



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



Escola Tècnica  
Superior d'Enginyeria  
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica  
Universitat Politècnica de València

# Guía de desarrollo de un videojuego con Unity en realidad virtual

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor:** Víctor Jarilla Romo

**Tutor:** Javier Lluch Crespo

2017-2018



# Resumen

---

La realidad virtual es una tecnología en crecimiento y actualmente se encuentra en etapas de evolución tempranas debido a su auge como plataforma de desarrollo sobre el año 2016.

Con este proyecto se crea una guía de iniciación para desarrolladores con la tecnología de realidad virtual. El desarrollo que se realiza durante la guía se hace uso del motor de videojuegos *Unity3D* y programación en C# junto al uso de las librerías *Google VR* que proporciona Google, estas librerías proporcionan las herramientas necesarias para realizar el desarrollo de un proyecto en realidad virtual completo.

Android es el sistema sobre el que se desarrolla el proyecto con la unión de *CardBoard* y *Daydream* que forman *Google VR* dando funcionalidades de ambas plataformas que confluyen en una demostración de videojuego.

**Palabras clave:** *Unity, Android, videojuego, realidad virtual, CardBoard, Daydream.*

# Resum

---

La realitat virtual és una tecnologia en creixement i actualment es troba en etapes d'evolució primerenques a causa del seu auge com a plataforma de desenvolupament sobre l'any 2016.

Amb este projecte es crea una guia d'iniciació per a desenvolupadors amb la tecnologia de realitat virtual. El desenvolupament que es realitza durant la guia es fa ús del motor de videojocs *Unity3D* i programació en C# junt amb l'ús de les llibreries *Google VR* que proporciona Google, estes llibreries proporcionen les ferramentes necessàries per a realitzar el desenvolupament d'un projecte en realitat virtual complet.

Android és el sistema sobre el qual es desenvolupa el projecte amb la unió de *CardBoard* i *Daydream* que formen *Google VR* donant funcionalitats d'ambdós plataformes que conflueixen en una demostració de videojoc.

**Paraules clau:** *Unity, Android, videojoc, realitat virtual, CardBoard, Daydream.*

# Abstract

---

Virtual reality is a growing technology and is currently in early stages of evolution due to its boom as a development platform on 2016.

With this project, an initiation guide for developers with virtual reality technology is created. The development carried out during the guide is made using the Unity3D video game engine and programming in C# along with the use of the *Google VR* libraries provided by Google, these libraries provide the necessary tools to carry out the development of a complete virtual reality project.

Android is the system on which the project is developed with the union of *CardBoard* and *Daydream* that form *Google VR* giving functionalities of both platforms that come together in a videogame demonstration.

**Keywords:** *Unity, Android, videogame, virtual reality, CardBoard, Daydream.*

# Tabla de contenidos

---

1. Introducción .....	9
2. Objetivos .....	11
3. Estado del arte .....	12
3.1. Crítica al estado del arte. ....	14
3.2. Propuesta.....	15
4. Análisis del problema. ....	17
4.1. Motor gráfico .....	17
4.2. Lenguaje de programación .....	18
4.3. Plataforma de programación .....	18
4.4. Plataforma Android .....	19
4.5. Google VR.....	20
4.6. Controlador .....	20
4.7. Servicios RV de Google.....	21
4.8. Soluciones alternativas .....	22
4.8.1. Posibles plataformas de realidad virtual .....	22
4.8.2. Motor de desarrollo .....	26
4.8.3. Controlador de realidad virtual .....	27
4.8.4. Solución .....	28
5. Diseño de la solución .....	29
5.1. Diseño Detallado.....	29
5.2. Tecnología utilizada .....	32
6. Desarrollo de la solución .....	33
7. Puesta a punto y Pruebas .....	37
8. Conclusiones .....	43
9. Trabajos futuros.....	45
10. Referencias.....	47
11. Anexo .....	49
[Anexo 1] Guía de desarrollo de un videojuego en realidad virtual: .....	49





# Índice de imágenes

---

Imagen 1: Sensorama. Fuente: Wikipedia.org.....	12
Imagen 2: Virtual Boy Fuente: theoldcomputer.com.....	13
Imagen 3: VFX-1 Fuente: Wikipedia.org.....	13
Imagen 4: Daydream Fuente: vr.google.com.....	14
Imagen 5: Logotipo Motor videojuegos Unity. Fuente: 3dgep.com.....	17
Imagen 6: Entorno de trabajo Unity .....	18
Imagen 7: Logotipo Visual Studio.....	19
Imagen 8: Sistema operativo Android .....	19
Imagen 9: Cardboard. Fuente: beartai.com.....	20
Imagen 10: The Controller Emulator. Fuente: uploadvr.com.....	21
Imagen 11: Servicios RV de Google .....	21
Imagen 12: Oculus Rift. Fuente: overclockers.co.uk.....	22
Imagen 13: Controladores Touch Oculus Rift. Fuente: cnet.com .....	23
Imagen 14: HTC Vive. Fuente: talkandroid.com.....	24
Imagen 15: Controlador Daydream. Fuente: vr.google.com .....	25
Imagen 16: Samsung Gear VR. Fuente: theverge.com .....	26
Imagen 17 PlayStation VR. Fuente: blog.eu.playtation.com .....	26
Imagen 19 Logotipo Unreal Engine 4.....	27
Imagen 18 Logotipo CryEngine. ....	27
Imagen 20: Flujo básico del objetivo del TFG.....	29
Imagen 21: Flujo de desarrollo con distintas tecnologías.....	30
Imagen 22: Retícula en el campo de visión del usuario .....	30
Imagen 23: Diseño base de movimiento del usuario .....	31
Imagen 24: HUD básico del usuario en la aplicación .....	31
Imagen 25: Vista básica de la posición de los cuadrados de movimiento .....	33
Imagen 26: Ejemplo de asociación de script creado con valores de entrada .....	35
Imagen 27: Pantalla Ajustes de un sistema Android .....	37
Imagen 28: Pantalla menú Pantalla de bloqueo y seguridad .....	37
Imagen 29: Procedimiento de instalación .....	38
Imagen 30: Aplicación del dispositivo .....	38
Imagen 31: Inicio de la instalación .....	38
Imagen 32: Instalación efectuada correctamente.....	38
Imagen 33: Visualización desarrollo final.....	39

Imagen 34: Icono aplicación The Controller Emulator .....	39
Imagen 35: Menú principal de la aplicación de realidad virtual. ....	40
Imagen 36: Interfaz The Controller Emulator .....	40
Imagen 37: Vista movimiento de cabeza en aplicación.....	41
Imagen 38: sistema de disparos con bala y puntuación.....	41
Imagen 39: Movimiento mirando las plataformas del suelo. ....	41



# 1. Introducción

---

En la industria del videojuego la innovación, creatividad y evolución son factores clave para un continuo existo dentro de este sector. Continuamente tanto grandes empresas como pequeños desarrolladores intentan hacerse hueco en el mundo del videojuego. Para ello deben de presentar al usuario un producto diferente y atractivo que haga que el consumidor quiera el producto.

Uno de los últimos saltos en dicho sector ha sido la realidad virtual (RV), la cual mediante unas gafas/cascos de RV se intenta crear una inmersión del usuario en un entorno totalmente diferente al real. Algunas grandes compañías han sacado su propia versión de las gafas de realidad virtual, como Sony con su videoconsola PlayStation 4 y sus PlayStation VR, HTC con HTC Vive o Facebook con Oculus Rift.

Por otro lado, el sector tecnológico evoluciona de forma continua, en concreto el sector de los *smartphones* ha crecido de forma exponencial con más de 5 mil millones de líneas telefónicas en todo el mundo. Este sector se consolida con un importante hueco en el sector del videojuego imponiéndose en los primeros puestos con unos ingresos de unos 60 mil millones de dólares durante el 2017.

Con el aumento del sector del videojuego, el desarrollo de éstos es cada vez más popular, es por ello que continuamente surjan más personas interesadas en aprender y conseguir desarrollar proyectos.

Actualmente la RV en una tecnología en auge, con un proceso de maduración que tiene que conseguir en un corto plazo si quiere establecerse en el sector y no ser una moda pasajera. Hoy en día no es asequible para el gran público adquirir un equipo especializado en RV para el ocio, ya que el rango puede oscilar desde unos 250€ hasta unos 1000€ aproximadamente solo en la adquisición del dispositivo de RV sin contar con la respectiva plataforma de como una videoconsola o un ordenador.

Es por esto por lo que el sector móvil junto a la RV puede ser un paso inicial para conocer la tecnología e introducirse de una forma cómoda en el mundo del videojuego sin suponer una gran inversión en *hardware* ya que Google en 2014 introdujo las *Google CardBoard* o *CartonGlass*, las cuales son unas gafas de RV hechas de cartón y dos lentes en las cuales insertas tu smartphone y poder disfrutar de una experiencia en realidad virtual.

En este proyecto se muestra una introducción sobre las primeras nociones en el campo del desarrollo de videojuegos en realidad virtual utilizando como sistema de ejecución un *smartphone*. Todo ello desarrollado en el motor *Unity3D*, el cual como se explica más adelante proporciona de una manera más intuitiva y fácil la construcción del videojuego.

En el siguiente capítulo se plantean los pilares fundamentales sobre los que sustenta el trabajo, para confluir en el objetivo final del proyecto de realidad virtual como es la guía de desarrollo.



A continuación, en el tercer capítulo, se recorre la historia de la realidad virtual desde sus inicios hasta su actualidad viendo la evolución y cambios que ha sufrido. Seguidamente se compara a otro trabajo realizado como trabajo de final de grado identificando el aspecto no abordado que justifica este proyecto.

En el cuarto capítulo se estudian las herramientas utilizadas durante el desarrollo viendo sus ventajas y desventajas y justificando el uso de éstas frente a otro tipo de herramientas alternativas, las cuales también se estudian para concluir en una única solución final en la que consiste el trabajo.

Entrando en el quinto capítulo se detalla el diseño de la solución, así como la tecnología utilizada para el desarrollo.

Continuando con el sexto capítulo se explica parte del desarrollo del proyecto con ejemplos de código explicado y funcionalidades que el trabajo aborda.

La puesta a punto y preparación del proyecto es el contenido del séptimo capítulo donde se detalla los pasos a seguir para la ejecución del proyecto de realidad virtual, así como muestras de su funcionamiento.

En los capítulos ocho y nueve se plantean las conclusiones a las que ha llevado este trabajo, así como los posibles caminos que se pueden tomar para futuros trabajos sobre el desarrollo de la realidad virtual.

En el apartado de referencias se encuentra toda la información que se usó para la redacción de este trabajo, así como enlaces para su comprobación.

Finalmente, en el anexo encontraremos la guía de desarrollo creada para este trabajo de fin de grado con el desarrollo e instalación de las herramientas necesarias.

Esta guía es independiente y creada para desarrolladores con intenciones de adentrarse en el mundo de la realidad virtual.

## 2. Objetivos

---

El presente proyecto se fundamenta en dos pilares principales para el desarrollo de un videojuego en el motor gráfico *Unity3D*. Principalmente se centra en explicar y dar al lector una idea inicial de los conocimientos de realidad virtual, todo esto con ayuda de la herramienta que nos proporciona Google con *GoogleVR* para *Unity3D* en dispositivos móviles con un sistema operativo Android.

El otro objetivo de este proyecto es presentar al usuario una guía inicial sobre el desarrollo de un videojuego en RV sin tener que hacer una inversión en equipamiento y tecnología de realidad virtual. En esta guía se introducen las herramientas utilizadas, su instalación y uso de éstas con ejemplos prácticos que confluirán en una pequeña demostración del videojuego.

La guía está pensada para aquellos usuarios con unos conocimientos básicos en programación, más concretamente en el lenguaje de programación C#, y uso del motor de videojuegos *Unity3D*.

