados al juego. En la figura 34 accedemos al componente Input del joystick sobre el botón AXIS_15 dentro del método *void Update()* para que se ejecute en cualquier momento del juego y definimos su comportamiento, en este caso instanciamos el disparo y su sonido cada vez que pulsemos sobre el botón 15 del controlador de movimiento.

```
void Update ()
{
    if (Input.GetKey("joystick button 15") && Time.time > nextFire)
    {
        nextFire = Time.time + fireRate;
        Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
        GetComponent<AudioSource>().Play();
    }
```

***Figura 34*** *Ejemplo código input motion controllers.*

### 3.9 Control de cámara

El control de la cámara se gestiona a través del MRTK, el cual se encarga de crear el componente y seguir al usuario a través de la escena. El SDK recoge los datos de los sensores del dispositivo que el usuario tiene en su cabeza y ajusta el movimiento del usuario en el juego. Por lo que cada vez que giramos la cabeza el sensor sabe exactamente hacia dónde estamos mirando.

Esto es debido a que el “Headset” de Acer utiliza la tecnología 6DoF [28], Figura 35. Este sistema permite libertad de movimiento, posiciona al usuario mediante tres vectores de traslación y tres vectores de rotación que ubican al usuario, de forma unívoca, en un sistema 3D rastreando la orientación de la cabeza y el movimiento del usuario a través de un espacio físico limitado.

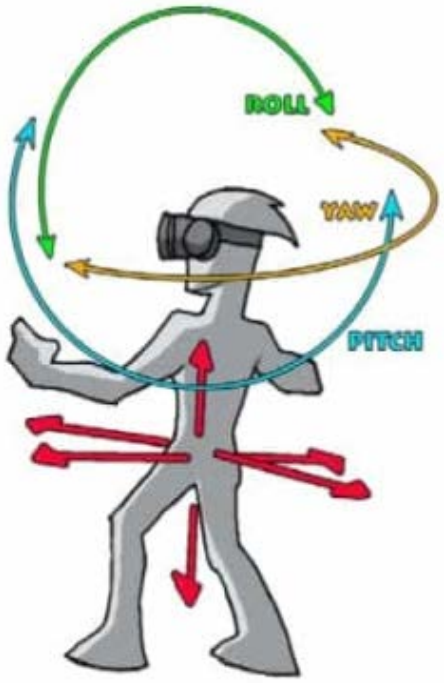



***Figura 35*** *6-DoF. Fuente [24]*

### 3.10 Ejemplo de uso de Unity 3D con el MRTK

Después de configurar el entorno de Unity y agregar las librerías de realidad mixta estamos listos para empezar a realizar pruebas de qué es lo que podemos hacer con las gafas de Acer.

Veamos ejemplos básicos para desarrollar una aplicación de RM utilizando el SDK MRTK de Microsoft para Unity [27]. El principal problema viene por la escasa información y aplicaciones que encontramos para esta realidad y sobre todo por tratarse de un SDK en versión *release candidate* la cual no está aún cerrada y puede contener algunos problemas. Dentro del kit de herramientas de realidad mixta disponemos de una serie de scripts y objetos prefabricados que se pueden encontrar en el kit de realidad mixta encontramos varias escenas disponibles en el paquete Examples. Podemos destacar la escena: "HandInteractionExamples.unity[22]", la cual contiene distintos tipos de interacciones y controles sobre distintos elementos ya prefabricados listos para ser utilizados.

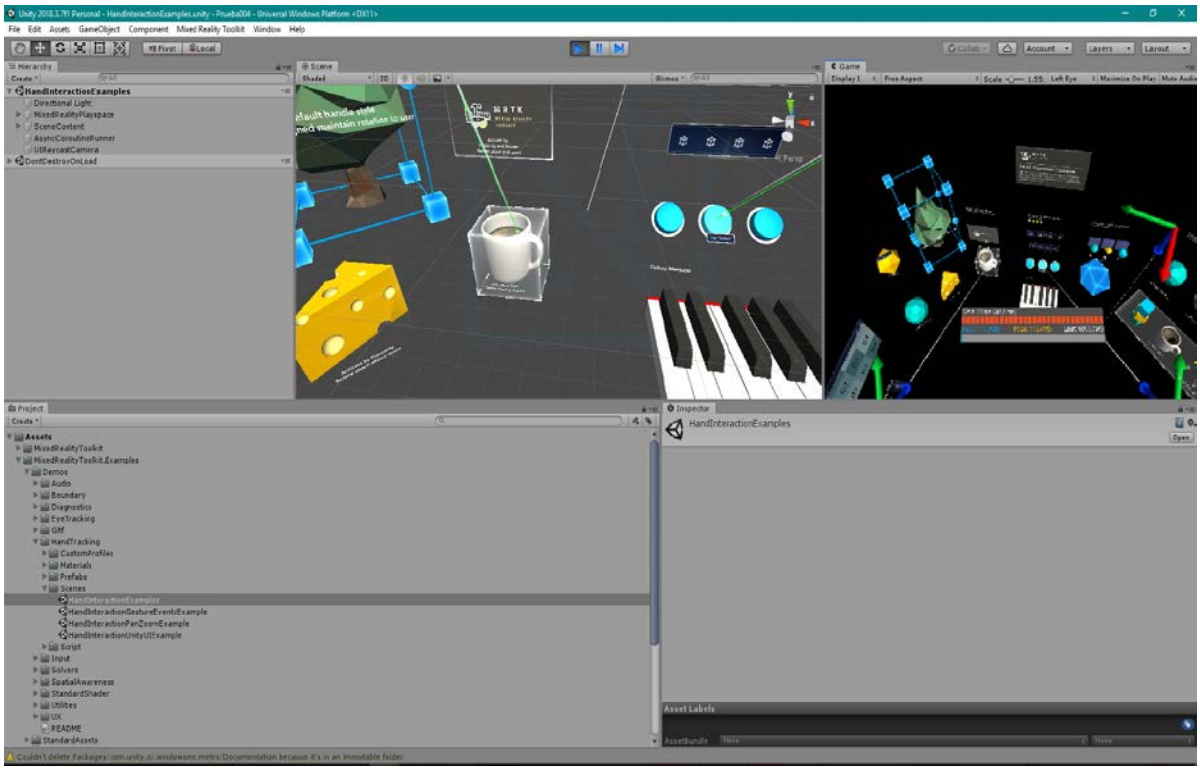


***Figura 36*** *Desarrollo en Unity.*

Los componentes prefabricados y *scripts* del paquete *Examples* pueden ser utilizados en otras escenas. De esta forma el desarrollador solo tiene que reprogramar la funcionalidad deseada sobre unos componentes y clases ya creados.

[22] https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/README_HandInteractionExamples.html

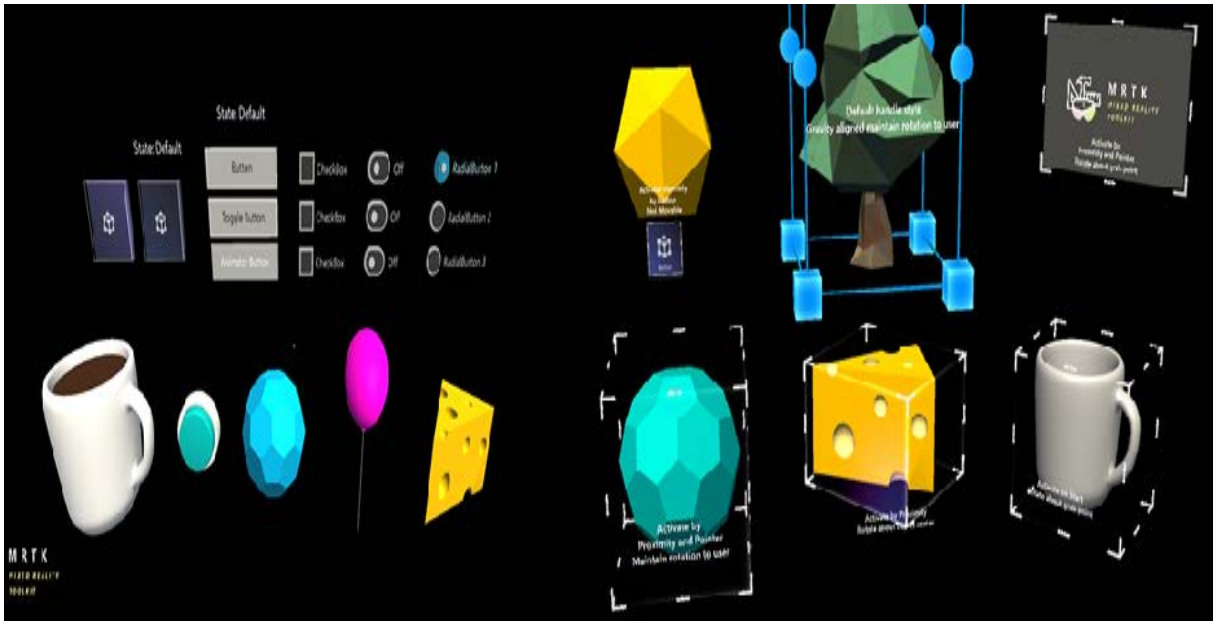



***Figura 37*** *Objetos prefabricados en unity MRTK.*

Podemos ver un ejemplo de los objetos prefabricados en la figura 37. Entre ellos contienen los siguientes elementos:

- *Button*: una colección de botones prefabricados, que al ser presionados o mantenidos por los mandos, realizan una serie de eventos personalizados. Además, ofrecen la posibilidad de ser activados a través de comandos de voz.

- *Interactables*: Este script permite cambiar el estado visual de un objeto (color, tamaño) al estar enfocado por las gafas o al presionarlo con los mandos.

- *Object collection*: se trata de un script para ayudar a diseñar una colección de objetos de diferentes formas (planos, cilindros, esferas…)

- Pizarra: son ventanas de texto plano 2D, estos elementos ya prefabricados, pueden ser utilizados para añadir menus o ventanas informativas de texto al usuario.

- *Manipulation Handler*: Este script aplicado a un objeto, permite mover, rotar y escalar un objeto.

# 4. Diseño del proyecto

En este apartado vamos a plantear los pasos realizados para el diseño de esta aplicación de realidad mixta desarrollada en Unity.

Para ello partiremos del Diagrama 1 en el cual podemos observar los pasos necesarios para la realización de este proyecto en Unity. En el primer paso partimos del MRTK como SDK para construir aplicaciones de RM en Unity. Seguidamente, después de importar las librerías necesarias podemos empezar a crear las escenas de juego, las cuales formarán la estructura y el entorno de la aplicación. En el tercer paso procedemos a configurar los controles de usuario (*Motion controllers*) en Unity para poder interactuar con la aplicación a través de los botones. En el cuarto paso añadiremos los objetos de juego (*GameObjects*)[23] y los modelos 3D que formaran parte de la aplicación. En el quinto y último paso desarrollaremos la implementación a través de *scripts* en C# que determinarán el comportamiento de los objetos de juego, finalmente realizaremos pruebas para comprobar el correcto funcionamiento.


***Diagrama 1*** *Flujo del desarrollo de la aplicación RM.*

[23] https://docs.unity3d.com/es/current/Manual/GameObjects.html

### 4.1 Herramientas utilizadas

En este apartado vamos a ver las herramientas necesarias para poder desarrollar en la realidad mixta de Microsoft[24]. Véase la tabla 6.

|  Windows SDK 18362+ |  Unity 2018.3+ |  Kit herramientas realidad mixta (MRTK) |  Visual Studio 2017 |
|---|---|---|---|
| Sistema operativo Windows 10 Fall Creators. | Motor Unity 3D con soporte para realidad mixta. | Kit de desarrollo multiplataforma de código abierto para aplicaciones de RM. | Visual Studio para editar e implementar código C#. |

***Tabla 6*** *Herramientas desarrollo realidad mixta.*

El SDK 18362+ proporciona las bibliotecas y herramientas para compilar aplicaciones de *Windows Mixed reality.* Por otra parte el motor gráfico de Unity nos ofrece la manera de crear aplicaciones de RM de una forma más rápida gracias a las librerías de MRTK que proporcionan las herramientas básicas para el desarrollo y uso de dispositivos de RM. Finalmente, necesitamos un entorno de desarrollo para escribir, depurar e implementar código mediante scripts en C# que darán la funcionalidad necesaria a nuestra aplicación.

### 4.2 Mecánica y objetivo del juego

Se trata de una aplicación de un solo jugador. Consiste en tres prototipos ambientados en el universo de *Star Wars* separados en tres escenas diferentes:

- **Entrenamiento Jedi**: Juego en primera persona. Esta escena consiste en intentar bloquear los disparos con un sable láser que son producidos por una esfera situada encima del jugador, la cual sigue al jugador en todo momento.

- **Defensa Jedi:** Este mini juego, es otra escena en la cual el jugador deberá destruir los enemigos, que son generados automáticamente con su espada láser.

- **Piloto de caza**: Por último, en esta escena, pilotaremos un caza *X-Wing* de *Star Wars* en tercera persona, controlando tanto su posición como el disparo con el objetivo principal de destruir la estrella de la muerte, además el jugador deberá ir destruyendo los cazas enemigos generados por la aplicación.

[24] https://docs.microsoft.com/es-es/windows/mixed-reality/install-the-tools

En los tres juegos el jugador deberá defenderse del ataque de los enemigos para conseguir sobrevivir. Además podrá pausar el juego en cualquier momento y salir de él cuando lo considere oportuno. El juego terminará cuando el jugador sea eliminado o cuando pulse “salir” en el menú del juego.

Durante el juego el jugador deberá destruir el mayor número de enemigos posibles en cada una de las escenas del juego. Por cada enemigo derrotado se incrementará su puntuación sin límite de tiempo. Cuando el jugador es alcanzado por un enemigo, el jugador será eliminado, aparecerá por pantalla el mensaje de *Game Over* y deberá empezar de nuevo con cero puntos en su marcador, esto finalizará el juego, obligando al jugador a empezar una partida nueva.