Como resultado final del proyecto y con los dos pilares anteriormente expuestos el objetivo final es dar al usuario unos conceptos básicos sobre la realidad virtual que permitan seguir con su aprendizaje y desarrollo en esta tecnología.

### 3. Estado del arte

---

La realidad virtual alberga un conjunto de tecnologías que simulan un entorno real o imaginario de forma virtual. La simulación de este entorno se realiza a través de unas gafas de realidad virtual que generan imágenes y sonidos que ayudan a profundizar en el entorno. El usuario de esta tecnología puede ser capaz de mirar alrededor del entorno e incluso interactuar con él mismo.

Originalmente no se ha concretado el origen de la realidad virtual, sobre la década de los 50 [1] se plantearon los primeros conceptos a nivel teórico. En 1962, Morton Heilig realizo un dispositivo llamado *Sensorama* que reproducía 5 películas que estimulaban los sentidos del oído, olfato vista y tacto (imagen 1).

En esa misma época, Douglass Engelbart utilizó pantallas de computadores como entrada y salida de dispositivos. En 1968, Ivan Sutherland diseñó un primer modelo de casco de realidad virtual y realidad aumentada. No obstante, el realismo conseguido con este dispositivo era bastante limitado debido a que la generación de los gráficos lo componían únicamente formas geométricas.



Imagen 1: Sensorama. Fuente: Wikipedia.org

Sobre la década de 1970 y 1980, el MIT (Instituto Tecnológico de Massachusetts) creó el *Aspen Movie Map*, la cual permitía recorrer las calles de la ciudad de Aspen, Colorado. Aunque las dos primeras versiones recreaban las calles mediante fotografías de la ciudad, una tercera versión ya generaba los gráficos por computador. En 1982 la compañía Atari fundó un estudio de realidad virtual, aunque debido a crisis de los videojuegos en 1983 [2] el estudio tuvo que cerrar. No obstante, algunos empleados continuaron con la investigación de esta tecnología.

Otro avance en esta tecnología se dio en 1991 con la invención de la primera habitación inmersiva, *The Cave* [3]. Esta habitación está compuesta por diversas imágenes proyectadas que permitían al usuario verse en relación con otros dentro de la misma habitación.

Ese mismo año Sega sacó al mercado las Sega VR Headset para máquinas recreativas [4] y la consola MegaDrive. Este dispositivo se componía de pantallas LCD, sonido estéreo y sensores para la detección del movimiento de cabeza. Por otro lado, la compañía Virtuality [5] sacó el sistema de realidad virtual multijugador que se componía de guantes y cascos más diversos usuarios con un coste total del dispositivo de \$73000.

En 1995 la empresa Nintendo lanzó *Virtual Boy* (imagen 2) que, aunque no se considera un dispositivo de realidad virtual como tal, era capaz de mostrar efectos 3D estereoscópicos mediante un efecto de paralaje. Ese mismo año se lanzó otro dispositivo de realidad virtual llamado VFX-1 (figura 3), su funcionamiento iba ligado a un PC y tenía compatibilidad con diferentes videojuegos como *System Shock* o *Quake*.



Imagen 2: *Virtual Boy* Fuente: [theoldcomputer.com](http://theoldcomputer.com)

Sobre el año 2001 se crea SAS3 o SAS Cube, la primera habitación inmersiva que funciona junto un computador. En 2010 se diseña el primer prototipo de las *Oculus Rift* por Palmer Luckey, las cuales ya proporcionaban un campo de visión de 90°. Pocos años después Valve sacó en 2014 un prototipo de *StemSight*, un dispositivo que separaba dos pantallas, una para cada ojo, con una resolución de 1K y lentes de Fresnel, las cuales proporcionaban una mayor apertura y corta distancia focal.



Imagen 3: *VFX-1* Fuente: [Wikipedia.org](http://Wikipedia.org)

Ese mismo año Facebook compro *Oculus VR* y Sony anunciaba las PlayStation VR para su videoconsola PlayStation 4. Por otro lado, Google anunciaba *CardBoard*, un soporte de realidad virtual que puede fabricarse con cartón y dar soporte a dispositivos móviles para la visualización de contenido de realidad virtual.

En 2015, HTC y Valve anuncian HTC Vive, una serie de dispositivos compuestos por casco, mandos y sensores que registran el movimiento del usuario por una habitación mediante infrarrojos.

En noviembre de 2016, Google lanza al mercado otra plataforma llamada *Daydream* (figura 4), que, aunque hace uso del smartphone como plataforma de ejecución requiere de unas gafas específicas que viene con un controlador que ayuda a la inmersión e interacción con el entorno virtual.

Actualmente numerosas compañías siguen investigando y desarrollando productos



Imagen 4: Daydream Fuente: vr.google.com

relacionados con la realidad virtual como pueden ser Google, Apple, Microsoft o Samsung. Ciertos aspectos de esta tecnología aún están por mejorar como una mayor y natural interacción con el entorno virtual o una mayor resolución de imagen.

Esta tecnología no es asequible y accesible a todos los usuarios, los últimos desarrollos como HTC Vive o Oculus Rift pueden rondar desde los 500€ hasta los 750€, además de tener un PC potente capaz de mover dicha tecnología.

Es por eso por lo que un primer paso para poder ver y experimentar de forma básica la realidad virtual sería *CardBoard* ya que no necesita más que un dispositivo móvil y un soporte de realidad virtual fabricado con cartón en su forma más básica.

### 3.1. Crítica al estado del arte.

Analizando distintos trabajos del pasado que abarcasen el tema de desarrollo sobre la realidad virtual se ha podido observar cómo se entra de forma directa en esta tecnología sin tener en cuenta un aspecto importante como puede ser el económico a la hora de desarrollar un videojuego o aplicación para distintas plataformas u otro aspecto como la iniciación en dicha tecnología para unos usuarios más noveles en este campo.

En el trabajo de fin de grado *Desarrollo de una aplicación de realidad virtual* por Pablo Marcos Miñarro, se crea y desarrolla una aplicación que permite al usuario moverse por el campus universitario. Toda esta aplicación se desarrolla mediante Oculus Rift y con el control Leap Motion.

Aunque el trabajo es una muestra excelente de desarrollo, la inversión económica necesaria para poder desarrollar este tipo de aplicación sería demasiado elevada para la iniciación en este tipo de tecnología de realidad virtual, además de una complejidad alta para alguien que quiera experimentar y entrar en la creación de aplicaciones de realidad virtual.

### **3.2. Propuesta**

Con este trabajo se pretende proporcionar las bases e iniciar de forma básica y adecuada a un nuevo desarrollador en la tecnología de realidad virtual sin tener que realizar una inversión económica grande, ya que como plataforma donde ejecutar la aplicación sería un *smartphone* con sistema operativo Android 7.0 y otro *smartphone* con un sistema Android 4.4 o superior uniendo la interacción básica con el entorno y el movimiento mediante *Cardboard* y *Daydream*.





## 4. Análisis del problema.

---

Como el objetivo es la iniciación a la tecnología de realidad virtual mediante un videojuego a un nuevo usuario, analizamos a continuación las distintas herramientas y tecnologías elegidas para hacer una introducción a la realidad virtual de una forma fácil y sencilla.

### 4.1. Motor gráfico

Como elección de desarrollo para nuestro proyecto de videojuego se ha elegido Unity3D (imagen 5). Este motor de videojuegos proporciona un conjunto de componentes y herramientas comunes como simulaciones de físicas o efectos de iluminación entre otros aspectos. Esto ayuda a la implementación debido a que no requiere de una inversión inicial en preparar estos aspectos para el desarrollo del videojuego.



*Imagen 5: Logotipo Motor videojuegos Unity. Fuente: 3dgep.com*

Unity dispone de un potente diseñador de niveles y una amplia documentación [6], y gracias a la continua aportación por parte de la comunidad de usuarios que crean contenido para el motor se dispone de una cuantiosa cantidad de recursos que pueden utilizarse libremente en nuestros proyectos.

Otro gran aspecto por el cual se ha elegido este motor es la facilidad de cambio de plataformas entre PC, Android, IOS, etc. [7] donde con un simple *click* el proyecto se adapta para funcionar en diversas plataformas.

Una ventaja frente a otros motores es la facilidad de observar los cambios realizados en escena mediante su editor visual ya que permite hacer cambios y probarlos de una forma fácil (imagen 6).

Además, el uso de archivos *scripts* que contienen las acciones requeridas por el programador, hacen de este motor una opción muy atractiva para los programadores con experiencia en desarrollo con lenguajes de programación orientados a objetos.

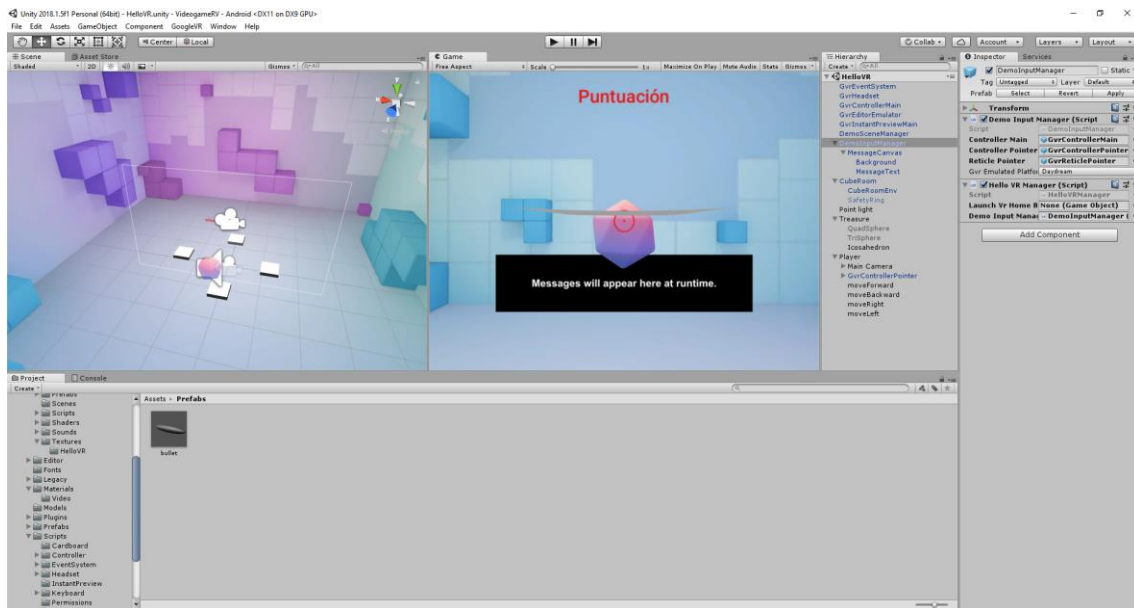


Imagen 6: Entorno de trabajo Unity

Por último, destacar que Unity posee una versión gratuita donde podemos desarrollar el proyecto y desde la versión Unity 5 este motor adquirió importantes mejoras en cuanto a calidad y realismo frente a sus competidores. Se introdujeron mejoras en iluminación en tiempo real y utilización de *shaders* (Programa informático para realizar cálculos gráficos en un lenguaje de sombreado) entre otras mejoras.

## 4.2. Lenguaje de programación

Al elegir como programa de desarrollo para el videojuego Unity, el lenguaje más adecuado para esta ocasión será C-Sharp (C#). Es un lenguaje orientado a objetos y desarrollado por Microsoft que deriva que C y C++ y hace uso del formato de objetos de la plataforma .NET (Framework de Microsoft), es un lenguaje extendido y muy utilizado en este tipo de aplicaciones. Además, es nativo de la aplicación Unity.