### 4.3 Estructura del juego

El juego se estructura como tres aplicaciones independientes dentro de escenas, cada una de ellas corresponderá a un juego distinto dentro de la misma temática. Al iniciar la aplicación el usuario podrá navegar a través del menú principal de la aplicación hacia las siguientes escenas (demos) consiguiendo un diseño flexible y escalable. En el Diagrama 2 podemos ver la secuencia de transiciones posibles entre las diferentes escenas del juego desde el inicio de la aplicación hasta su finalización pasando por las diferentes escenas del juego.

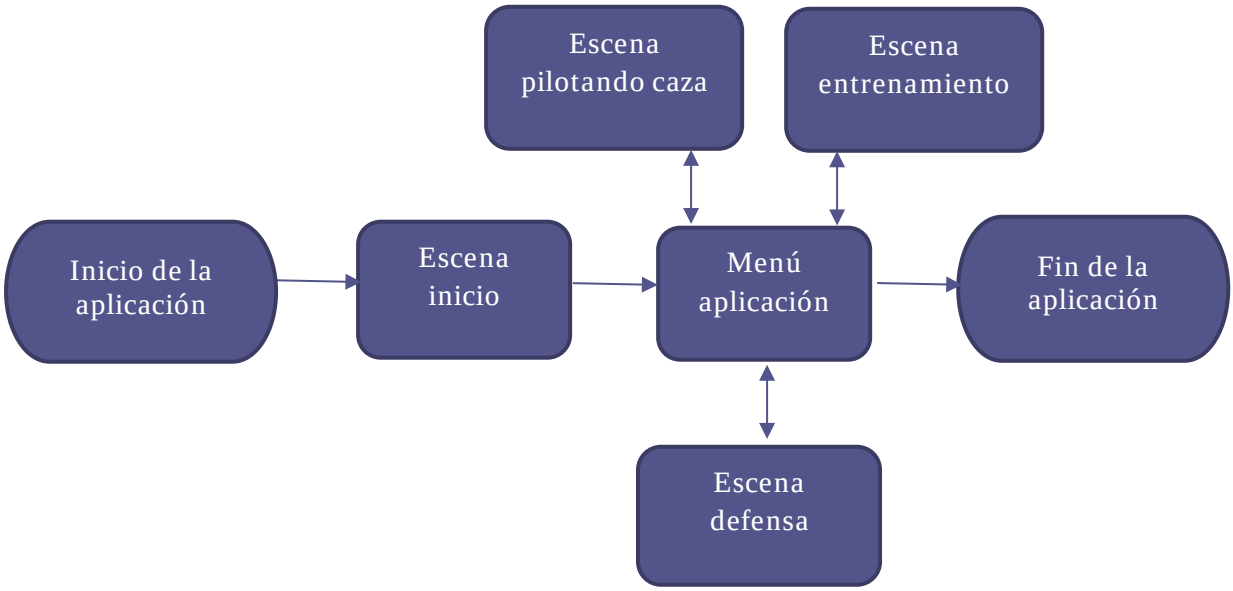



***Diagrama 2*** *Secuencia de la aplicación.*

Todas las escenas comparten una serie de componentes principales que permiten el funcionamiento del juego. Estos componentes podemos clasificarlos en tres grupos, como se muestra en el Diagrama 3:

- Sistemas de entrada: los *Motion controllers*, son los encargados de interactuar con el jugador, permiten movernos a través de la escena, atacar, coger y seleccionar objetos.

- Lógica de la aplicación. Consta de cuatro componentes principales, el primero de ellos es la librería MRTK que se encargara de conectar las gafas y controladores Acer con el motor Unity. También permitirá controlar la cámara del jugador actualizando en cada momento la posición del mismo dentro de la escena. El *Game Controller* es una clase creada por nosotros que contendrá el comportamiento del juego. Finalmente tenemos el componente Marcador que se encargará de realizar un conteo y reseteo de los puntos ganados.

- Sistemas de salida. En este apartado disponemos de dos componentes: el *Headset* que es el dispositivo que el usuario utiliza para introducirse dentro del juego y el sonido que permite al jugador una inmersión más detallada del juego.

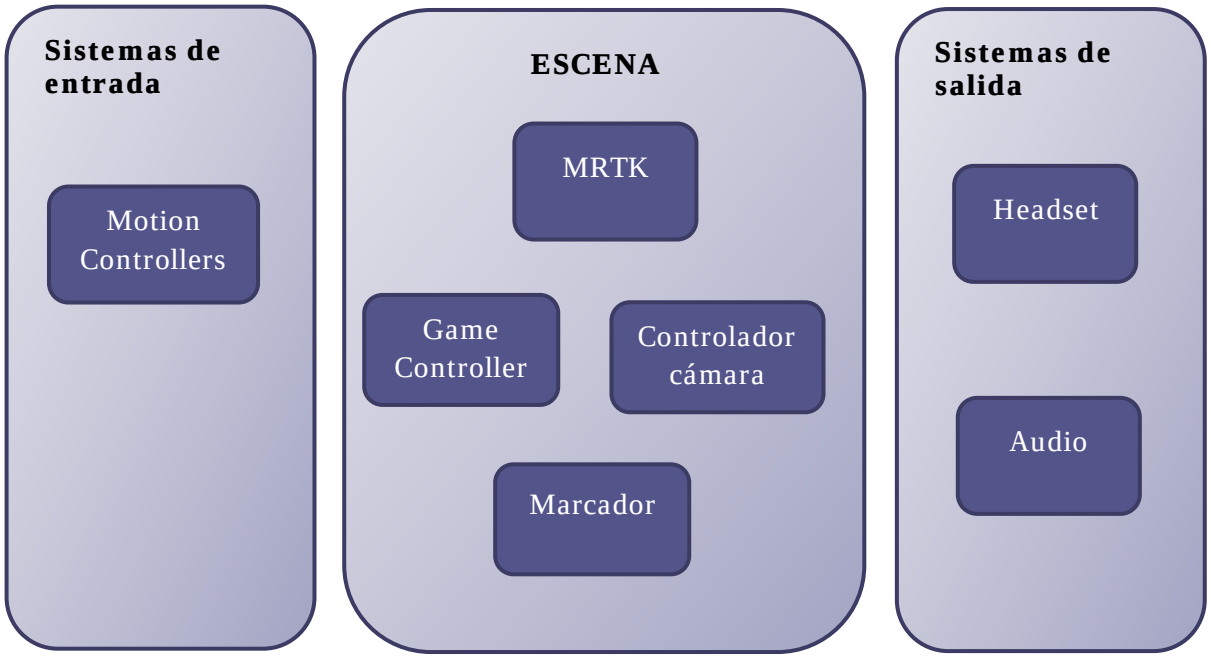



***Diagrama 3*** *Componentes principales.*

### 4.4 Casos de uso

Finalmente, vamos a describir las interacciones del usuario. Para ello realizaremos tres casos de uso, que tienen la finalidad de mostrar de manera clara y sencilla un diagrama, indicando las distintas acciones que un usuario puede realizar dentro de la aplicación.

En este caso podemos distinguir un caso de uso general para el menú del juego y dos casos de uso distintos para las escenas del juego. Estos casos de uso ayudarán a definir el comportamiento de la aplicación y los requisitos necesarios para el desarrollo del juego.

En la Figura 38 tenemos el caso de uso del menú del juego. El usuario puede realizar distintas acciones como la transición entre escenas o salir del juego.

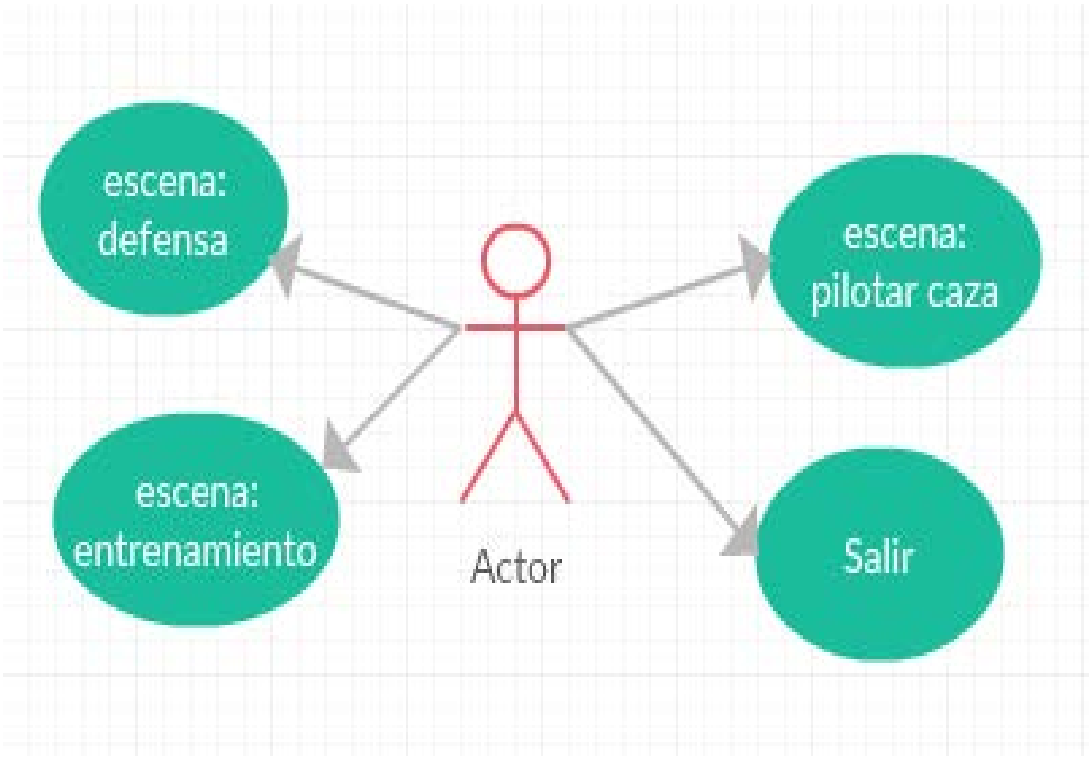



***Figura 38*** *Caso de uso "Menú usuario".*

En el segundo caso de uso, véase la Figura 39, observamos qué acciones puede realizar el usuario dentro de las escenas de juego “entrenamiento jedi” y “defensa jedi”.

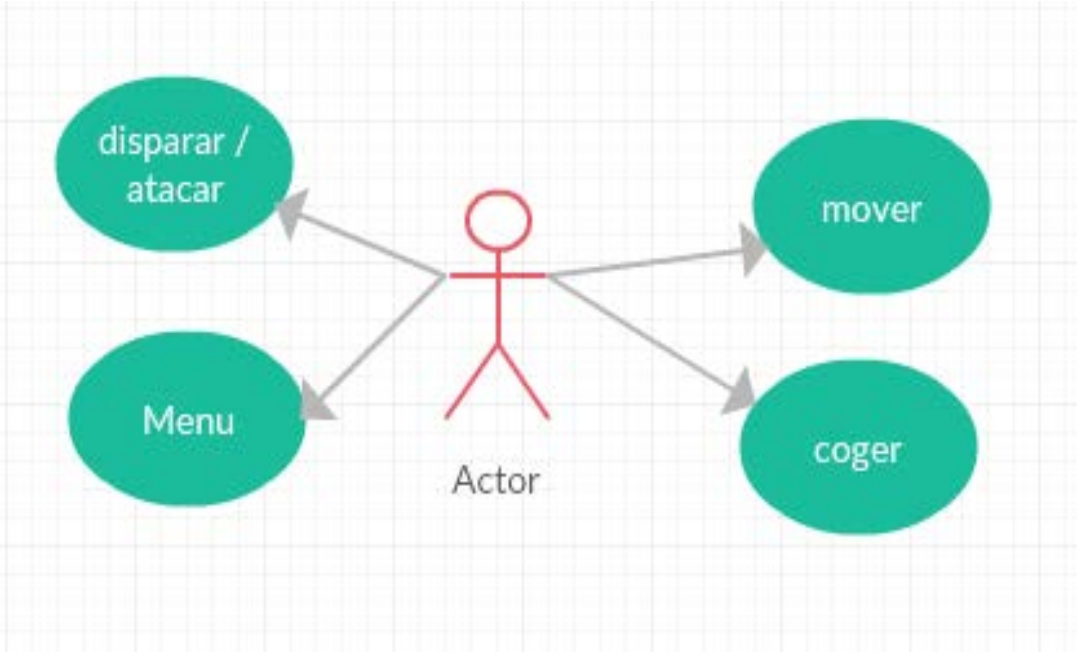



***Figura 39*** *Caso de uso escenas entrenamiento jedi y defense jedi.*

En la escena “piloto de caza”, podemos observar el siguiente diagrama de uso, véase Figura 40.

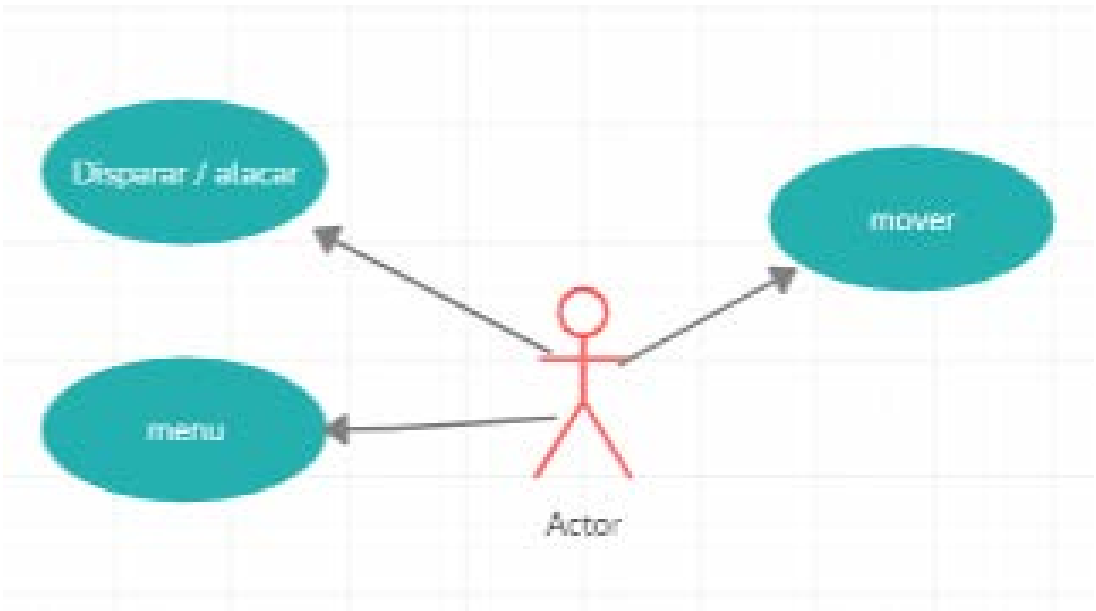



***Figura 40*** *Caso de uso escena piloto de caza.*

## 4.5 Gestión del proyecto

Para el desarrollo de este trabajo utilizaremos una metodología de desarrollo iterativa e incremental basada en prototipos ejecutables. Cada iteración, produce una versión del producto mejorada, hasta llegar a una versión final. Esta metodología dividirá el proyecto en tres fases, las cuales serán validadas al finalizarse con el fin de comprobar su correcto funcionamiento.

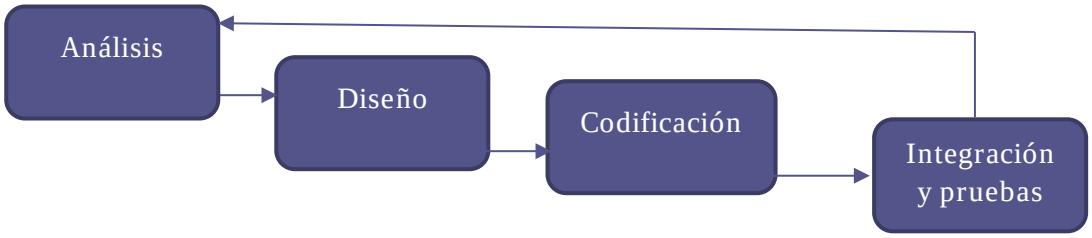



***Diagrama 4*** *Fases del Proyecto.*

- Análisis, en este punto se ha hecho un estudio sobre los diferentes sistemas de desarrollo de realidad mixta y qué herramientas utilizar.

- Diseño, en esta fase describimos cómo va a ser la aplicación, describiendo sus diagramas de flujo y casos de uso, con el fin de facilitar el desarrollo de la aplicación.

- Codificación, es la parte encargada de implementar la lógica de la aplicación, esta parte la desarrollaremos más adelante en el capítulo 5.

- Integración y pruebas, esta fase se encarga de comprobar el correcto funcionamiento de la aplicación, detectar posibles fallos y solucionarlos.

### 4.5.1 Planificación

En este apartado vamos a realizar una planificación del proyecto, con el objetivo de gestionar las tareas y los tiempos necesarios para cada una de ellas. Para plasmar la gestión del tiempo, hemos optado por un diagrama de Gantt. Figuras 41 y 42. El diagrama de Gantt, es una herramienta con la cual podemos planificar y programar tareas para el desarrollo de un proyecto, con el que podremos realizar un control del tiempo para cada tarea.

Este proyecto está planificado en tres fases, las cuales vamos a describir a continuación. La primera fase, consta sobre cómo realizar la gestión del mismo, estudiando los recursos necesarios para su creación, la información necesaria y el tiempo preciso para adaptar y configurar los equipos a un estado óptimo para el desarrollo del proyecto.

La segunda fase, consta sobre el diseño, desarrollo y planificación del proyecto. En esta etapa, una vez analizada toda la información se realizará la estructura necesaria, así como las tecnologías y herramientas necesarias para la implementación y puesta en funcionamiento de este proyecto.

En la última fase, se realizaran las pruebas necesarias, las cuales comprobarán el correcto funcionamiento de la aplicación.

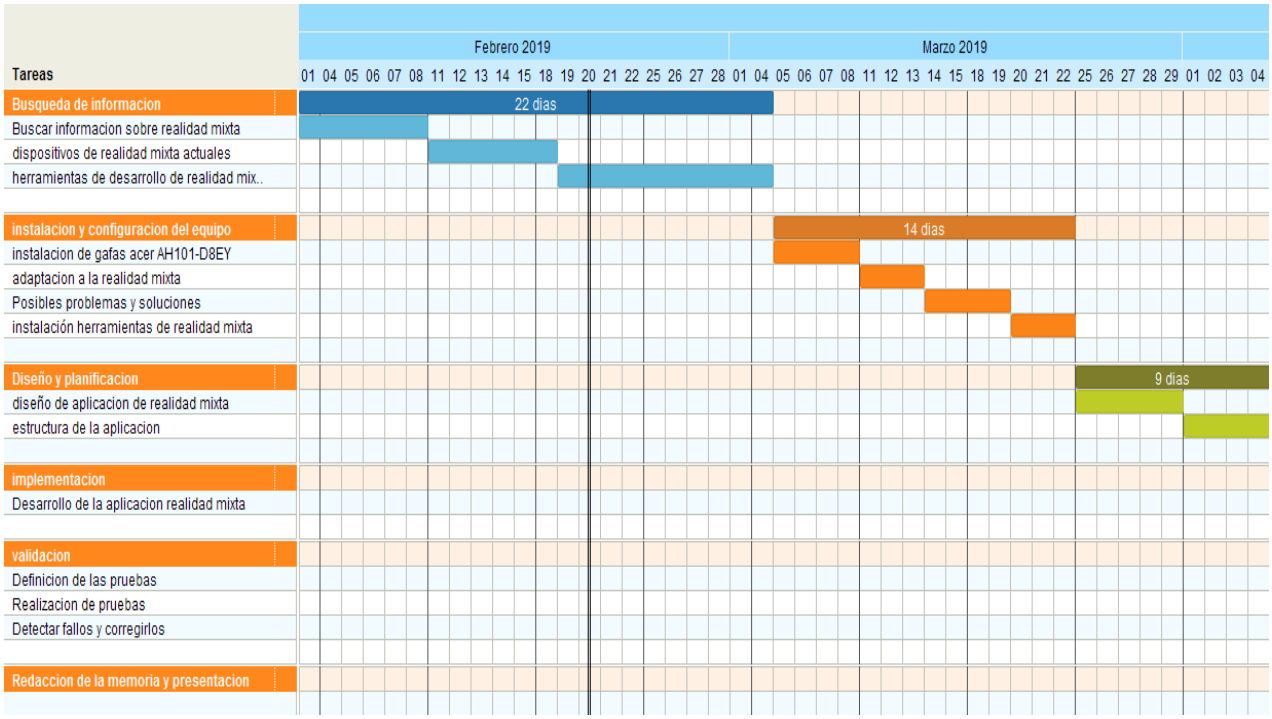



***Figura 41*** *Diagrama de Gant parte 1.*

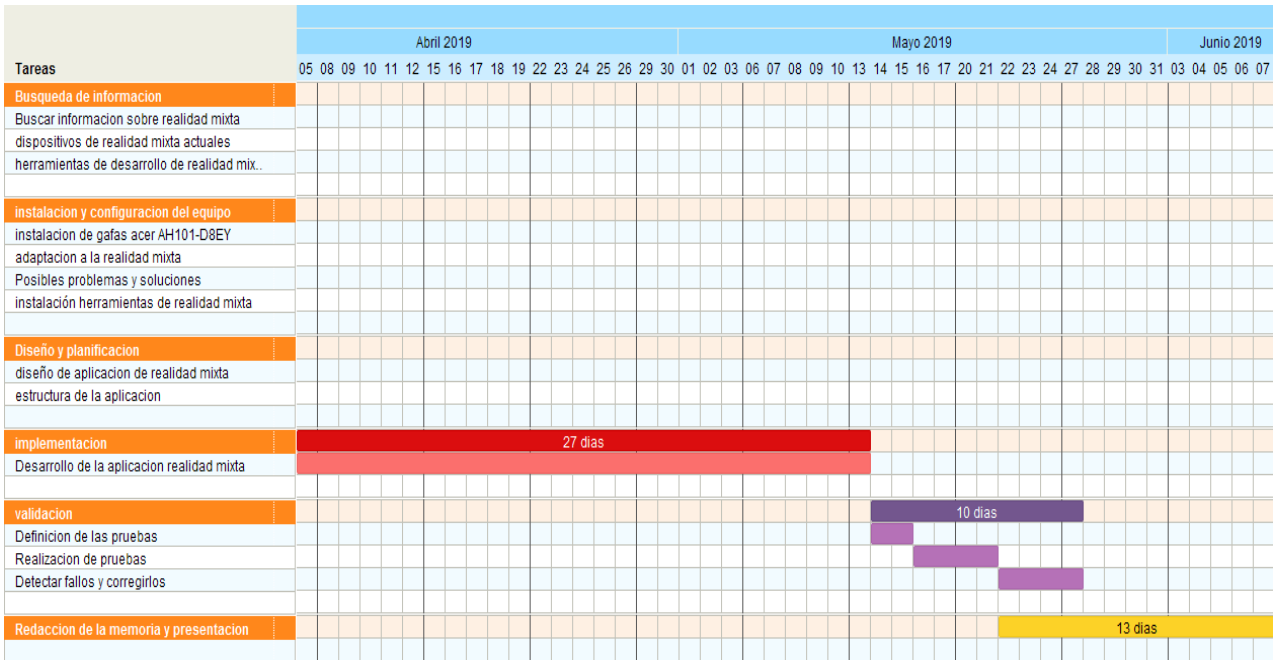



***Figura 42*** *Diagrama de Gant parte 2.*

### 4.5.2 Presupuesto

Para una buena gestión de los recursos y para el desarrollo de este proyecto, es importante desglosar los costes necesarios para la realización del mismo. Las tablas 7,8 y 9 reflejan los gastos desglosados.

| ***Costes directos*** | | |
|---|---|---|
| ***Nóminas*** | **Tiempo** | **Precio** |
| *Programador* | 12 semanas | 300 €/semana |
| ***Costes Material*** | | |
| *Ordenador* | | 600 € |
| *Sistema operativo Windows 10*[25] | | 145 € |
| *Tarjeta gráfica GeForce GTX 1050Ti D5 4GB GDDR5*[26] | | 174 € |
| *Licencia Unity* | | 0 € |
| *Headset Acer mixted reality*[27] | | 360 € |
| ***Total*** | | 4879€ |

***Tabla 7*** *Costes directos.*

| ***Costes indirectos*** | | |
|---|---|---|
| ***Concepto*** | **Tiempo** | **Precio** |
| *Conexión a internet* | 3 meses | 60 €/mes |
| ***Total*** | | 180 € |

***Tabla 8*** *Costes indirectos.*

| ***Costes totales*** | |
|---|---|
| ***Concepto*** | **Precio** |
| *Costes directos* | 4879 € |
| *Costes indirectos* | 180 € |
| ***Total*** | 5059 € |

***Tabla 9*** *Costes totales.*

[25] https://www.microsoft.com/es-es/p/windows-10-home/d76qx4bznwk4/1NT3
[26] https://www.pccomponentes.com/gigabyte-geforce-gtx-1050ti-d5-4gb-gddr5
[27] https://www.amazon.es/Acer-AH101-Mixed-auricular-controladores-inal%C3%A1mbricos/dp/B0763N26PW

# 5. Desarrollo del proyecto

Este proyecto tiene como objetivo final la construcción de un juego de realidad mixta. Este juego consistirá en desarrollar tres prototipos distintos dentro de una misma temática de juego. Estos prototipos son tres escenas distintas con ciertos componentes comunes pero con un comportamiento y jugabilidad distintas.

### 5.1 Construcción del escenario

Para empezar, debemos abrir un nuevo proyecto en Unity 3D, con las configuraciones de los capítulos 3.6, 3.7 y 3.8. El primer paso consistirá en crear un escenario de juego. Para ello tendremos que crear una escena y agregarle los componentes. En la escena creada añadiremos una plataforma que hará la función de suelo. También necesitaremos un cielo o *skybox* (caja que contiene las imágenes del fondo de la escena).