## 4.3. Plataforma de programación

Como plataforma de programación donde desarrollar el proyecto se ha optado por *Visual Studio Code* (imagen 7). Este editor de código fuente comercializado por Microsoft para los sistemas operativos *Microsoft*, *Linux* y *MacOS* es compatible con varios lenguajes de programación y proporciona modificar los archivos con ciertas características como:

- Depuración de código
- Resaltado de sintaxis.
- Predicción inteligente de código.



*Imagen 7: Logotipo VisualStudio*

Un punto por destacar para la elección de este editor es que es gratuito y de código abierto (*open source*) además de su versatilidad y fácil funcionamiento.

#### **4.4. Plataforma Android**

Android (imagen 8) es un sistema operativo de código abierto basado en Linux y desarrollado por Google. Es uno de los sistemas operativos más extendidos y usados en dispositivos móviles y Unity permite el desarrollo de aplicaciones nativas para estos



*Imagen 8: Sistema operativo Android*

dispositivos. Además, se puede hacer uso de la realidad virtual mediante *CardBoard*.

Estas características hacen que sea el sistema operativo idóneo para el desarrollo del proyecto de realidad virtual.

## 4.5. Google VR

Google VR son unas librerías lanzadas por la empresa de Google de libre acceso desde las cuales se pueden realizar unos primeros pasos para la experimentación con la realidad virtual [8].

Inicialmente estas librerías albergaban *CardBoard* (imagen 9), con las cuales se podían realizar una limitada interacción e inmersión en el mundo de la realidad virtual. No obstante, su factor económico es muy ventajoso debido a que podemos probar dicha tecnología desde nuestra casa con un smartphone con Android a partir de la versión 4.4 en adelante.



Imagen 9: Cardboard. Fuente: beartai.com

Por otra parte la implantación de *Daydream* en 2016 hizo que la experiencia con realidad virtual con un *smartphone* se acerque un paso más a una autenticidad e interacción más adecuada. No obstante, este requiere de la adquisición de sus propias gafas de realidad virtual *Daydream View*.

Para este desarrollo se ha elegido simular parcialmente el mando de *Daydream* junto a *Cardboard* para fusionar e intentar conseguir una experiencia lo más auténtica posible sin la necesidad de otro dispositivo de realidad virtual.

## 4.6. Controlador

Para la interacción con el entorno de nuestro sistema de realidad virtual se ha obtenido por recrear una simulación del mando de *Daydream* utilizando la aplicación *The Controller Emulator* (imagen 10) de Google [9].

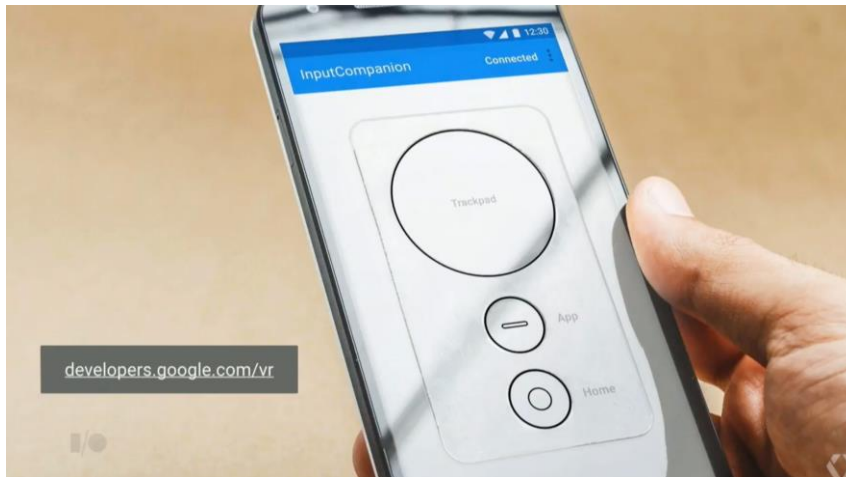


Imagen 10: The Controller Emulator. Fuente: uploadvr.com

La instalación de esta aplicación se ha realizado en otro dispositivo Android. Este dispositivo puede tener una versión de Android inferior, 4.4 o superior, al de nuestro principal smartphone, 7.0 o superior.

La conexión entre dispositivos se hace de forma automática mediante *bluetooth* lo que nos proporciona una forma fácil y sencilla de simular el mando original.

## 4.7. Servicios RV de Google

Debido a la elección de un sistema operativo Android con una versión 7.0 o superior, un desarrollo en la plataforma Unity y la utilización parcial de librerías de *Daydream* el uso de esta aplicación nos permitirá ejecutar de forma inmediata en nuestro smartphone el desarrollo que hagamos en el motor de videojuegos con la instalación de *Servicios RV de Google* (imagen 11)



Imagen 11: Servicios RV de Google

## 4.8. Soluciones alternativas

Para la realización de este proyecto se han podido optar por distintas alternativas en varios campos como la elección de la plataforma donde realizar el desarrollo de realidad virtual, el motor de desarrollo, editor de código o el controlador para la interacción.

### 4.8.1. Posibles plataformas de realidad virtual

Existen varias plataformas de realidad virtual más adecuadas donde experimentar una inmersión mayor que la que puede proporcionar un *smartphone* con *CardBoard*.

#### Oculus Rift

Oculus Rift (imagen 12) [10] es una plataforma desarrollada por Oculus VR, compañía independiente donde tuvo éxito en la financiación de este proyecto a través de *Kickstarter* donde las características principales del dispositivo son:



Imagen 12: Oculus Rift. Fuente: [overclockers.co.uk](http://overclockers.co.uk)

- Pantalla OLED estereoscópica con una resolución total de 2160x1200 (1080x1200 por cada ojo).
- 90 Hz de tasa de refresco.
- Campo de visión de 110°.
- Separación entre lentes ajustable.
- Auriculares de audio 3D integrados.
- Admite el uso de controladores.

Actualmente se puede adquirir desde la página oficial por 449€ donde el paquete viene con el visor, controladores Touch (imagen 13) y dos sensores que rastrean tus movimientos y los trasladan a la realidad virtual.



Imagen 13: Controladores Touch Oculus Rift. Fuente: cnet.com

Por otra parte, es necesario un PC con unas especificaciones mínimas para poder usar esta tecnología:

## Requisitos mínimos

### **Tarjeta gráfica**

NVIDIA GTX 1050 Ti/AMD Radeon RX 470 o superior

### **Tarjeta gráfica alternativa**

NVIDIA GTX 960/AMD Radeon R9 290 o superior

### **CPU**

Intel i3-6100/AMD Ryzen 3 1200, FX4350 o superior

### **Memoria**

8 GB de RAM o más

### **Salida de vídeo**

Salida de vídeo HDMI 1.3 compatible

### **Puertos USB**

1 puerto USB 3.0 más 2 puertos USB 2.0

### **SO**

Windows 10

Este aspecto supone una inversión en el dispositivo de realidad virtual y el PC que ejecutar dicha tecnología.

Como ventaja al utilizar esta herramienta podríamos obtener una mejor experiencia de inmersión en la realidad virtual.

Como desventaja el factor económico nos limita ya que buscamos una iniciación sencilla y asequible para cualquier usuario interesado en la realidad virtual.

## HTC Vive

Otra posible alternativa podría ser el dispositivo creado por HTC y Valve, HTC Vive (figura 14).



*Imagen 14: HTC Vive. Fuente: talkandroid.com*

Este dispositivo al igual que Oculus Rift busca proporcionar al usuario la mayor experiencia de alto nivel en realidad virtual con unas características adecuadas:

### Especificaciones del visor

- Pantallas: Dual AMOLED 3.6" diagonal
- Resolución: 1080 x 1200 pixels por ojo (2160 x 1200 pixels combinadas)
- Refresco: 90 Hz
- Campo de visión: 110°
- Sensores: SteamVR Tracking, G-sensor, giroscopio, proximidad
- Conexiones: HDMI, USB 2.0, stereo 3.5 mm auricular jack, Power, Bluetooth
- Input: micrófono integrado.
- Distancia interpupilar y ajuste de la distancia del objetivo

### Especificaciones del controlador

- Sensores: SteamVR Tracking
- Input: Panel táctil multifuncional, botones de agarre, botón de Sistema, botón de Menú.
- Duración de la batería: Aproximadamente 6 horas
- Conexiones: Micro-USB charging port

Si se requiere un área para que se trace el movimiento del jugador por el espacio la habitación deberá de tener unas dimensiones 2x1,5 metros mínimo y 5 metros máximo.

También necesitaremos unas especificaciones recomendadas por el fabricante para poder hacer funcionar las gafas:



## Requisitos recomendados

### Tarjeta gráfica

NVIDIA® GeForce® GTX 1060 o AMD Radeon™ RX 480, equivalente o superior

### CPU

Intel i5-4590/AMD FX 8350 o superior

### Memoria

4 GB de RAM o más

### Salida de vídeo

Salida de vídeo HDMI 1.4 compatible

### Puertos USB

1 puerto USB 2.0

### SO

Windows 10

Aunque con ciertos requisitos técnicos más asequibles la adquisición de este dispositivo desde la página oficial es de \$499 lo que nos hace decantarnos por otra alternativa a la hora de cumplir con nuestro objetivo en nuestro proyecto.

## Daydream View

Como última alternativa se analiza el dispositivo *Daydream View*[11] de Google la cual podría ser la alternativa que más cerca se queda de la elegida.

El dispositivo viene junto a un controlador inalámbrico (imagen 15) con sensores para detectar la orientación del sensor y la posición de la mano del usuario, se compone de un panel táctil, un botón de acceso al menú, otro con funciones específicas de casa aplicación y un control de volumen.



Imagen 15: Controlador Daydream. Fuente: vr.google.com

No obstante, para poder elegir este dispositivo como base para el desarrollo del proyecto debemos de tener en cuenta que este visor solo funciona en una serie de smartphones específicos [12] de entre los cuales se encontramos marcas como Samsung con Galaxy S8 y S8+, Google con Pixel 2XL y Pixel, etc.

Una ventaja frente a los otros visores analizados sería su precio, que, aunque considerablemente más barato, 109€, hace que debido a la limitación por el terminal móvil y su precio nos decantemos por *Cardboard* en este caso.

## Otras posibles plataformas

Empresas como Samsung con *Samsung Gear VR* (imagen 16) presentaron en 2016 un producto con unas características parecidas a las *Daydream View* con un precio similar de unos 100€ aproximadamente y con limitaciones como el uso exclusivo para terminales Samsung.



*Imagen 16: Samsung Gear VR. Fuente: theverge.com*

Por otro lado, la compañía Sony lanzaba ese mismo año las PlayStation VR (imagen 17) diseñadas para conectarse exclusiva y directamente a la videoconsola PlayStation 4.



*Imagen 17 PlayStation VR. Fuente: blog.eu.playstation.com*

### 4.8.2. Motor de desarrollo

Las posibles alternativas de las que disponemos para realizar un videojuego son realmente amplias, sin embargo, el motor Unity tiene algunas ventajas que lo hace el más indicado para el desarrollo de este proyecto.

Otros motores potenciales para el desarrollo del proyecto de realidad virtual son Unreal Engine 4, desarrollado por la empresa Epic Games [13] (imagen 19) y CryEngine, desarrollado por Crytek [14] (imagen 18). Ambos motores están especialmente enfocados al desarrollo de videojuegos con gráficos en 3D, y disponen de opciones más avanzadas en este aspecto para conseguir resultados más realistas y un mejor rendimiento en hardware de última generación.



*Imagen 19 Logotipo Unreal Engine 4.*

*Fuente: eliassoftware.com*



*Imagen 18 Logotipo CryEngine.*

*Fuente: cryengine.com*

Unity, por otra parte, está centrado en un método de desarrollo más ágil, basado en el uso del editor y scripts en C# que permiten hacer cambios y probarlos rápidamente, tiene una curva de aprendizaje más ligera, la cual lo hace idóneo para la iniciación y entendimiento de cualquier tecnología nueva.