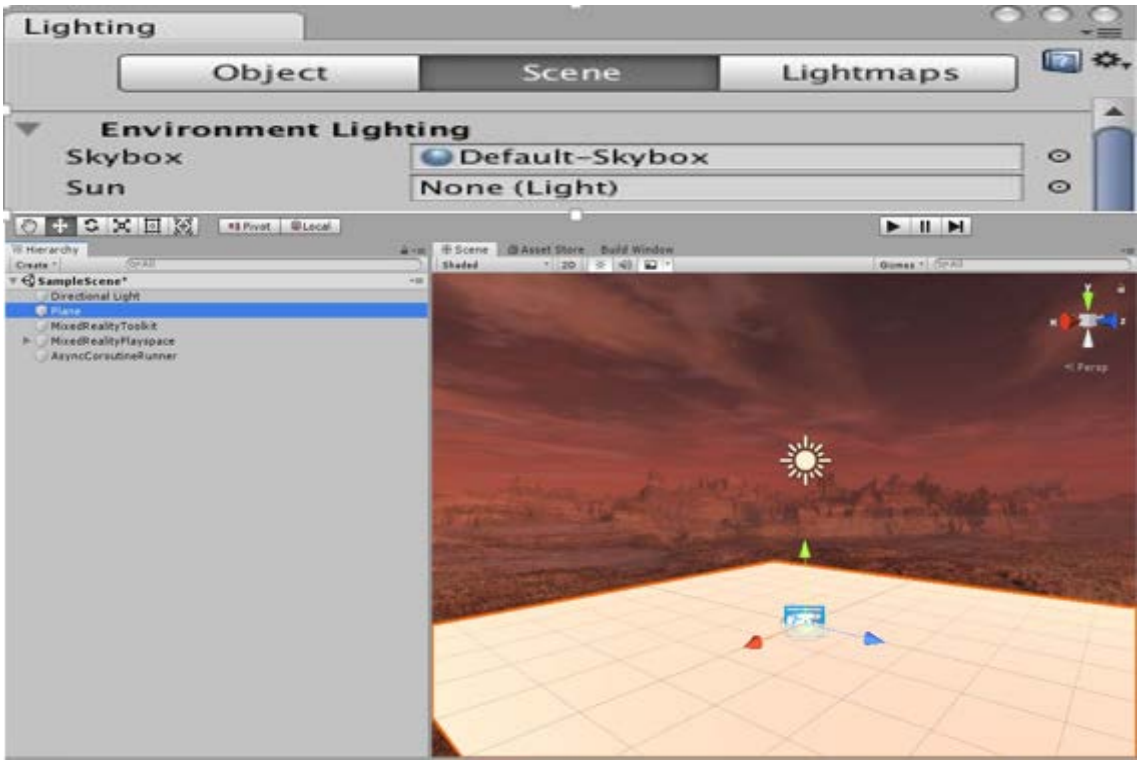



***Figura 43*** *Escenario Unity.*

Las escenas “entrenamiento jedi” y “defensa jedi” compartirán el mismo escenario y *skybox* con el fin de dar una imagen más consistente del juego. En la tercera escena “piloto de caza”, nos encontramos un *skybox* distinto simulando el espacio, pero compartiendo los modelos 3D y temática de las anteriores escenas.

## 5.2 Implantación de los modelos 3D en las escenas

En el desarrollo de este juego se han utilizado diversos modelos 3D, adquiridos mediante modelos 3D libres y desde la tienda de Unity [30]. Estos modelos aparecen en las diferentes escenas del juego compartiendo diseño y funcionalidad. A continuación vamos a ver los objetos 3D que se utilizarán a lo largo de las escenas.

### 5.2.1 Espada láser

Para esta aplicación necesitamos un modelo 3D de un sable láser[28] con el que interactuaremos en el juego. Este componente será utilizado en las escenas "entrenamiento jedi" y "defensa jedi" siendo la parte fundamental, tanto para atacar como para defender. Partimos de un objeto prefabricado en el que añadiremos un haz de luz. Para sustituir la espada por el objeto del controlador de movimiento, necesitamos añadir la espada prefabricada dentro del componente *GizmoRight* (componente del kit de realidad mixta asociado al mando derecho), ver figura 44. Con esto conseguimos mover la espada láser con el controlador de movimiento.

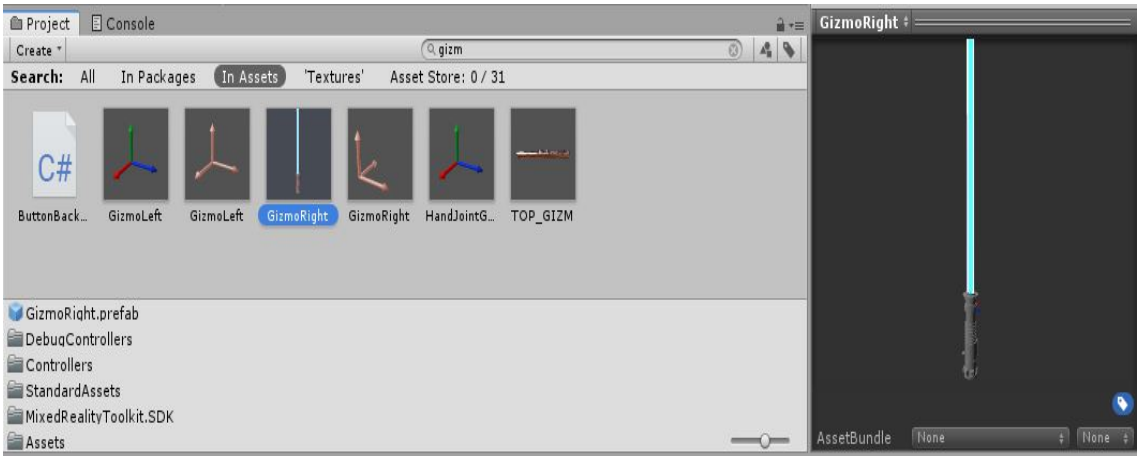



***Figura 44*** *Elemento prefab de unity.*

La espada consta de un cilindro al que se le ha dado una textura y luminosidad para aparentar una espada láser. Para detectar las colisiones a través de la espada necesitamos añadir un componente de Unity llamado *Mesh Collider (figura 45)*, y activar la opción *Is Trigger* que se encargara de detectar cuando otro objeto entra en su área, de esta forma a través del script DestroyByContact (figuras 49 y 50), cuando un objeto que tenga este script entre en contacto con la espada, el objeto será destruido.

[28] https://free3d.com/3d-model/lightsaber-all-colors-and-2-types-778009.html

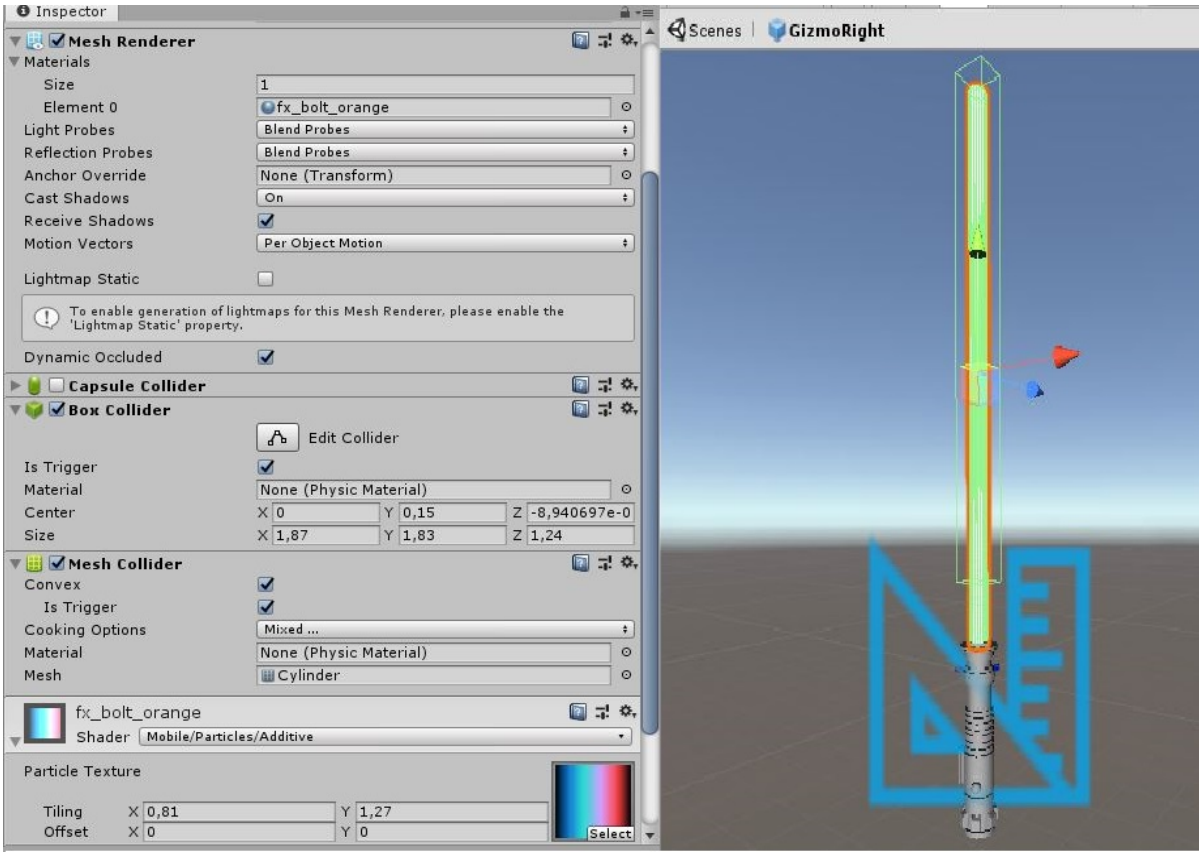



***Figura 45*** *componentes de la espada laser.*

### 5.2.2 Robot BB8

Utilizaremos un modelo 3D libre[29] sobre el que añadiremos un script con el que configurar su comportamiento (escenas "entrenamiento jedi" y "defensa jedi"). En este ejemplo utilizaremos el script llamado *movimiento* que se muestra en la figura 47 para darle un movimiento aleatorio al objeto dentro de la escena. Este script permite rotar y mover un objeto del juego en la escena, a través del método *recalcularPosicion()* obtenemos una posición aleatoria para los ejes X y Z y en el método *Update()* realizamos el movimiento del objeto y su rotación.

***Figura 46*** *Objeto 3D BB8.*

[29] https://www.highend3d.com/maya/downloads/character-rigs/c/star-wars-bb-8-rig-for-maya

```
5   public class movimiento : MonoBehaviour
6   {
7       public float velocidad = 1f;
8       public float rotacion = 1f;
9
10      Vector3 posicion;
11      Vector3 direccion;
12
13      float radio = 5f;
14
15      void recalcularPosicion()
16      {
17          posicion = transform.position + Random.insideUnitSphere * radio;
18          posicion.y = 0;  // posicion fija en el eje y
19      }
20
21      void Start()
22      {
23          recalcularPosicion();
24      }
25
26
27      void Update()
28      {
29          direccion = posicion - transform.position;
30          if (direccion.magnitude < 0.25f)
31              recalcularPosicion();
32
33          Quaternion direccionRotacion = Quaternion.LookRotation(direccion, Vector3.up);
34          transform.rotation = Quaternion.Lerp(transform.rotation, direccionRotacion, rotacion * Time.deltaTime);
35          transform.position += transform.forward * velocidad * Time.deltaTime;
36      }
37  }
```

***Figura 47*** *Script movimiento.*

### 5.2.3 Modelo caza imperial en escena de “Piloto de caza”

Otro componente importante para mejorar la inmersión en el universo Star Wars, se trata de las naves espaciales, las cuales dan un aspecto visual y sonoro de la aplicación y ayudan a introducir al jugador en el mundo virtual. Las escenas cuentan con una colección de objetos 3D de distintas naves espaciales, las cuales se van generando automáticamente alrededor del entorno del jugador.

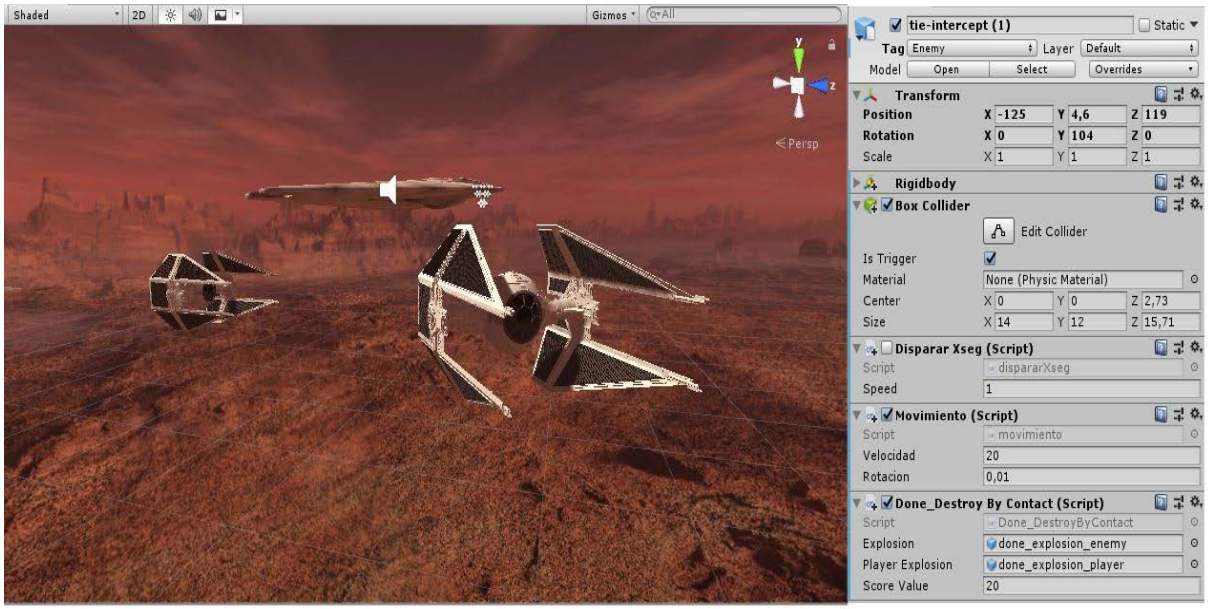



***Figura 48*** *Cazas Star Wars.*

Las naves espaciales están creadas a partir de modelos 3D[30]. Estas contienen distintos componentes Unity que condicionan su comportamiento. Las naves contienen el componente *Box collider* con la opción *Is Trigger* activada para detectar colisiones. Este componente permite que, al recibir un disparo de parte del jugador las naves serán automáticamente destruidas.