En cuanto a las plataformas, estos tres motores soportan tanto videoconsolas como PC, dispositivos móviles y realidad virtual, aunque en sus primeras versiones Unreal Engine y CryEngine estaban dirigidos principalmente a PC y videoconsolas.

El coste de las licencias de uso es un requisito a tener muy en cuenta para este proyecto. Primero CryEngine es actualmente una herramienta de código abierto y usarla es completamente gratuito. Por otra parte, Unity y Unreal Engine también son gratuitos para un uso personal o académico, pero si el desarrollo tiene fines económicos se deberá de pagar una licencia.

### **4.8.3. Controlador de realidad virtual**

Como alternativas para el control que interactúa con el entorno en nuestra realidad virtual se pueden considerar varias alternativas.

Desde hacer uso de la simple vista del usuario para interactuar con el entorno, el uso de un controlador bluetooth que se conecte con nuestro dispositivo Android o hacer uso de los controladores si el equipo de realidad virtual los trae consigo.

En el caso de este trabajo se opta por la simulación de un mando del dispositivo *Daydream View* junto a la interacción visual que el usuario puede aportar.

#### 4.8.4. Solución

En definitiva, las herramientas para el desarrollo de esta guía y proyecto serán:

- Unity debido a su gran extensión y su versión gratuita.
- Desarrollo en Android al ser uno de los sistemas más utilizados en dispositivos móviles y versatilidad.
- *CardBoard* junto a *Daydream* serán la base del proyecto al ser asequible su acceso y fácil entendimiento de la realidad virtual.
- Emulación del controlador que interactuará con el entorno por su fácil funcionamiento y conexión junto a otros dispositivos.

Básicamente con este proyecto se busca la introducción a la realidad virtual con las herramientas más asequibles, pero a su vez más potentes que podemos encontrar.

## 5. Diseño de la solución

---

En el presente apartado se dará una idea sobre el diseño y la creación de la guía adjunta a este trabajo [Anexo 1] donde se explica paso a paso desde los requisitos a nivel de hardware, pasando por la instalación de todas las herramientas necesarias y la implementación y modificación de código que se realiza para el proyecto.

### 5.1. Diseño Detallado

Se plantea un esquema donde poder ver el sistema de cual se compondrá el desarrollo final (imagen 21), siendo el flujo desde la creación de la aplicación mediante la guía, la preparación del smartphone que hará de controlador la ejecución de la aplicación creada y finalmente la prueba de las características añadidas del proyecto.

No obstante, el principal objetivo conjunto a este desarrollo es la guía de desarrollo del videojuego en realidad virtual que ayudará a un desarrollador a iniciarse en esta tecnología (imagen 20).



*Imagen 20: Flujo básico del objetivo del TFG*

El desarrollo final que se consigue con la guía presentada es una pequeña demostración que se puede conseguir uniendo las tecnologías de *CardBoard* y *Daydream* para aquellas personas que no pueden o quieren invertir en un sistema de realidad virtual de mayor calidad.

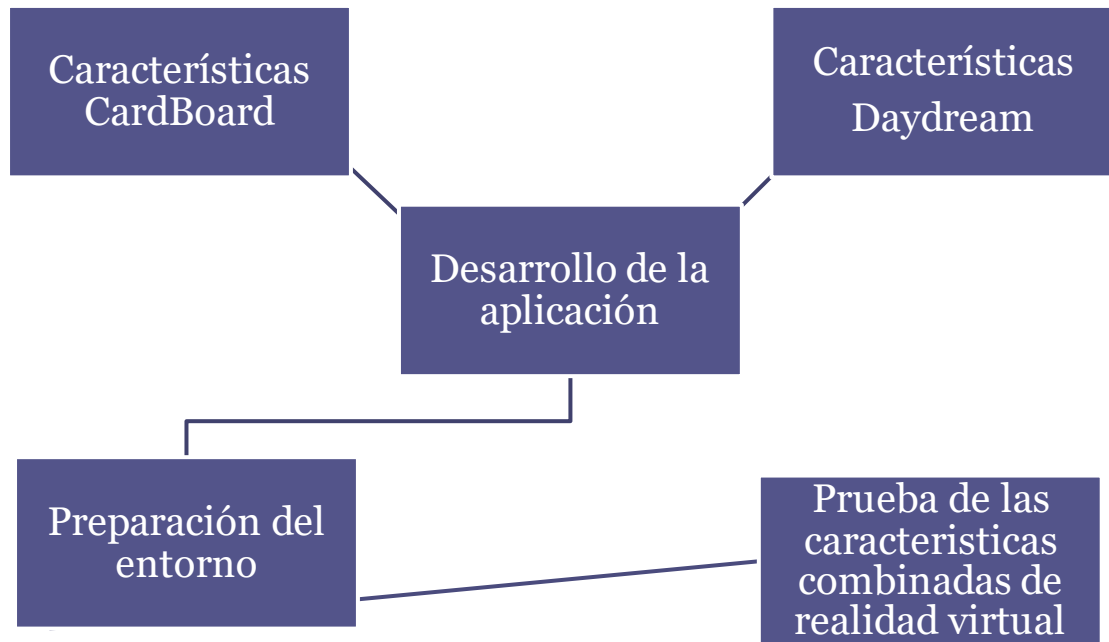


Imagen 21: Flujo de desarrollo con distintas tecnologías

Una de las principales funciones en la que se basa un sistema de realidad virtual es el movimiento de cabeza que realiza el usuario y una funcionalidad indispensable en un proyecto como este.

Para tener un control mayor sobre lo que realmente el usuario está mirando en el entorno en la guía se muestra cómo crear e interactuar con ciertos objetos con una mirilla que establecerá el punto focal de nuestra mirada como se muestra en la siguiente imagen.

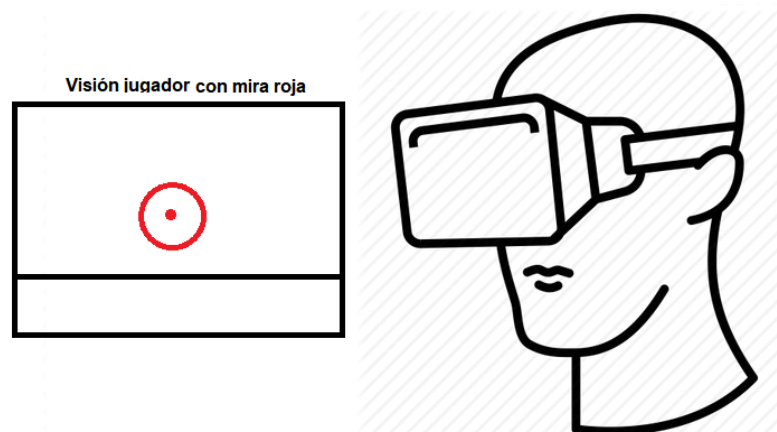
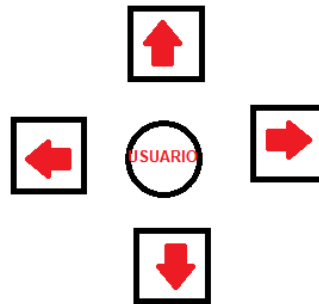


Imagen 22: Reticula en el campo de visión del usuario

Las funciones que nos plantea *CardBoard* son realmente sencillas ya que se limitan a movimiento de cabeza que el usuario puede realizar por el entorno desde una posición fija.

Con esta limitación aprovecharemos este efecto moviéndonos por un espacio delimitado. El movimiento se realiza cuando el usuario mira directamente sobre uno de los 4 cuadrados que tiene a su alrededor.



*Imagen 23: Diseño base de movimiento del usuario*

Esta funcionalidad nos permitirá explotar un poco más las funciones principales de *CardBoard* y plantear una alternativa al movimiento que puede dar un controlador externo.

El sistema de interacción con el entorno se relaja mediante un pequeño juego de disparos donde el usuario al disparar desde el controlador externo disparará una bala que al colisionar con alguna de las formas que flotan en el entorno éstas desaparecerán y sumará una pequeña puntuación en un HUD (información que ve en pantalla el jugador) (imagen 24). Si la bala fallase esta desaparecería en 10 segundos.



*Imagen 24: HUD básico del usuario en la aplicación*

Entrando en las opciones que nos proporciona *Daydream* la limitación viene debido a que el dispositivo que hace de controlador no dispone de los sensores que el mando de *Daydream* posee originalmente. No obstante, con la emulación de éste en nuestro smartphone secundario lograremos hacer el sistema de disparos mediante un simple gesto en la pantalla del móvil emulador.

El sistema de colisiones que dispone el juego es básico pero eficiente haciendo que la tanto el usuario no pueda traspasar los muros que albergan el entorno y las balas rebotan contra dichos muros y formas hasta desaparecer.

## 5.2. Tecnología utilizada

Las tecnologías principales utilizadas para el desarrollo que han hecho posible la creación de la guía son varias:

- Librerías *GoogleVR SDK* para Unity v1.130.1.
- *SDK* de Android-28.
- Unity 2018.1.5f1.
- *Smartphone* con sistema operativo Android 7.0.
- *Smartphone* con sistema operativo 5.0.
- *Visual Studio Code* como editor de código.

Todo ese proyecto se ha desarrollado en un computador con un sistema operativo Windows 10.

Durante el desarrollo de la guía y la aplicación fue ciertamente costosa la preparación de todo el entorno como el aprendizaje y entendimiento de las librerías de *Google VR*. Entendiendo que dentro de estas librerías la implantación de funcionalidades de *Daydream* y *CardBoard* estaban separadas, pero a su vez la tecnología *Daydream* toma el control principal de estas librerías donde hay demostraciones en las cuales puedes observar algunas funcionalidades.

El desarrollo y aprendizaje en Unity ha hecho que por mi parte adquiera conocimientos básicos y medios sobre el funcionamiento de un motor de videojuegos, ya que suponía un reto enfrentarse a una tecnología nueva como es la realidad virtual junto al desarrollo en un motor de videojuegos que no he utilizado con anterioridad.

El lenguaje utilizado es C# con el cual ya estaba familiarizado por el ámbito laboral donde adquirí destreza con el lenguaje de programación y a permito seguir con la evolución y practica en programación.

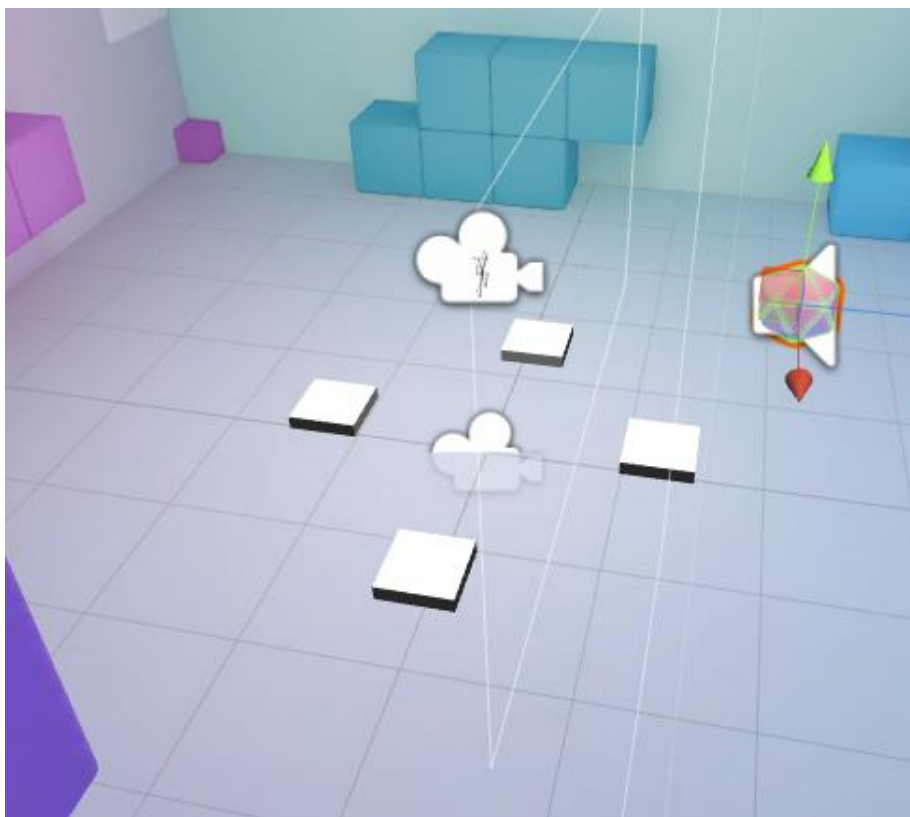


## 6. Desarrollo de la solución

---

Al tratarse de un desarrollo basado en una guía para implantar a nuevos usuarios conocimientos sobre la realidad virtual, en dicha guía adjunta [Anexo 1] se especifican aspectos básicos como la instalación y preparación del entorno de trabajo, desarrollo y explicación del diseño anteriormente mencionado.

Algunos aspectos destacables de la guía sería, entre otras funcionalidades, el movimiento del jugador por el espacio. El movimiento se basa en moverse en dos ejes principales X y Z mientras la mirada del usuario permanezca en uno de los cuadrados del suelo.



*Imagen 25: Vista básica de la posición de los cuadrados de movimiento*

La rotación del usuario sobre su eje y mirando otro cuadrado cambia automáticamente la dirección de movimiento. Este movimiento se ve interrumpido cuando el jugador mueve la retícula fuera de los cuadrados.

El código que acompañaría a esta funcionalidad sería el siguiente:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMovement : MonoBehaviour {

    // Use this for initialization

    public Transform playerTf;

    Vector3 playerVector;

    void Start () {

    }

    // Update is called once per frame
    void FixedUpdate () {
        playerTf.position += playerVector;
    }

    public void moveXIn(float x) {
        playerVector = new Vector3(x,0,0);
    }

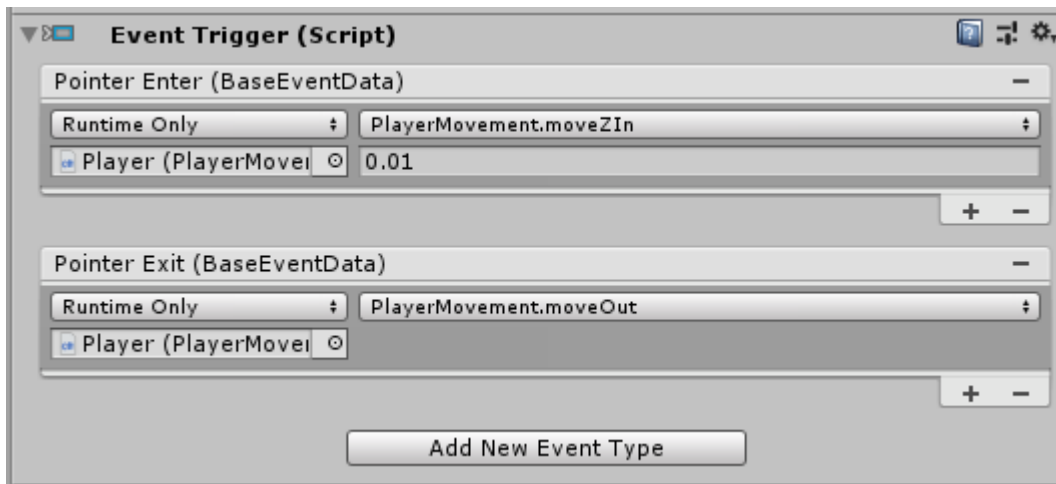
    public void moveZIn(float z)
    {
        playerVector = new Vector3(0, 0, z);
    }