Estos objetos 3D tienen configurados dos "scripts" para definir su comportamiento en el juego. Uno de ellos para configurar su movimiento dentro de la escena y otro para destruir el objeto al ser detectado por el *Trigger*. El script *DestroyByContact* podemos verlo en las figuras 49 y 50. El método *Start()* guarda la referencia al objeto "GameController" para poder añadir la puntuación del jugador al destruir una nave enemiga.

```
public class DestroyByContact : MonoBehaviour
{
    public GameObject explosion;
    public GameObject playerExplosion;
    public int scoreValue;
    private Done_GameController gameController;

    void Start ()
    {
        GameObject gameControllerObject = GameObject.FindGameObjectWithTag ("GameController");
        if (gameControllerObject != null)
        {
            gameController = gameControllerObject.GetComponent <Done_GameController>();
        }
        if (gameController == null)
        {
            Debug.Log ("Cannot find 'GameController' script");
        }
    }
```

***Figura 49*** *Script DestroyByContact parte 1*

El método OnTriggerEnter(Collider other) de la figura 50 identificará y destruirá el objeto que detecte y a él mismo, siempre que el objeto detectado no tenga la etiqueta de "Enemy". En caso de que el objeto a destruir sea el propio jugador, se llamará al método *GameOver().*

[30] https://free3d.com/3d-model/tie-interceptor-4544.html

```
// detectar colision

void OnTriggerEnter (Collider other)
{
    if (other.tag == "Boundary" || other.tag == "Enemy")
    {
        return;
    }

    if (explosion != null)
    {
        Instantiate(explosion, transform.position, transform.rotation);
    }

    if (other.tag == "Player")
    {
        Instantiate(playerExplosion, other.transform.position, other.transform.rotation);
        gameController.GameOver();
    }

    gameController.AddScore(scoreValue);

    Destroy (gameObject);       // destruir objeto del script
}
}
```

***Figura 50*** *Script DestroyByContact parte 2.*

La nave "Player"[31] (ver figura 51) será la nave que el jugador manejará en la escena, esta nave contiene dos componentes principales: *capsule collider*, para detectar colisiones y el Script *DestroyByContact* que destruirá la nave al ser colisionada por un rayo láser enemigo.

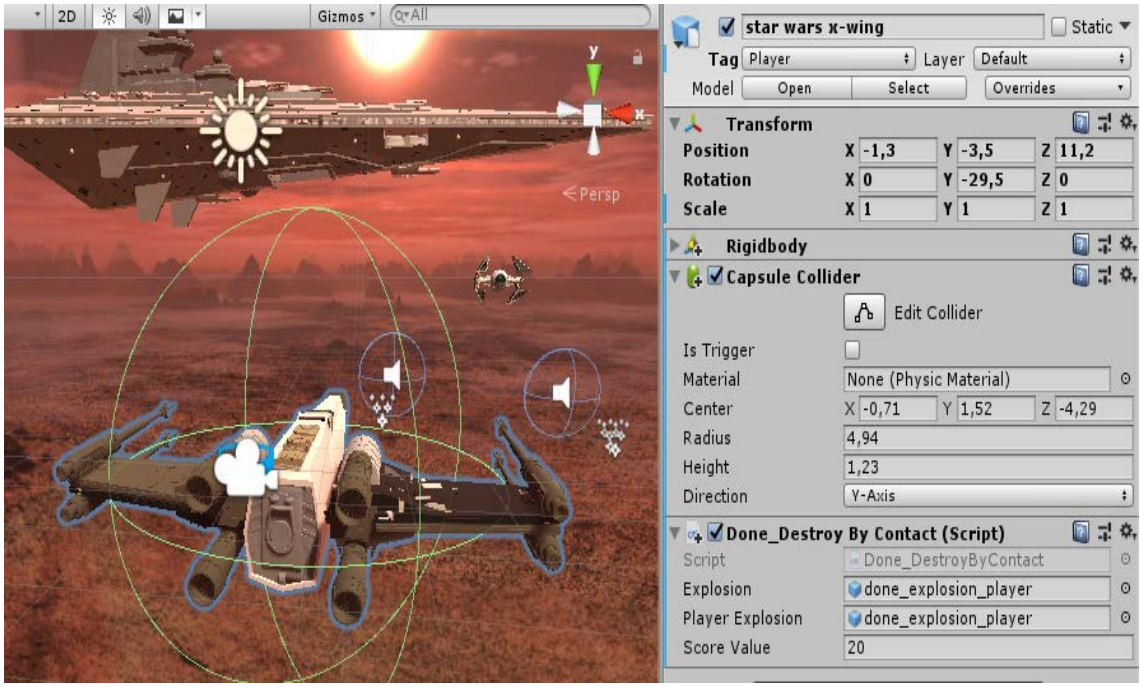



***Figura 51*** *Nave Player X-wing.*

[31] https://free3d.com/3d-model/star-wars-x-wing-23129.html

El rayo láser (figura 52) es el objeto que utilizaremos para destruir los objetos de juego (naves, asteroides o jugador), en nuestro caso en la escena “Piloto de caza” utilizaremos este rayo para destruir las naves enemigas que vayan surgiendo en el juego. En el juego aparecerán dos tipos de láser, el del jugador, que tendrá la etiqueta “*Player*”, para distinguirse de los rayos enemigos, y el láser de tipo “*Enemy*” para las naves enemigas, este último aparecerá de color rojo, distinguiéndose así del rayo del jugador, de esta forma cuando un disparo con la etiqueta “*Enemy*” alcance al jugador se ejecutará al método gameOver(), finalizando así el juego (ver figura 50).

Este objeto dispone del componente *Capsule Collider*, el cual tiene la función de detectar las colisiones que se produzcan en este objeto y tres scripts con los que modificaremos su comportamiento, el primero de ellos *Done_Mover* se utiliza para añadir la velocidad a la que se desplazará el rayo láser. El segundo, *Done_DestroyByContact,* que añadirá las explosiones y sonido al objeto destruido a la vez que el rayo se destruirá. Este script también sumará 10 puntos después de destruir el objeto. Finalmente, el script *Done_Destroy_By_time* destruirá el láser a los tres segundos de ser lanzado si este no ha colisionado con otro objeto del juego.

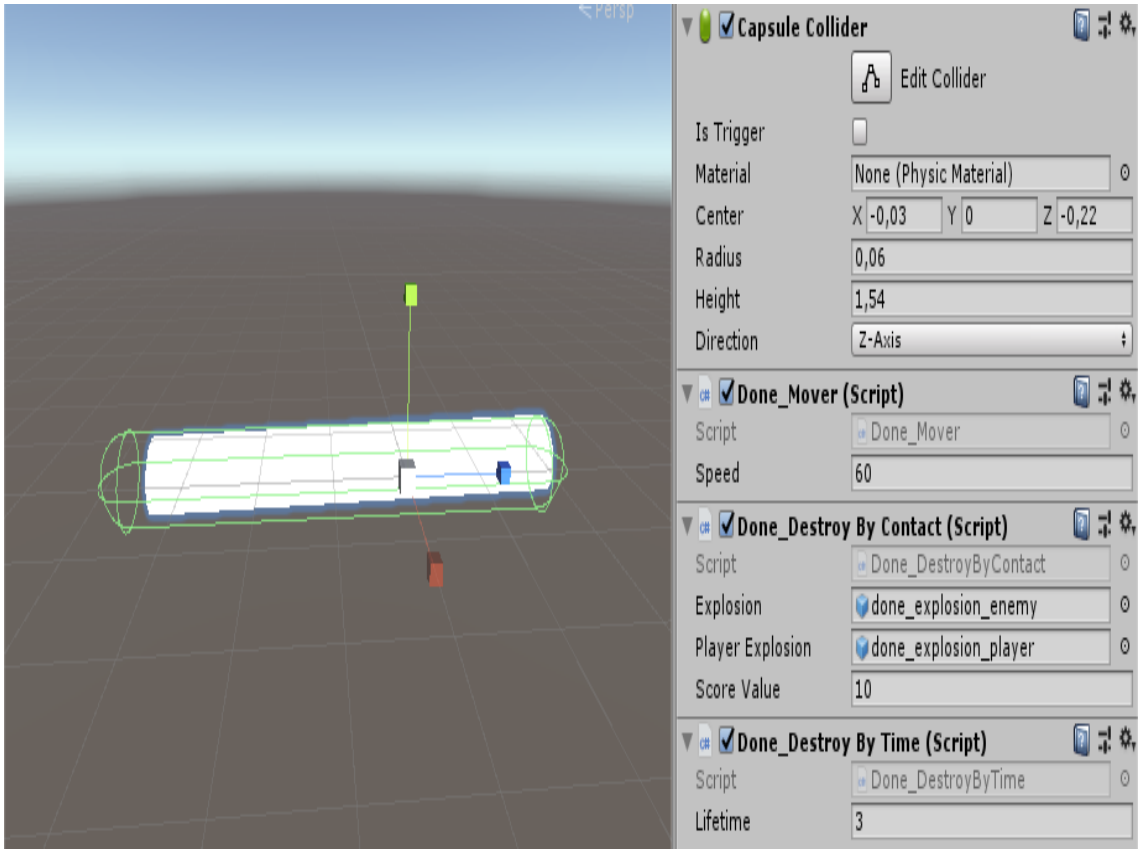



***Figura 52*** *Rayo láser.*

### 5.2.4 *Game controller*

Se trata de un componente de Unity que se añade a la escena y se encarga de definir el comportamiento del juego dentro de la escena. Este componente está diseñado para las escenas “defensa jedi” y “piloto de caza” y tiene asociado el Script “*Game Controller*” (Figura 53). Este script se ha creado de manera genérica, permitiendo usarse en las diferentes escenas del juego solo cambiando los tiempos de *spawn* y el propio objeto que se generará, como por ejemplo en la escena “Piloto de caza”, dónde se generaran naves espaciales que nos dispararán cuando nos acerquemos a la “estrella de la muerte”. En el ejemplo de la escena “defensa jedi”, se generaran asteroides que debemos ir destruyendo.

```
public class GameController : MonoBehaviour
{
    public GameObject[] hazards;
    public Vector3 spawnValues;
    public int hazardCount;
    public float spawnWait;
    public float startWait;
    public float waveWait;


    void Start()
    {

        StartCoroutine(SpawnWaves());
    }


    IEnumerator SpawnWaves()
    {
        yield return new WaitForSeconds(startWait);
        while (true)
        {
            for (int i = 0; i < hazardCount; i++)
            {
                GameObject hazard = hazards[Random.Range(0, hazards.Length)];
                Vector3 spawnPosition = new Vector3(Random.Range(-spawnValues.x, spawnValues.x), spawnValues.y, spawnValues.z);
                Quaternion spawnRotation = Quaternion.identity;
                Instantiate(hazard, spawnPosition, spawnRotation);
                yield return new WaitForSeconds(spawnWait);
            }
            yield return new WaitForSeconds(waveWait);
        }
    }
```

***Figura 53*** *Script GameController, escenas defense Jedi y piloto de caza.*

En la clase GameController (Figura 53), tenemos una serie de atributos públicos como son: un "vector" de GameObject llamado *hazards*, que serán los objetos generados (*spawn* de enemigos), un vector *spawnValues* para inicializar la posición de los objetos generados, *hazardCount* que establecerá la cantidad de enemigos generados inicialmente y los siguientes tres atributos *spawnWait*, espera entre la instanciación de los enemigos, *startWait* que establece la espera de inicio y *waveWait* que pospondrá la corrutina[32] un tiempo establecido al finalizar el bucle *for*. Al tratarse de atributos públicos podemos configurar su valor dentro de Unity, con la utilidad de darle un comportamiento distinto en cada escena del juego sin tener que modificar el código.

Dentro del script encontramos también el método: *IEnumerator SpawnWaves*() que es una corrutina encargada de gestionar el spawn de los enemigos, en ella instanciamos a los enemigos "*hazards*" con su posición en la escena y su rotación, en este caso a cero a través de la clase *Quaternion*[33]*.* Además el método establecerá un tiempo de espera entre el *spawn* de los enemigos mediante la línea *yield return new WaitForSecond(spawnWait).* La sentencia *yield* permite detener la ejecución de una corrutina durante un periodo de tiempo establecido. Para ejecutar la corrutina, se utiliza el método *StartCorrutine(SpawnWaves())*, dentro del método *start*(), el cual se ejecutara al instanciar el componente *GameObject* en la escena.

### 5.2.5 Disparador

En la escena "entrenamiento jedi", tenemos el objeto de juego encargado de dispararnos, este modelo es una esfera flotante, que nos seguirá en todo momento dentro de la escena y que al pulsar sobre el gatillo del controlador este objeto instanciara un rayo láser y un sonido de disparo hacia nosotros. El jugador deberá defenderse bloqueando los disparos con su espada. Cada vez que bloqueamos un disparo, se nos incrementará la puntuación hasta que el jugador falle el bloqueo y este sea alcanzado por el rayo láser, entonces el juego finalizará.

[32] https://docs.unity3d.com/es/current/Manual/Coroutines.html
[33] https://docs.unity3d.com/es/current/Manual/QuaternionAndEulerRotationsInUnity.html


***Figura 54*** *Esfera de la escena " entrenamiento jedi".*

El objeto disparador de la figura 55 contiene tres componentes principales: Mesh Collider, encargado de detectar las colisiones, un script encargado de hacer el disparo y una fuente de audio que añadirá el sonido del disparo cada vez que se produzca un disparo.

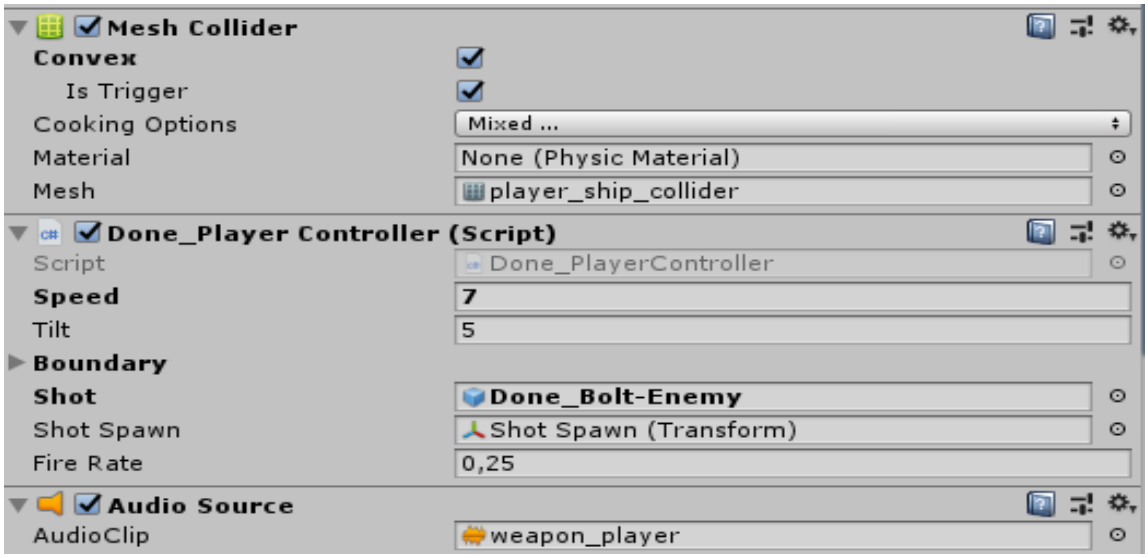



***Figura 55*** *Componentes disparador.*

En la figura 56, encontramos el script Done_PlayerController del game object disparador producirá un Done_Bolt-Enemy (un rayo láser) cada vez que pulsemos el botón 15 (gatillo) de nuestro "motion controller" reproduciendo el audio asociado al componente.

```
public GameObject shot;
public Transform shotSpawn;
public float fireRate;
private float nextFire;

void Update ()
{
    if (Input.GetKey("joystick button 15") && Time.time > nextFire)
    {
        nextFire = Time.time + fireRate;
        Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
        GetComponent<AudioSource>().Play();
    }
```

***Figura 56*** *Script PlayerController para disparar láser.*

### 5.2.6 Menú del juego

Está compuesto por un componente "*canvas"* y cuatro botones, tal como aparece en la figura 57. Los tres primeros botones son para acceder a las diferentes escenas del juego, mientras que el último botón sirve para salir de la aplicación. Cuando el menú se encuentra visible, el juego se quedará en "pausa" hasta que el jugador vuelva a reanudar el juego.

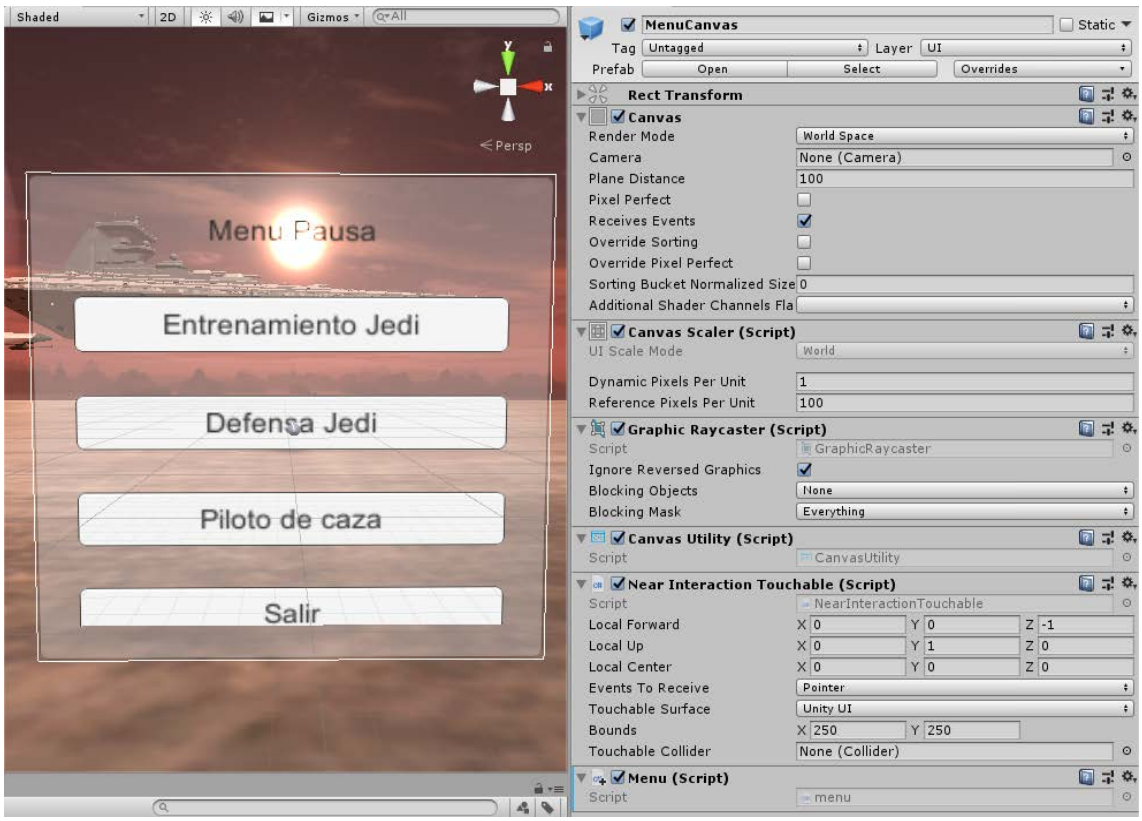



***Figura 57*** *Menú del juego.*

El componente *MenuCanvas* está asociado al script de la figura 58 *menú*, que recibe un componente canvas y lo inicializa a "false" para que se oculte al iniciar el juego. En el método *Update()* detectamos cuando el jugador pulsa sobre el botón 7 del joystick o sobre la tecla P del teclado e inicializa el método *Pause()* que habilitará o deshabilitará el menuCanvas y pausará el juego hasta que el jugador vuelva a pulsar el botón. Finalmente tenemos los métodos *salir()*, que al pulsar sobre el botón salir del menuCanvas saldremos de la aplicación y el método *cargarNivel(string nombreNivel*) que cargará la escena sobre la que pulsemos.

```
using UnityEngine;
#if UNITY_EDITOR
using UnityEditor;
#endif

public class menu : MonoBehaviour{
    Canvas canvas;

    void Start(){
        canvas=GetComponent<Canvas>();
        canvas.enabled = false;
    }

    void Update(){
        if (Input.GetKeyDown(KeyCode.P) || Input.GetKey("joystick button 7")){
            Pause();
        }
    }

    public void Pause(){
        canvas.enabled = !canvas.enabled;
        //detiene o continua el juego
        Time.timeScale = Time.timeScale == 0 ? 1 : 0;
    }

    public void salir(){
        #if UNITY_EDITOR
        EditorApplication.isPlaying = false;
        #else
        Application.Quit();
        #endif
    }

    public void CargaNivel(string nombreNivel){
        SceneManager.LoadScene(nombreNivel);
    }
}
```

***Figura 58*** *Script Menú.*

Finalmente para añadir la funcionalidad al botón del menú, debemos añadir en Unity en el evento On Click() de la figura 59, la clase, el método y el nombre de la escena a la que queremos acceder al pulsar sobre el botón del menú.

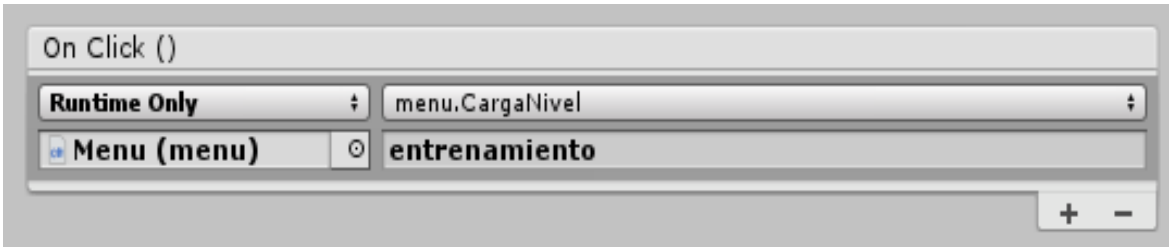



***Figura 59*** *Ejemplo de asignación de valores al evento On Click().*

### 5.2.7 Marcador

Con el fin de obtener una puntuación este script consiste en sumar los puntos producidos por el jugador en cada escena, así por ejemplo cada vez que se consigue bloquear el disparo con la espada se ganan 10 puntos o cada nave enemiga destruida se consiguen otros 10 puntos. Al terminar el juego se muestra el total de puntos en la pantalla de fin de juego, véase la Figura 60.

Otro punto a destacar es la posibilidad de añadir un gráfico de consumo de los recursos del ordenador, disponible desde el MRTK, el cual se puede activar para controlar el consumo de la aplicación.


***Figura 60*** *Ejemplo de la pantalla fin de juego.*

En el siguiente “script” (Figura 61 y 62) podemos ver el código del marcador del juego, observamos en el método *Update(),* la condición de pulsar la tecla R o el botón 5 del *joystic* para reiniciar el juego y recargar la escena al punto inicial.

```
void Update()
{
    if (restart)
    {
        if (Input.GetKeyDown(KeyCode.R) || Input.GetKey("joystick button 5"))
        {
            SceneManager.LoadSceneAsync("entrenamiento", LoadSceneMode.Single);
        }
    }

    if (gameOver)
    {
        restartText.text = "Press 'R' Restart";
        restart = true;
        return;
    }

}
```

***Figura 61*** *Script Marcador, Game Over.*

Los tres métodos de la figura 62 corresponden a añadir el marcador *AddScore(int newScore),* actualizar marcador si el jugador sigue en partida *U*pdateScore() y el método para acabar el juego *GameOver()* que mostrará el mensaje “Game Over!” y permitirá poder reiniciar el juego.

```
public void AddScore(int newScoreValue)
    {
        score += newScoreValue;
        UpdateScore();
    }

    void UpdateScore()
    {
        if(gameOver) return;
        scoreText.text = "Score: " + score;
    }

    public void GameOver()
    {
        gameOverText.text = "Game Over!";
        gameOver = true;
    }
```

***Figura 62*** *Script Marcador, actualizar marcador.*

### 5.3 Pruebas y evaluación de los prototipos

Finalmente como dijimos en el apartado 4.5 al finalizar el proceso de desarrollo debemos analizar los resultados obtenidos. En este apartado comprobaremos si la aplicación funciona correctamente según lo establecido en las fases de diseño y codificación.

Una vez creadas las escenas del juego y programado su comportamiento, se realizarán una serie de pruebas desde el entorno de ejecución de “Unity” dónde comprobaremos si nuestra aplicación realiza el comportamiento programado:

- **Prueba 1.** Escena entrenamiento jedi
  Se comprueba la funcionalidad de la escena, el jugador puede moverse e interactuar con los objetos de la escena, el marcador suma los puntos cada vez que el jugador bloquea el disparo con su espada. Además, cuando es alcanzado por un disparo aparece el mensaje de “Game Over” y el marcador no sigue sumando. Ver la figura 60.

- **Prueba 2.** Escena defensa jedi
  En esta escena comprobamos cómo el jugador puede moverse e interactuar con los objetos de la escena, comprobamos el correcto funcionamiento del marcador cada vez que se destruye un asteroide y finalmente se verifica que al ser alcanzado por un asteroide aparece el mensaje de “Game Over” y el marcador no sigue sumando. La figura 63 muestra un instante durante la ejecución de esta escena.

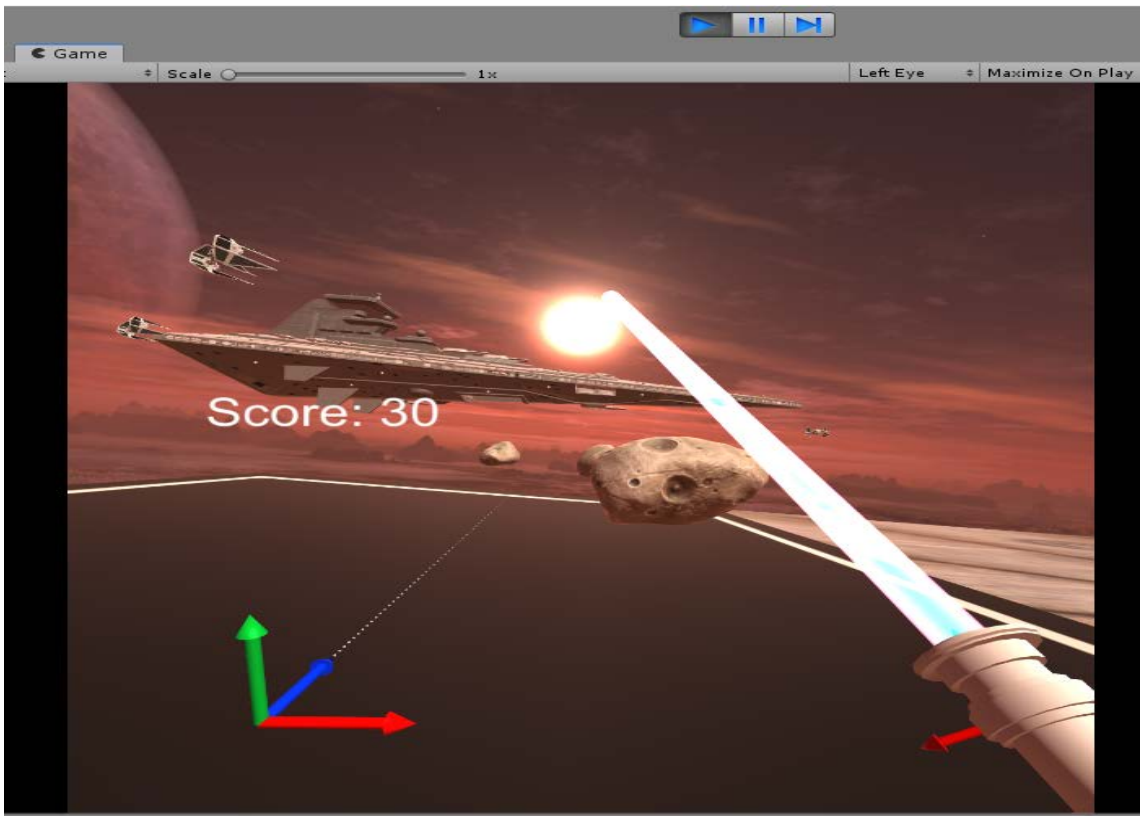



***Figura 63*** *Escena defensa jedi.*

- **Prueba 3.** Escena piloto de caza
  En la última escena comprobamos cómo podemos movernos a través de la escena, que podemos destruir las naves enemigas hasta finalmente llegar a la estrella de la muerte y destruirla, en caso contrario si un enemigo nos alcanza aparecerá el mensaje de “Game Over” y finalizará el juego.