    public void moveOut() {
        playerVector = new Vector3(0, 0, 0);
    }
}
```

Como se puede observar la clase o script que acompañaría a esta funcionalidad vendría dada por una función:

- `FixedUpdate()`: Por cada *frame* o imagen por segundo se actualiza la posición del jugador, *playerTf*, en función de un vector que se modifica por los otros métodos.
- `moveXIn()`: Esta función suma una variable *x* a un vector la posición *X*.
- `moveZIn()`: Mismo funcionamiento con el eje *Z*.
- `moveOut()`: Al retirar la mirada del cuadrado se activaría esta función que deja de sumar la posición.

En Unity una vez creado este script se le asociaría a cada cuadrado creado un *Event Trigger* que ejecutaría por distintas acciones alguna funcionalidad:



*Imagen 26: Ejemplo de asociación de script creado con valores de entrada*

Este *Event Trigger* tiene un *Point Enter* que asocia la visión del jugador sobre el objeto y un *Point Exit* que se activaría al dejar de mirar dicho cuadrado.

A cada uno de estos *Event Type* se le asociaría el script, y seleccionaría una función adecuada (*<nombre del script>.moveZIn* y *<nombre del script>.moveOut*) y un valor de entrada por defecto, 0.01 en nuestro caso que marcaría la velocidad a la que se desplazaría nuestro jugador.

Este es solo un ejemplo de algunas funcionalidades que la guía posee en su interior y se explica con más detalle.

Para la preparación de la guía, previamente se adquirieron conocimientos óptimos en distintos campos como son en el desarrollo de videojuegos con el motor de videojuegos Unity y librerías de *Google VR* para poder desarrollar una guía completa sobre las bases de la realidad virtual.



## 7. Puesta a punto y Pruebas

Si se ha seguido el desarrollo de la guía se habrá obtenido como solución una muestra de lo que la realidad virtual puede ofrecer en cuanto a el campo de los móviles inteligentes sin una inversión en tecnología de realidad virtual mas allá de la soportada por el propio *smartphone*.

Para la preparación y ejecución de la aplicación se debe de disponer de dos terminales móviles con un sistema operativo Android con versiones 7.0 y 4.4. En este caso en particular los terminales son un Samsung Galaxy S6 Edge como dispositivo principal y un Motorola C como *smartphone* que hará de controlador.

Una vez obtenido el archivo *.apk* generado por Unity en el dispositivo principal, se modificará en ajustes la opción de <<Fuentes desconocidas>>:

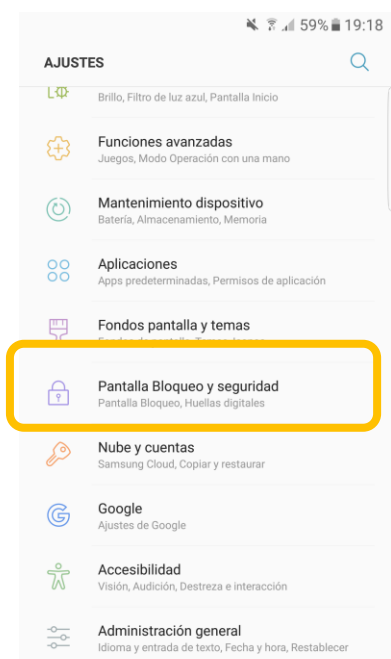


Imagen 27: Pantalla Ajustes de un sistema Android

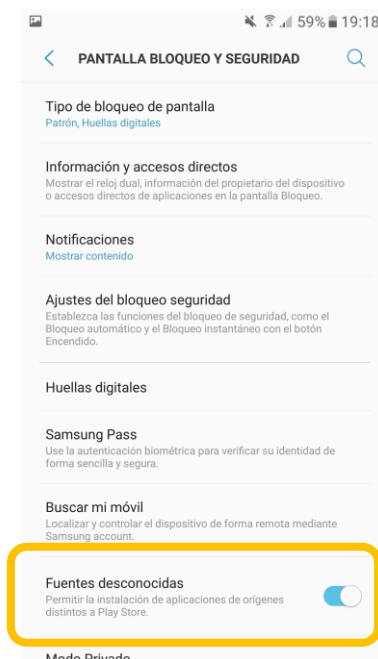


Imagen 28: Pantalla menú Pantalla de bloqueo y seguridad

Con esta opción se dan permisos al dispositivo para la instalación de la aplicación.

Una vez dado los permisos se busca la aplicación en el terminal allá donde se haya guardado y procedemos a su instalación:

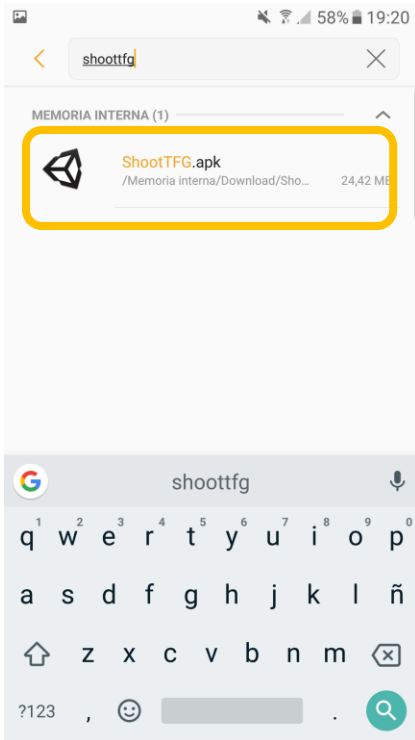


Imagen 30: Aplicación del dispositivo

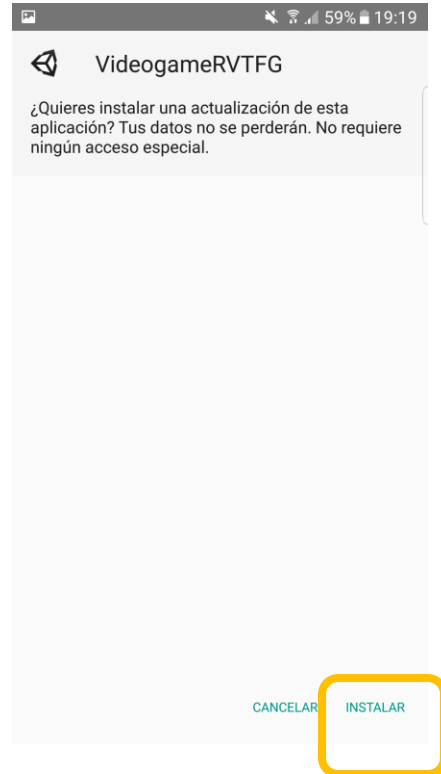


Imagen 31: Inicio de la instalación

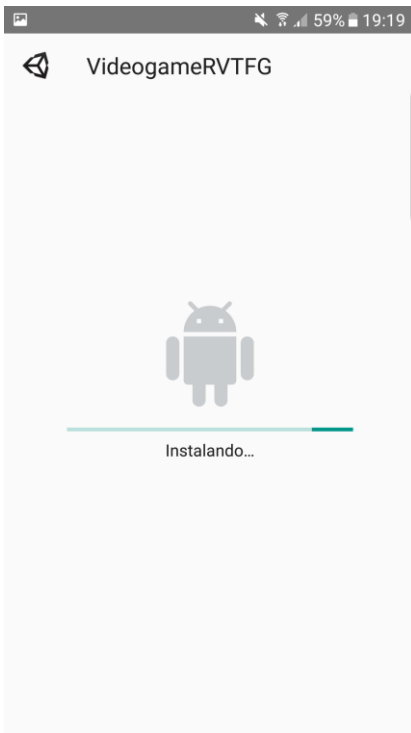


Imagen 29: Procedimiento de instalación

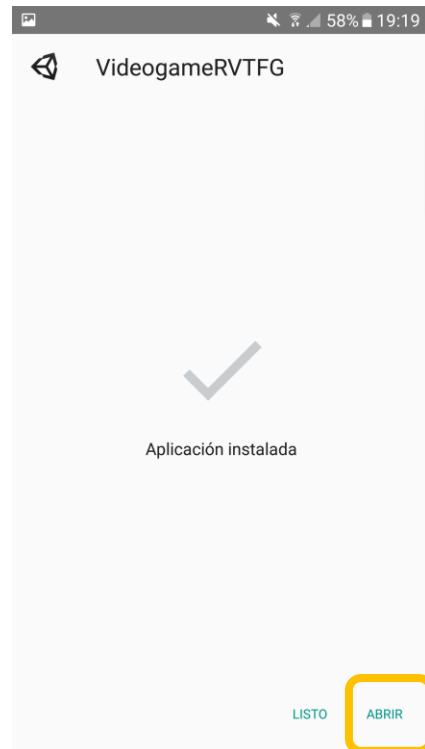
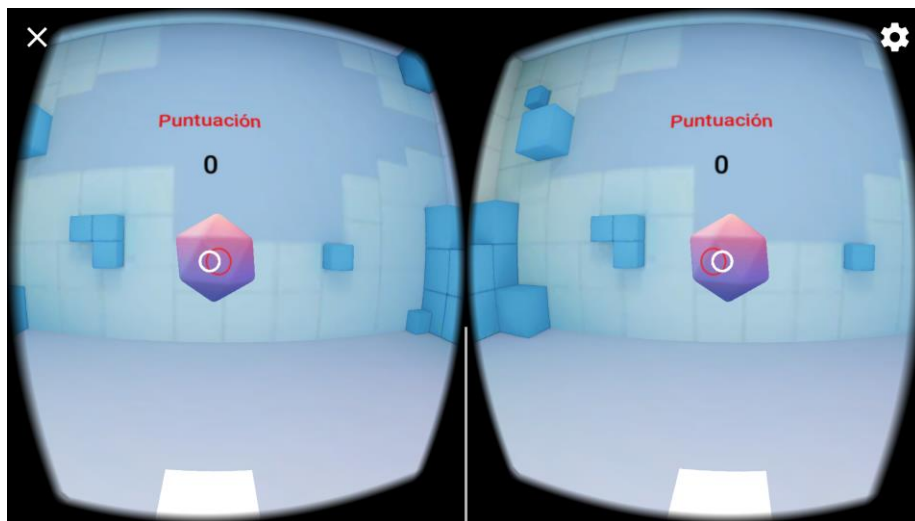


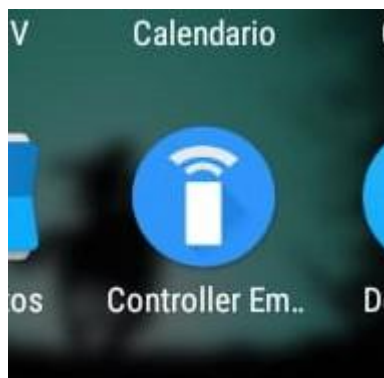
Imagen 32: Instalación efectuada correctamente

Si la instalación se hecho correctamente se abrirá el juego y automáticamente el dispositivo entrará en modo de realidad virtual, presentándonos el desarrollo efectuado.



*Imagen 33: Visualización desarrollo final*

Seguidamente y antes de interactuar con el entorno se conectará el dispositivo que hará de controlador. Desde el dispositivo secundario abrir la aplicación *The Controller Emulator* (imagen 34), tener en cuenta que la conexión bluetooth debe estar activa en ambos dispositivos, automáticamente preguntará por la sincronización con dispositivos cercanos. Seleccionar el smartphone principal y ya quedaría la conexión establecida.



*Imagen 34: Icono aplicación The Controller Emulator*

Por otra parte, mediante el engranaje que aparece en la parte superior derecha de la pantalla acceder al menú de la aplicación por defecto (imagen 35). Una vez dentro del menú seleccionar:

**Opciones del desarrollador > Controller emulator device**

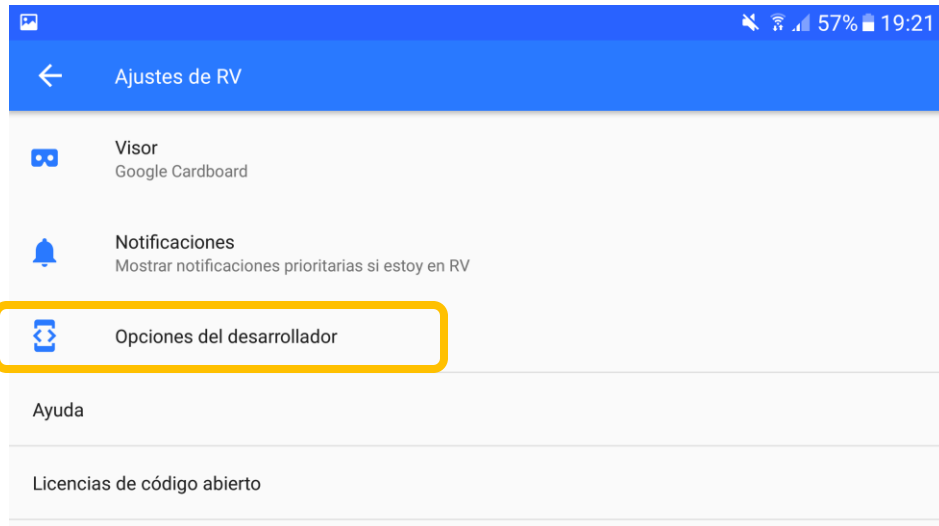


Imagen 35: Menú principal de la aplicación de realidad virtual.

Seguidamente aparecerá una lista de los dispositivos sincronizados con este terminal, seleccionar el dispositivo que contiene la emulación del mando y automáticamente veremos como el *smartphone* secundario notifica del éxito de la conexión (figura 36).

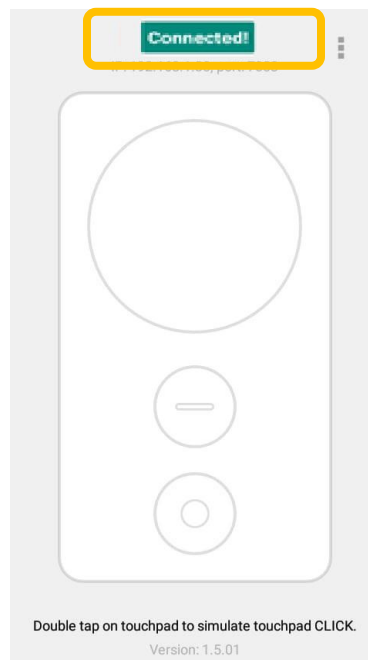


Imagen 36: Interfaz The Controller Emulator

Una vez hechas las conexiones se probará la aplicación el correcto funcionamiento de las funcionalidades implantadas con anterioridad como pueden ser:



- Sistema de movimiento de cabeza (imagen 37).



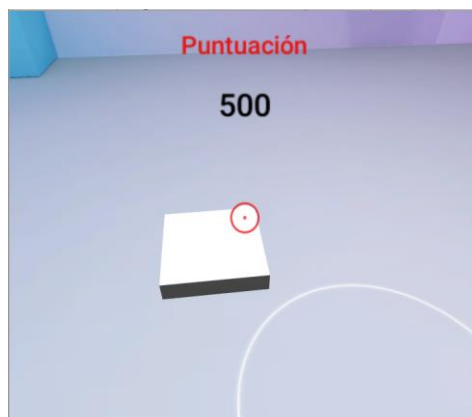
*Imagen 37: Vista movimiento de cabeza en aplicación.*

- Sistema de disparos y puntuación (imagen 38).



*Imagen 38: sistema de disparos con bala y puntuación*

- Movimiento por la escena (imagen 39)



*Imagen 39: Movimiento mirando las plataformas del suelo.*



## 8. Conclusiones

---

Con la creación de este proyecto se ha realizado con éxito los objetivos principales del trabajo, la creación de una guía de desarrollo en *Unity* mediante *Google VR* y el desarrollo de un pequeño videojuego mostrando las funcionalidades oportunas.

En primer lugar, la creación de una guía detallada para introducir a nuevos desarrolladores en la realidad virtual se ha enfocado desde un nivel bajo de la tecnología como puede ser la realidad virtual en *smartphones*.

Utilizando *Google VR* y una de sus demostraciones básicas se puede ver y entender el funcionamiento de esta tecnología y sus limitaciones. Estas limitaciones son claras, debido al uso de dispositivos móviles como plataforma donde ejecutar la aplicación limita la sensación de inmersión debido a las especificaciones técnicas de estos dispositivos. Por otra parte, tanto la interacción con el entorno y movimiento en éste está limitado de igual forma al no disponer de sensores que nos ayuden en este objetivo.

Esta guía establece las bases y funcionalidades básicas de la realidad virtual de forma sencilla e intuitiva gracias al motor Unity y establece un inicio para poder seguir con el desarrollo del videojuego implantando funcionalidades de forma creativa o evolucionando a nuevas herramientas en el sector de la realidad virtual.

Este trabajo ha supuesto un reto a nivel teórico debido a la poca familiaridad que se pueden adquirir durante los estudios universitarios con alguna de las herramientas utilizadas durante el desarrollo como es Unity o tecnologías actuales de realidad virtual. No obstante, al cursar la rama de computación y haber cursado asignaturas de sistemas gráficos e interactivos has ayudado a entender el funcionamiento de un videojuego.

En definitiva, se considera que se han logrado los objetivos planteados en el comienzo del trabajo tanto a nivel teórico, introduciendo las nuevas tecnologías de realidad virtual y el trabajo realizado con el motor Unity, todo ello confluyendo en una guía resultante, y a nivel practico mostrando una demostración de las bases fundamentales de la realidad virtual.



## 9. Trabajos futuros

---

En cuanto a trabajos futuros se pueden abordar distintos caminos para el continuo aprendizaje con la realidad virtual.

Por una parte, se puede seguir con el desarrollo de la demostración implantada en este trabajo dando desde:

- La evolución cambiando formas e interfaces.
  - El modelado puede evolucionar y texturizarse acorde a las funcionalidades que se le quieran dar al videojuego.
  - Una creación de menú principal.
- Añadiendo funcionalidades nuevas como un sistema de movimiento más elaborado con la ayuda de controladores externos.
  - El uso de controladores bluetooth puede desembocar en un sistema que permita al jugador saltar e interactuar de una forma más fluida con el entorno.
- Creación de un sistema de niveles y avance en el videojuego.
  - En cuanto al desarrollo se le puede dar profundidad añadiendo dificultad haciendo que los agentes a los que les disparen te devuelvan esos disparos.

Otro punto de avance en este trabajo puede ser dar el salto a un sistema de realidad virtual más completo y cercano al objetivo real de esta tecnología, la inmersión total del jugador. Dando el paso a una tecnología más avanzada se pueden migrar funcionalidades básicas que se adoptaron a los límites que nos suponía *CardBoard* y *Daydream* como el movimiento por el entorno o la interacción con este mismo con los mandos apropiados que hacen que nuestros movimientos se reflejen en el entorno virtual. Desarrollo en plataformas más evolucionadas permitirán crear videojuegos más elaborados y complejos con entornos más realistas y mejores físicas exprimiendo al máximo el computador en el cual se desarrolle el proyecto.





# 10. Referencias

---

- [1] Sensorama  
Disponible en <<http://www.medienkunstnetz.de/works/sensorama/>>  
[Fecha de consulta: 23/04/2018]
- [2] Historia de la realidad virtual  
Disponible en <[https://en.wikipedia.org/wiki/Virtual\\_reality](https://en.wikipedia.org/wiki/Virtual_reality)>  
[Fecha de consulta: 23/04/2018]
- [3] The Cave: audio visual experience automatic virtual environment  
Disponible en <<http://dl.acm.org/citation.cfm?doid=129888.129892>>  
[Fecha de consulta: 23/04/2018]