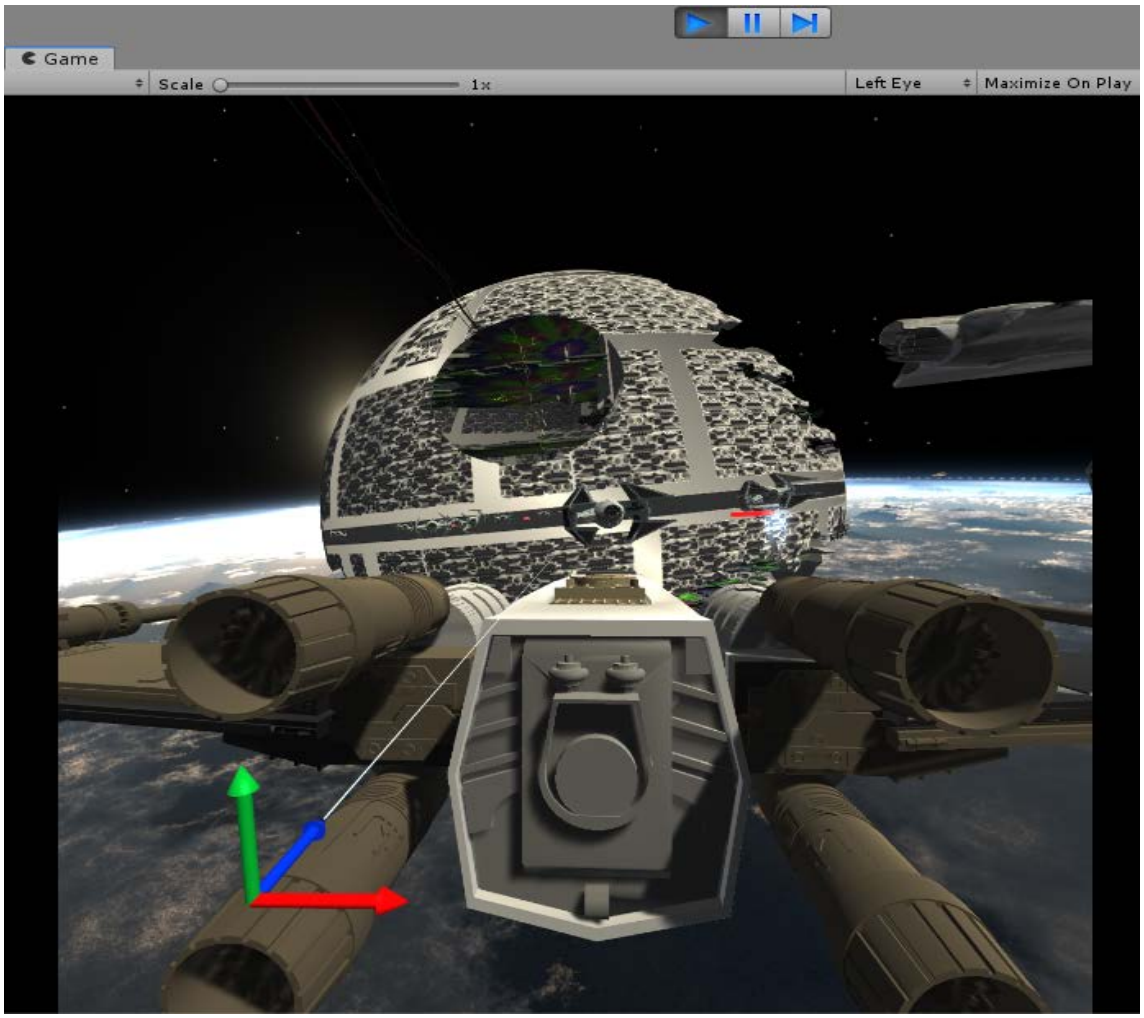



***Figura 64*** *Escena piloto de caza.*

- **Prueba 4.** Pausar y finalizar juego
  En esta prueba accedemos al menú del juego para comprobar que la escena se pausa correctamente y podemos elegir entre salir del juego o cambiar a otra escena, véase la figura 65.

  En el menú, cuando pulsamos sobre una de las escenas del juego, comprobamos que carga la nueva escena, pero, una vez dentro de la nueva escena desaparecen los mandos del juego.

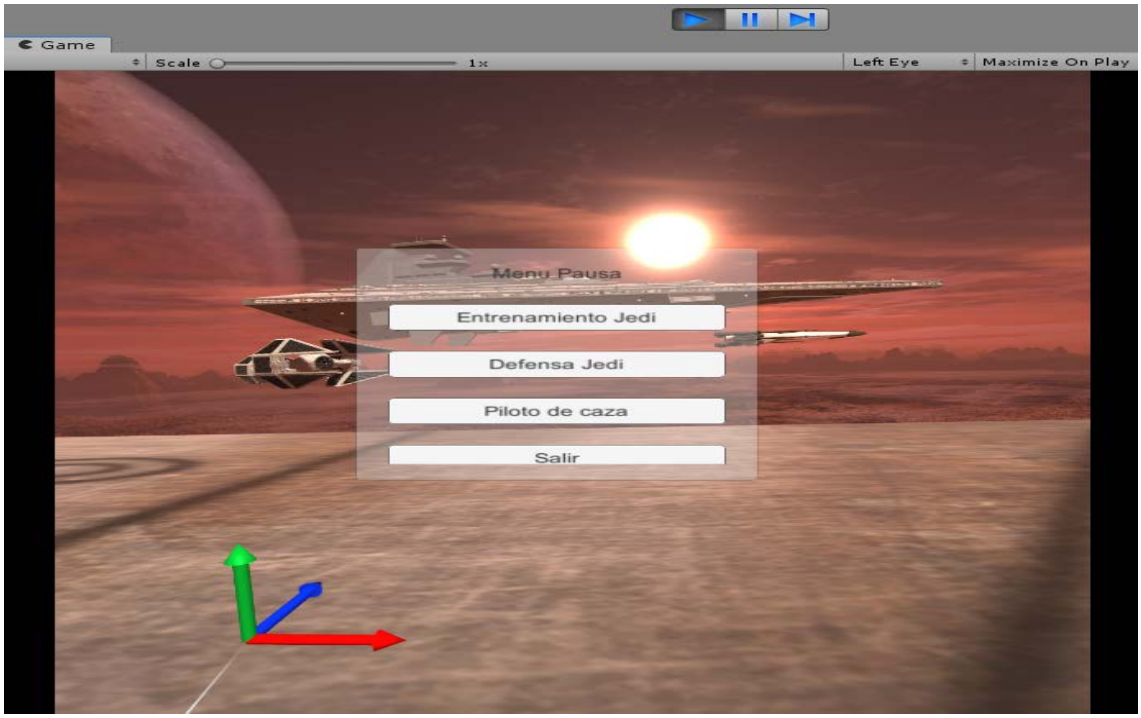



***Figura 65*** *Menú pausa, salir, selección de escenas.*

En la figura 66 podemos ver como el componente *DontDestroyOnLoad* que crea el MRTK en Unity guarda la configuración de la cámara y del controlador de movimiento para evitar que no se destruya al cambiar de escena, pero en la figura 67 después de cambiar de escena los controladores desaparecen y, aunque intentes activarlos físicamente desde los botones de encender/apagar, estos no aparecen en la escena hasta que sales y vuelves a entrar.

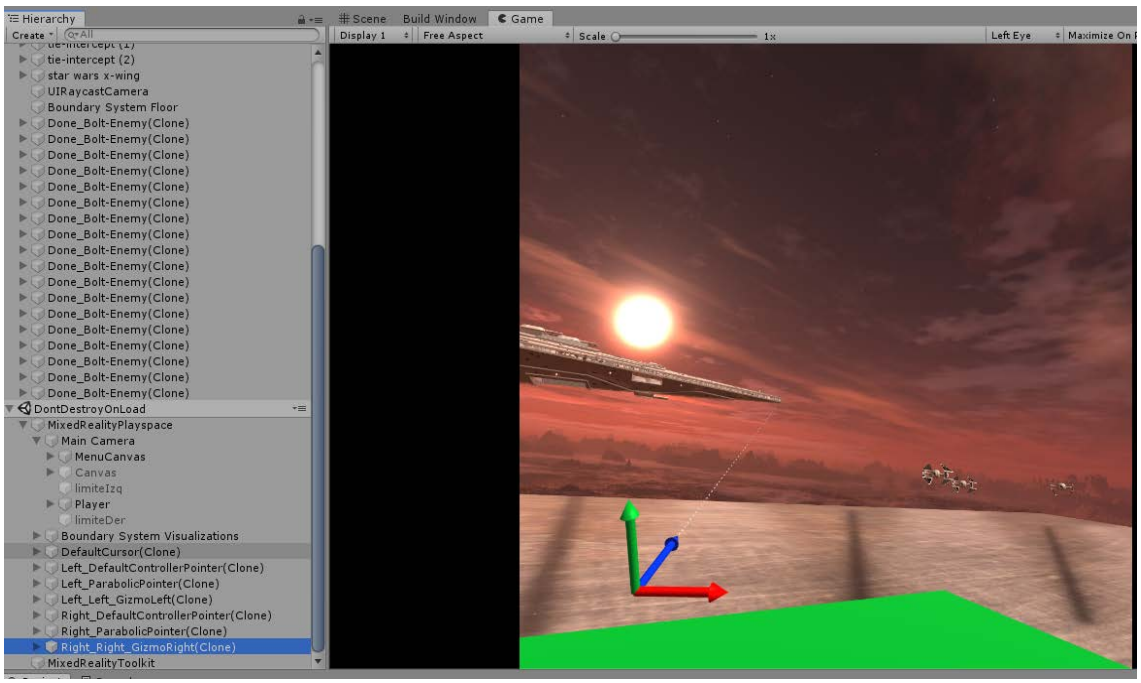



***Figura 66*** *Antes de cambiar de escena en Unity.*

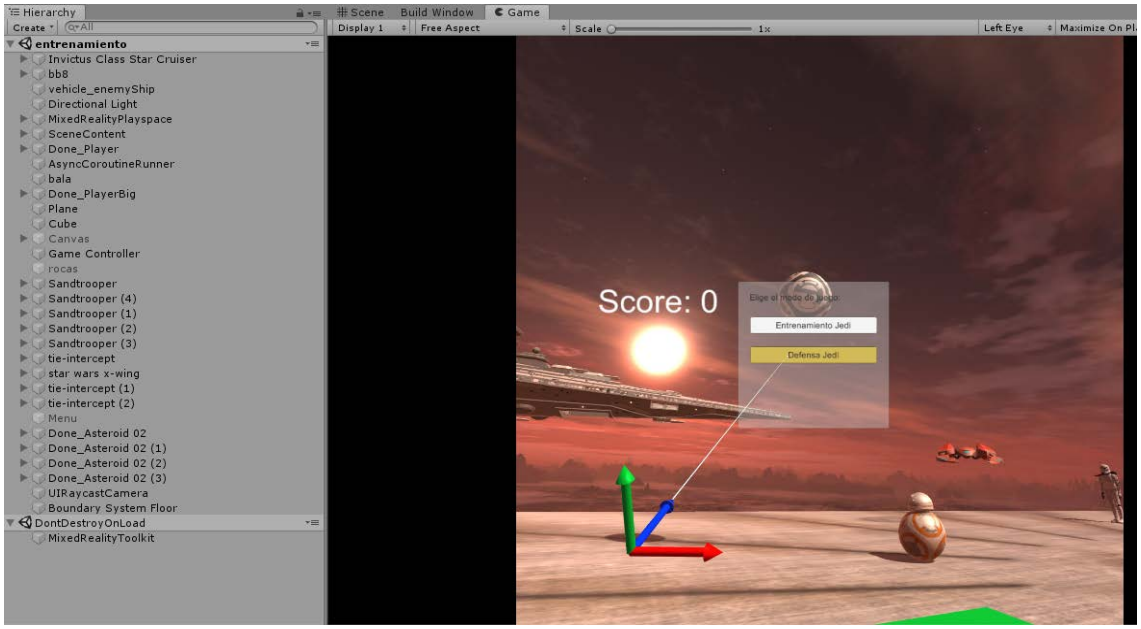
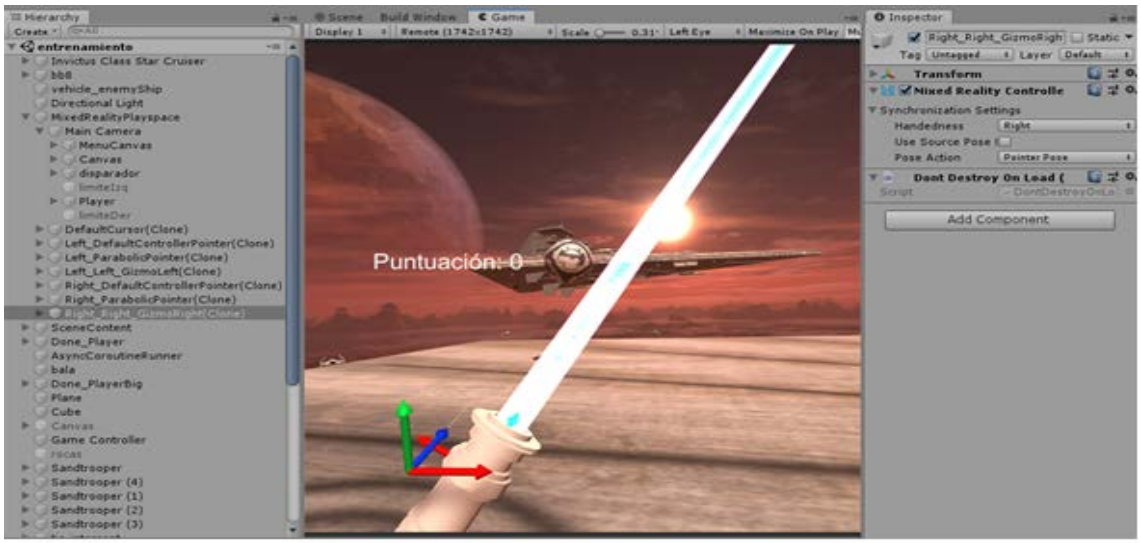



***Figura 67*** *Ejemplo de cambio de escena y como queda el componente DontDestroy.*

Visto que el MRTK para Unity no puede preservar los controladores de movimiento al hacer un cambio de escena, se intenta conservar dichos objetos manualmente a través del método *DontDestroyOnLoad*[34] de Unity, sobre los objetos que deseamos que no se destruyan al cargar una nueva escena. En la figura 68, podemos ver como añadimos el script *DontDestroyOnLoad* directamente sobre el objeto del controlador que maneja la espada láser.




```
public class DontDestroyOnLoad : MonoBehaviour
{
    void Awake()
    {
        GameObject[] objs = GameObject.FindGameObjectsWithTag("Right_Right_GizmoRight(Clone)");

        DontDestroyOnLoad(this.gameObject);
    }
}
```

***Figura 68*** *Uso del script dontDestroyOnLoad en Unity*

[34] https://docs.unity3d.com/ScriptReference/Object.DontDestroyOnLoad.html

### 5.3.1 Detección de errores

Después de analizar las pruebas realizadas podemos ver cómo en la prueba 4 cuando se cambia de escena, se pierde la referencia a los controladores en la siguiente escena. Después de buscar y analizar las posibles causas de este comportamiento, se descubre que los controladores se crean y se destruyen automáticamente por los proveedores de entrada controlados por el motor de Unity y el MRTK [31]. Para realizar proyectos en realidad mixta se debe instanciar el script *MixedRealityToolkit* dentro de la escena tal como se puede ver en la figura 66, esta instancia es la encargada de registrar, actualizar y liberar los servicios [32]. Cuando se cambia o se reinicia la escena, carga la instancia *DontdestroyOnLoad* en la que se guardan la configuración de la cámara y de los componentes MRTK, pero no de los controladores de movimiento y al pasar de escena estos no vuelven a crearse, por lo que no se puede interactuar con ellos.

Para intentar resolver este problema se ha intentado invocar a los controladores de movimiento en tiempo de ejecución a través de un script llamando al método *DontDestroyOnLoad*[35] desde una clase nueva y añadiendo los controladores, ver figura 68, pero no se ha conseguido los resultados esperados. Por esto pensamos que podría deberse a un problema en el MRTK que gestiona Unity ya que se trata de una versión aun en *release candidate*[36]. Como solución a este problema se realizan los diferentes prototipos en escenas distintas.

[35] https://docs.unity3d.com/ScriptReference/Object.DontDestroyOnLoad.html
[36] https://es.wikipedia.org/wiki/Ciclo_de_vida_del_lanzamiento_de_software#Versi%C3%B3n_candidata_a_definitiva_(RC)

# 6. Conclusiones

Con el desarrollo de este proyecto se han logrado alcanzar los objetivos marcados en el trabajo: estudiar las diferentes tecnologías de inmersión de la realidad, introducir las herramientas y entornos de desarrollo y finalmente desarrollar una aplicación en Unity para mostrar las capacidades de la realidad mixta.

En primer lugar se ha analizado la evolución de la realidad virtual, pasando por la realidad aumentada y llegando a la realidad mixta, profundizado sobre éstos últimos analizando los dispositivos existentes hoy en día.

En segundo lugar hemos creado una guía de uso para las gafas de realidad mixta Acer AH101-D8EY. En dicha guía se han visto los posibles problemas que pueden surgir durante su configuración, así como los requisitos y limitaciones del dispositivo. Una de ellas se encuentra en las cámaras, las cuales no pueden ser utilizadas para ver a través de ellas y experimentar una realidad mixta auténtica, por lo que se limitan a una realidad virtual. También hemos analizado las diferentes herramientas y aplicaciones que Microsoft proporciona para el desarrollo de la realidad mixta.

Finalmente se ha realizado el desarrollo de un videojuego en Unity, creando tres prototipos diferentes dentro de una misma aplicación con el fin de mostrar las funcionalidades de la realidad mixta y su inmersión de la realidad en un mundo virtual. En este apartado, nos hemos encontrado con ciertas limitaciones de software por parte de Microsoft y Unity al no disponer de un acceso total a los sensores del dispositivo, como es el caso de las cámaras o el sistema de vibración de los mandos.

## 6.1 Valoración personal

Después de todo el trabajo de investigación y desarrollo, puedo decir que este trabajo ha sido un reto a nivel de desarrollo debido a las dificultades de una tecnología en desarrollo. Además he tenido que aprender desde cero el entorno de desarrollo de Unity y añadir todos los SDKs necesarios para desarrollar en realidad mixta solucionando todos los problemas que iban surgiendo. Por otra parte, el lenguaje de programación de los scripts para configurar el comportamiento del juego me ha resultado más sencillo al haber trabajado anteriormente con Visual Studio y C# en la asignatura de Ingeniería del Software.

Por tanto después de todo este trabajo puedo decir que la realidad mixta de Microsoft se encuentra más cerca de la realidad virtual que de una realidad híbrida. Se trata de una tecnología que está empezando a distribuirse a los usuarios, por lo que nos encontramos con unos dispositivos con ciertas limitaciones y con una discreta tienda de aplicaciones. A pesar de esto, creo que esta tecnología se está desarrollando en la buena dirección ya que las posibilidades que ofrece son enormes.

En cuanto a las gafas de Acer nos encontramos con un dispositivo bastante completo aunque con ciertas limitaciones. Una de ellas son los controladores de movimiento, los cuales si los alejas de las cámaras del dispositivo pierden la referencia a estos afectando a su funcionamiento.

### 6.2 Trabajos futuros

En cuanto a los posibles trabajos futuros y mejoras, la aplicación desarrollada se trata de tres prototipos que sirven para demostrar las aplicaciones y el uso de unas gafas de realidad mixta, por lo que se pueden añadir nuevas funcionalidades y niveles de juego:

- En primer lugar se ha observado un fallo con los sensores de la posición de los mandos que hace que en ocasiones se deba reiniciar la aplicación. Esto es debido a que el SDK MRTK se encuentra en (*release candidate*), una versión candidata a ser una versión final con posibles fallos.

- En cuanto al juego se podría añadir distintos niveles de juego en cada una de las escenas construidas, ampliando así su duración y jugabilidad.

- Realizar un guardado de la puntuación obtenida para después mostrarla en un *ranking* de puntuación.

- Añadir un sistema de configuración y calibrado de los mandos.

- Agregar diferentes animaciones a los objetos y dotar a los enemigos de una inteligencia para mejorar la dificultad.

# 7. Bibliografía

[1] Realidad digital: el enfoque pasa de la tecnología a la oportunidad (17 de Junio de 2019). Obtenido de deloitte: https://www2.deloitte.com/insights/us/en/focus/tech-trends/2018/immersive-technologies-digital-reality.html

[2] Realidad Virtual (14 de Febrero de 2019). Obtenido de revistaeducacionvirtual: https://revistaeducacionvirtual.com/archives/2024

[3] X. Basogain, M. Olabe, K. Espinosa, C. Rouèche y J.C. Olabe. (2007) Realidad aumentada en la educación; una tecnología emergente.

[4] Paul Milgram and Fumio Kishino. (1994) A Taxonomy of Mixed Reality Visual Displays. IEICE TRANS. INF. & SYST., VOL. E77-D.

[5] Pasado y presente de la realidad virtual Universitat Oberta de Catalunya (11 de Junio de 2019). Obtenido de cv.uoc: http://cv.uoc.edu/annotation/8ebfc11d61d9fb2feed41b629265e634/463715/PID_00150738/modul_1.html

[6] Aspen Movie Map (6 de Julio de 2019). Obtenido de: wikivisually.com https://wikivisually.com/wiki/Aspen_Movie_Map

[7] Cascos realidad virtual Sega (24 de Mayo de 2019). Obtenido de segaretro: https://segaretro.org/Sega_VR

[8] Historia de la realidad aumentada. Tablero configuración Tom Caudell (14 de Julio de 2019). Obtenido de vertebrae: https://blog.vertebrae.com/history-augmented-reality-1

[9] Google Glass (4 de Junio de 2019). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Google_Glass

[10] Realidad mixta (5 de Noviembre de 2018). Obtenido de Microsoft: https://docs.microsoft.com/es-es/windows/mixed-reality/mixed-reality

[11] MRTK (6 de Junio de 2019). Obtenido de Microsoft: https://docs.microsoft.com/es-es/windows/mixed-reality/mrtk-getting-started

[12] OpenHMD (6 de Junio de 2019). Obtenido de Openhmd: http://www.openhmd.net/

[13] Build Wagon (7 de Junio de 2019). Obtenido de Buildwagon: https://www.buildwagon.com/guide.html

[14] MixCast (7 de Junio de 2019). Obtenido de Mixcast: https://mixcast.me/sdk-details

[15] Unity (24 de Febrero de 2019). Obtenido de Unity: https://unity3d.com/es/public-relations

[16] Unreal Engine (24 de Febrero de 2019). Obtenido de Unrealengine: https://www.unrealengine.com/what-is-unreal-engine-4

[17] CryEngine (24 de Febrero de 2019). Obtenido de Cryengine: https://www.cryengine.com

[18] Documentación de Unity (11 de Enero de 2019). Obtenido de Unity: https://docs.unity3d.com/es/current/Manual/UnityManual.html

[19] Dispositivos inmersivos y holográficos (13 de Noviembre de 2018). Obtenido de Unity: https://docs.unity3d.com/Manual/wmr_sdk_overview.html

[20] Precio y características dispositivos RM (21 de Noviembre de 2018). Obtenido de lifewire: https://www.lifewire.com/best-windows-mixed-reality-headsets-4173017

[21] Motion controllers (15 de Marzo de 2019). Obtenido de altvr: https://help.altvr.com/hc/en-us/articles/360002348514-Installing-Altspace-for-Windows-Mixed-Reality

[22] Instalación gafas Acer AH101-D8EY (1 de Diciembre de 2018). Obtenido de Microsoft: https://docs.microsoft.com/es-es/windows/mixed-reality/enthusiast-guide/install-windows-mixed-reality

[23] Portal de realidad mixta. (4 de Diciembre de 2018). Obtenido de Microsoft: https://www.microsoft.com/es-es/p/mixed-reality-portal/9ng1h8b3zc7m?activetab=pivot%3Aoverviewtab

[24] Problemas realidad mixta de Windows (9 de Enero de 2019). Obtenido de Microsoft: https://support.microsoft.com/es-es/help/4041252/windows-10-use-mixed-reality-portal

[25] configuración motion controllers en Unity. (3 de Junio de 2019). Obtenido de Microsoft: https://docs.microsoft.com/es-es/windows/mixed-reality/gestures-and-motion-controllers-in-unity

[26] Input motion controllers (16 de Mayo de 2019). Obtenido de Microsoft: https://docs.microsoft.com/fr-fr/windows/mixed-reality/mixed-reality-213

[27] Descargar MRTK (16 de Abril de 2019). Obtenido de github: https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/DownloadingTheMRTK.html

[28] Botones motion controllers (10 de Mayo de 2019). Obtenido de github: https://github.com/MicrosoftDocs/mixed-reality/blob/master/mixed-reality-docs/mixed-reality-213.md

[29] Orientación Headset 3/6DoF. (22 de Mayo de 2019). Obtenido de emiliusvgs: https://emiliusvgs.com/que-es-3dof-6dof/

[30] modelos 3D para Unity. (7 de Abril de 2019). Obtenido de free3d: https://free3d.com/es/modelos-3d/star-wars

[31] API Documentación MRTK (11 de Julio de 2019). Obtenido de: microsoft.github https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/Input/Controllers.html

[32] API Documentación MRTK (13 de Julio de 2019). Obtenido de: microsoft.github https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/Architecture/FrameworkAndRuntime.html

# Glosario de abreviaturas y términos

**Realidad Virtual:** Simula una nueva realidad sumergiendo al usuario en un nuevo entorno digital 3D aislándolo del mundo real.

**Realidad Aumentada:** Superpone elementos digitales creados por ordenador en un entorno real. Esta tecnología no necesita de unas gafas para poder mostrar estos elementos superpuestos.

**Realidad Mixta:** combina un entorno real con objetos digitales, Esta combinación permite al usuario interactuar tanto con elementos reales como virtuales.

**Inmersivo:** experiencia digital multisensorial llevada a cabo por dispositivos de RV, RA o RM en que el usuario se sumerge en un mundo simulado por ordenador.

**Plugin:** Complemento que se relaciona a otros elementos para crear funcionalidades o comportamientos nuevos.

**Prefab:** actúa como una plantilla, almacenando un objeto con sus propiedades permitiendo crear nuevas instancias de un objeto en la escena.

**Código abierto:** Es un modelo de desarrollo de software basado en la colaboración abierta. El código fuente publicado bajo esta licencia permite su modificación y distribución de forma libre.

**API:** interfaz de programación de aplicaciones, es un conjunto de herramientas que ofrece acceso a funciones de un determinado software permitiendo que un servicio se comunique con otro servicio sin necesidad de saber cómo se implementan dichas funciones.

**SDK**: un kit de desarrollo de software es un conjunto de herramientas de desarrollo de software que le permiten al programador o desarrollador de software crear una aplicación informática para un sistema concreto.

**Unity:** Motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, OS X y Linux.

**UWP**: plataforma universal de Windows, es una plataforma común de aplicaciones presentes en todos los dispositivos que cuentan con Windows 10 y sus variantes.