- [4] Sega VR  
Disponible en <[http://segaretro.org/Sega\\_VR](http://segaretro.org/Sega_VR)>  
[Fecha de consulta: 27/04/2018]
- [5] Máquinas Virtuality  
Disponible en <[https://en.wikipedia.org/wiki/Virtuality\\_\(gaming\)](https://en.wikipedia.org/wiki/Virtuality_(gaming))>  
[Fecha de consulta: 12/05/2018]
- [6] Documentación Unity  
Disponible en <<https://unity3d.com/es/learn>>  
[Fecha de consulta: 20/04/2018]
- [7] Plataformas Unity  
Disponible en <<https://unity3d.com/es/unity/multiplatform/>>  
[Fecha de consulta: 02/06/2018]
- [8] SDK Google VR para Unity  
Disponible en: <<https://developers.google.com/vr/>>  
[Fecha de consulta: 03/04/2018]
- [9] The Controller Emulator  
Disponible en:<<https://developers.google.com/vr/daydream/controller-emulator>>  
[Fecha de consulta: 06/04/2018]
- [10] Sitio web de Oculus Rift disponible en <<https://www.oculus.com/>>  
[Fecha de consulta: 06/05/2018]
- [11] <<http://www.androidauthority.com/daydream-vr-ready-phones-specs-727780/>>  
[Fecha de consulta: 16/05/2018]
- [12] Teléfonos compatibles con Daydream  
Disponible en:  
<[https://vr.google.com/intl/es\\_es/daydream/smartphonevr/phones/](https://vr.google.com/intl/es_es/daydream/smartphonevr/phones/)>  
[Fecha de consulta: 16/05/2018]

[13] Web de Unreal Engine

Disponible en < <https://www.unrealengine.com/what-is-unreal-engine-4> >

[Fecha de consulta: 07/05/2018]

[14] Web de CryEngine

Disponible en < <https://www.cryengine.com/> >

[Fecha de consulta: 07/05/2018]



# 11. Anexo

---

[Anexo 1] **Guía de desarrollo de un videojuego en realidad virtual:**

## Guía de desarrollo de un videojuego en realidad virtual



Autor: Víctor Jarilla Romo

# Índice

Guía de desarrollo de un videojuego en realidad virtual .....	1
Índice .....	2
Introducción .....	3
Objetivo .....	3
Primeros pasos .....	4
Herramientas necesarias e instalación .....	4
Instalaciones y revisiones .....	4
Smartphone .....	4
<i>Cartonglass</i> .....	6
Dispositivo de control bluetooth .....	6
Unity3D .....	7
<b>Punto a tener en cuenta ante la instalación de Unity, preparación del                 smartphone y preparación del proyecto de Unity</b> .....	8
<b>Preparación de nuestro Smartphone y entorno para trabajar con Android</b> .....	8
Google VR en Unity3D .....	13
Configuración del proyecto de Unity con Google VR .....	14
Configuración de opciones de construcción y opciones de jugador .....	16
<b>Conceptos sobre el desarrollo del proyecto</b> .....	17
<b>Desarrollo</b> .....	17
<b>Estudio de la demostración básica</b> .....	17
<b>Cambios para realizar en la demostración</b> .....	18
Creación de HUD base .....	18
Movimiento del usuario .....	20
Interacción con el entorno (Disparo del jugador) .....	23
Creación de la bala .....	24
Sistema de disparos del jugador .....	27
Sistema de puntuaciones .....	31
Exportación en nuestro dispositivo móvil .....	34

# Introducción

La realidad virtual representa desde el 2016 un salto tecnológico que puede otorgar al mundo del entretenimiento, laboral y educativo un aspecto de implicación debido a su mayor inversión en la realidad creada por el desarrollador.

Un primer paso para el desarrollo de videojuegos con esta tecnología puede ser en el mundo del Smartphone. En este ámbito algunas de las principales compañías como *Google* con *Daydrem*s o *Samsung* con *Samsung VR* han desarrollado sus propios equipos de realidad virtual (RV). *Google* incluso ha creado librerías y extensiones para motores gráficos para poder desarrollar proyectos en RV.

Esta guía está dirigida a aquellas personas con conocimientos previos en programación en C# y motor gráfico *Unity3D* ya que no se explican los conceptos básicos de programación o *Unity3D* que el usuario debe de tener para comprender y seguir con facilidad la guía.

En dicha guía se ayuda y explica paso a paso desde la instalación del motor gráfico y extensiones correspondientes de realidad virtual hasta el propio desarrollo del videojuego utilizando como base del videojuego una de las demos que vienen con las librerías de *Google VR* añadiendo elementos e interacciones con el entorno. Además, utilizaremos las herramientas el motor gráfico *Unity3D* y como soporte de todo esto un Smartphone con sistema operativo Android.

## Objetivo

El objetivo final de la guía es orientar y dar a conocer la tecnología de realidad virtual enfocados al mundo de los *smartphones* mediante *Google VR* utilizando unas *cartonglass* que sirven exclusivamente como soporte para el *Smartphone* y un dispositivo móvil que interactuará con el entorno en nuestro desarrollo de realidad virtual.

Todo esto confluye en una pequeña demostración de una versión inicial de un videojuego con diferentes interacciones con el entorno.



## Primeros pasos

### Herramientas necesarias e instalación.

Para poder desarrollar y aprender a utilizar este tipo de tecnologías necesitamos unos requisitos mínimos que cumplir a nivel de hardware, sobre todo en nuestro Smartphone.

- 1- Dispositivo *Android* con sistema operativo *Android 7.0* o mayor.
- 2- Unas *cartonglass* o soporte donde colocar el *Smartphone*.
- 3- Dispositivo *Android* con sistema 5.0 para el soporte del emulador.
- 4- Motor gráfico *Unity3D* en su versión más reciente.
- 5- Extensiones de *Unity3D*, *Google VR*.

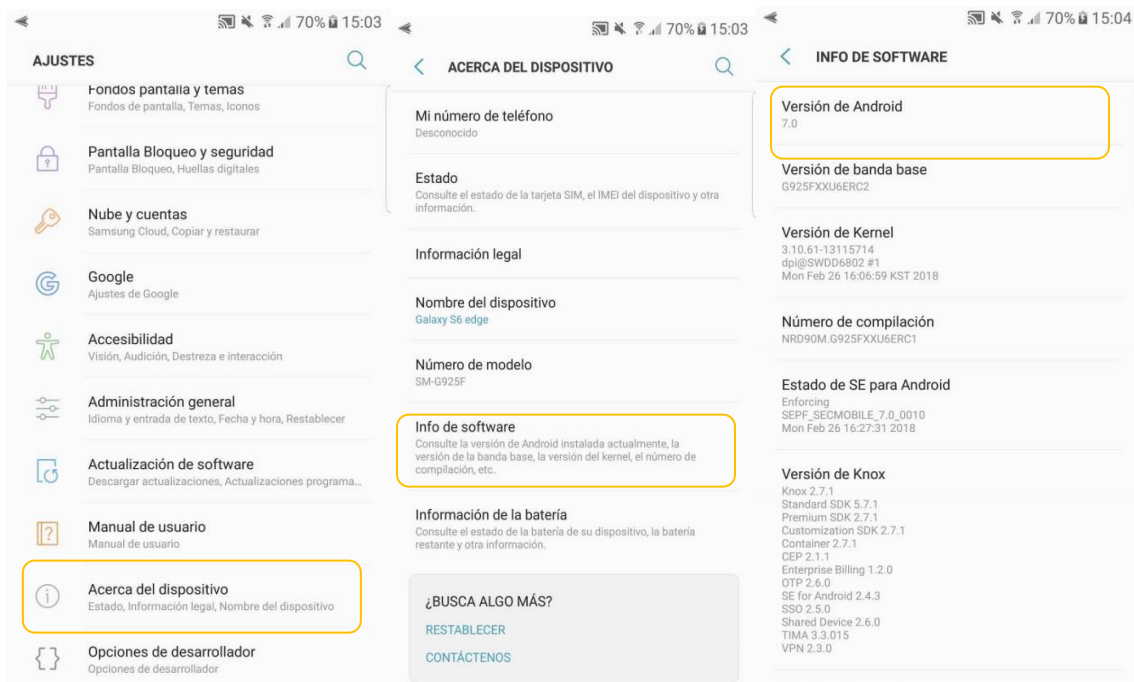
### Instalaciones y revisiones

En este apartado comprobaremos las versiones de nuestros dispositivos, programas instalación y preparación de todos los entornos para una posterior realización del proyecto de videojuego en realidad virtual.

#### Smartphone

Primero deberemos de comprobar la versión de nuestro dispositivo móvil para ello tenemos que acceder a la siguiente ruta:

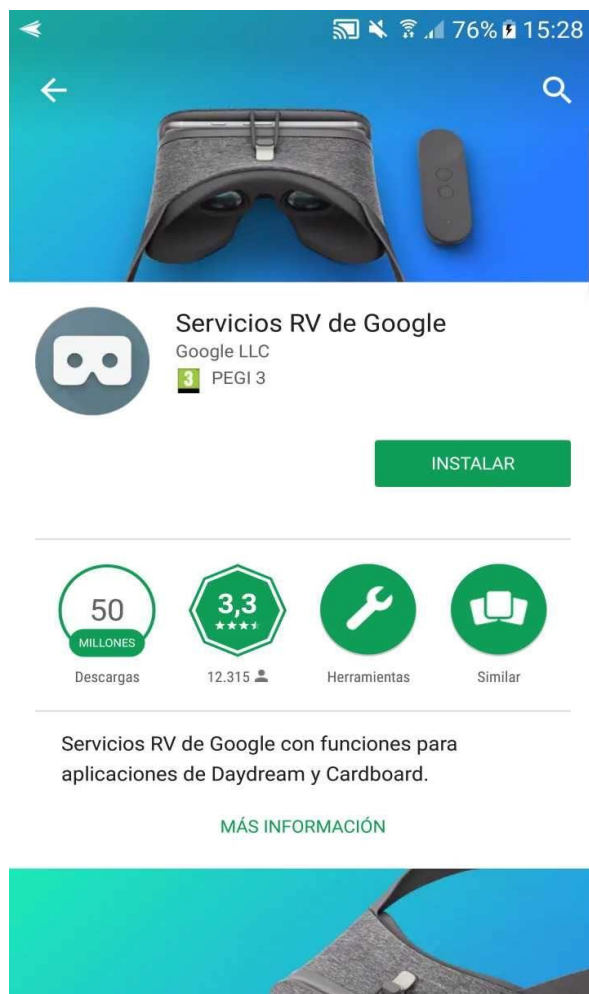
Ajustes/ Acerca del dispositivo/ Información del software



Como se observa en la imagen la versión de nuestro dispositivo Android es la 7.0.

Debido a compatibilidades con las librerías de *Google VR* nuestro dispositivo debe de tener dicha versión como mínimo para poder hacer uso de las funcionalidades que vemos en la guía.

Una vez comprobada la versión de nuestro dispositivo deberemos de instalar desde la *PlayStore* la aplicación *Servicios RV de Google*, esta aplicación de Google nos proporciona las herramientas necesarias para poder dar, entre otras, un soporte desde Unity3D a nuestro móvil y poder realizar pruebas en vivo de nuestro desarrollo mediante la conexión con un cable USB desde nuestro dispositivo móvil hasta nuestro PC.



### *Cartonglass*

Aunque no imprescindible o necesario, para poder obtener una experiencia más auténtica de esta tecnología poseer unas *cartonglass* ayudará a conseguir una inmersión en nuestro proyecto más realista.



### Dispositivo de control bluetooth

Como dispositivo de control utilizaremos otro smartphone que simulara un controlador de las gafas Daydream de Google.



Como soporte de instalación deberemos de instalarnos *The Controller Emulator* en nuestro dispositivo que hará de controlador.

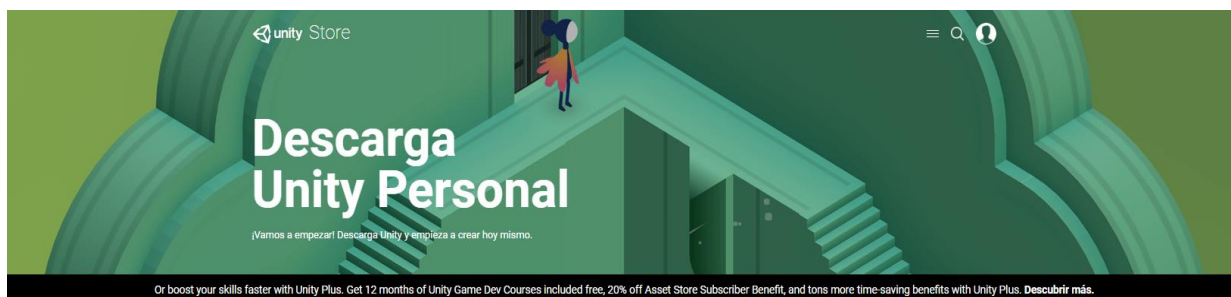


## Unity3D

Con este motor realizaremos el desarrollo del proyecto y veremos las principales funciones que nos proporciona en conjunto a la realidad virtual.

Utilizaremos la versión gratuita *Unity3D Personal* para nuestro desarrollo. Para poder instalar una versión reciente de Unity3D bastará con acceder a esta dirección oficial donde podremos descargar de forma gratuita una versión de Unity3D actualizada.

<https://store.unity.com/es/download?ref=personal>



### Aceptar términos

Al hacer clic, confirmo que puedo utilizar Unity Personal conforme a los [Términos de servicio](#), ya que mi compañía o yo cumplimos con los siguientes requisitos:

- No percibimos más de \$100 mil en ingresos brutos anuales, independientemente de si Unity Personal se usa para fines comerciales, o para un proyecto o prototipo interno.
- No hemos recaudado fondos por más de \$100 mil.
- En este momento no estamos usando Unity Plus o Pro.

Si no cumples los requisitos para usar Unity Personal, haz clic [aquí](#) para conversar con nuestro equipo sobre el producto más interesante para ti.

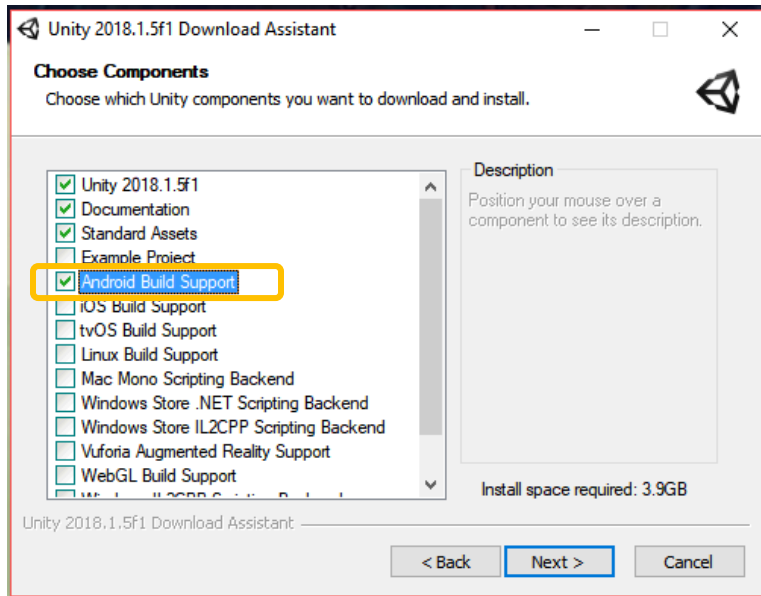
Descargar el instalador para Windows

Descarga Unity Hub (Vista previa)

¿Quieres descargar el instalador para Mac OS X?  
Elige Mac OS X

## Punto a tener en cuenta ante la instalación de Unity, preparación del smartphone y preparación del proyecto de Unity

Hay que tener en cuenta que durante la instalación de Unity3D tengamos habilitado la opción de *Android Build Support* para poder seguir sin problemas con el desarrollo.



## Preparación de nuestro Smartphone y entorno para trabajar con Android

Otro de los puntos a tener en cuenta es tener preparado nuestro entorno de trabajo para el desarrollo en Android.

### 1- Instalación del kit de desarrollo de Java.

Descargue e instale [Java Development Kit \(JDK\)](#) . Para utilizarlo con Unity es necesario la versión de 64 bits JDK 8 (1.8).

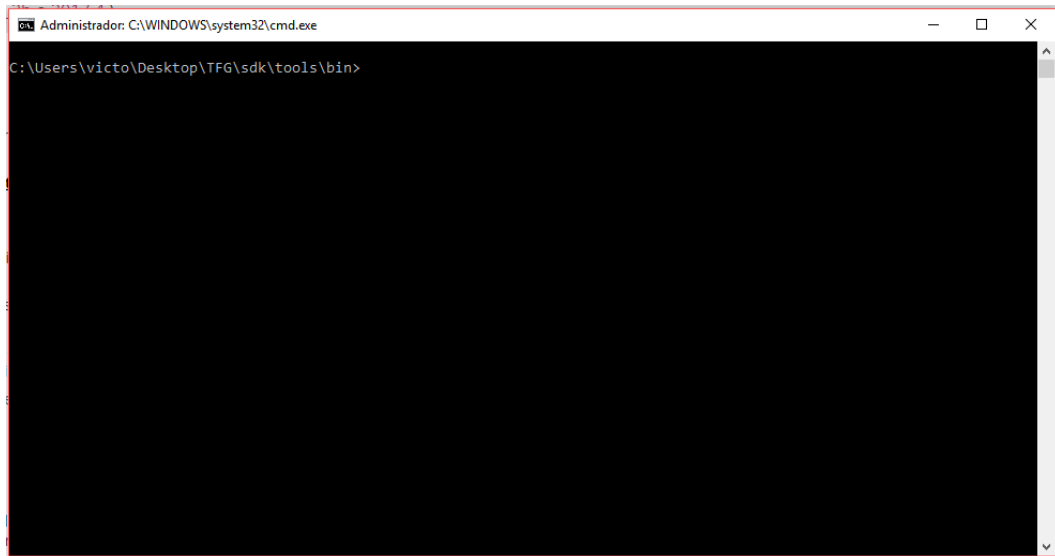
### 2- Instalación del SDK de Android.

Nuestra instalación del SDK lo haremos desde una línea de comandos, aunque también existe la posibilidad de instalar desde *Android Studio*.

- 2.1- Descargue la [herramienta de línea de comandos del software Android](#) .
- 2.2- Descomprime la carpeta de herramientas en una ubicación en tu disco duro.
- 2.3- Abra una ventana de comandos.
- 2.4- Navegue a la carpeta *bin* en la ubicación donde descomprimió la carpeta de herramientas:

```
"install folder"> tools> bin
```

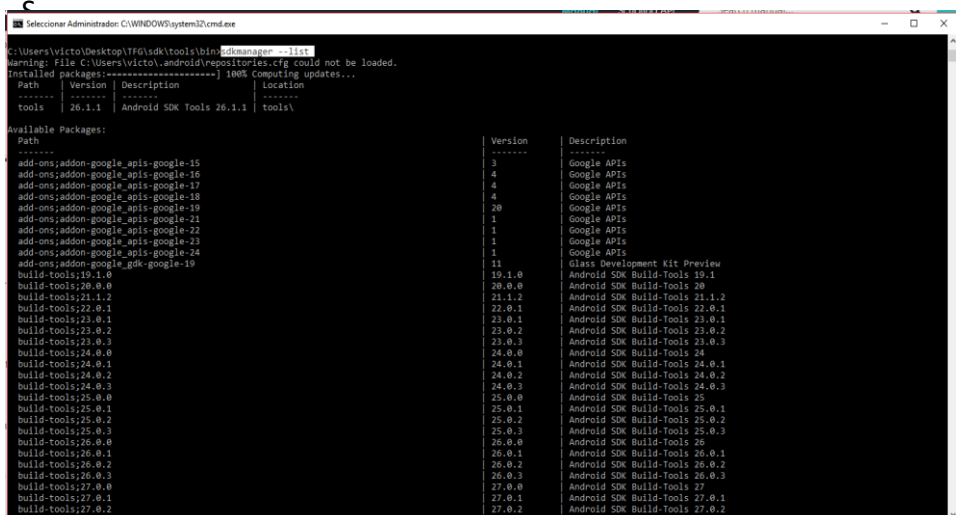




2.5- Use la herramienta de línea de comandos *sdkmanager* para recuperar la lista de paquetes que puede instalar. Los paquetes instalables incluyen *Platform SDKs*, *Build Tools*, *Platform tools* y otras herramientas.

`sdkmanager--list`

2.6-



versión de *Platform SDK* para instalar. Los *Platform SDK* tienen la siguiente forma en la lista:

`platforms;android-xx.`

El xx indica el nivel de SDK. Cuanto mayor sea el número, más nuevo será el paquete. El formato general del comando para la instalación del paquete es:

`sdkmanager <nombre del paquete>.`

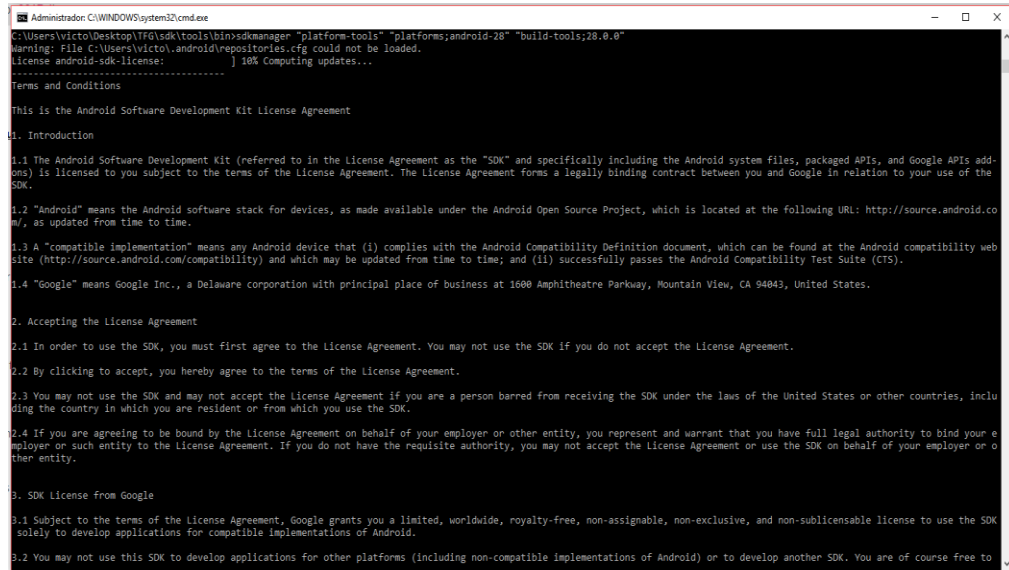
Puede instalar las Herramientas de plataforma y las Herramientas de compilación correspondientes al mismo tiempo.

En nuestro caso ejecutaremos esta línea de comandos:



sdkmanager "platforms-tools" "platforms; android-28" "build-tools; 28.0.0"

A



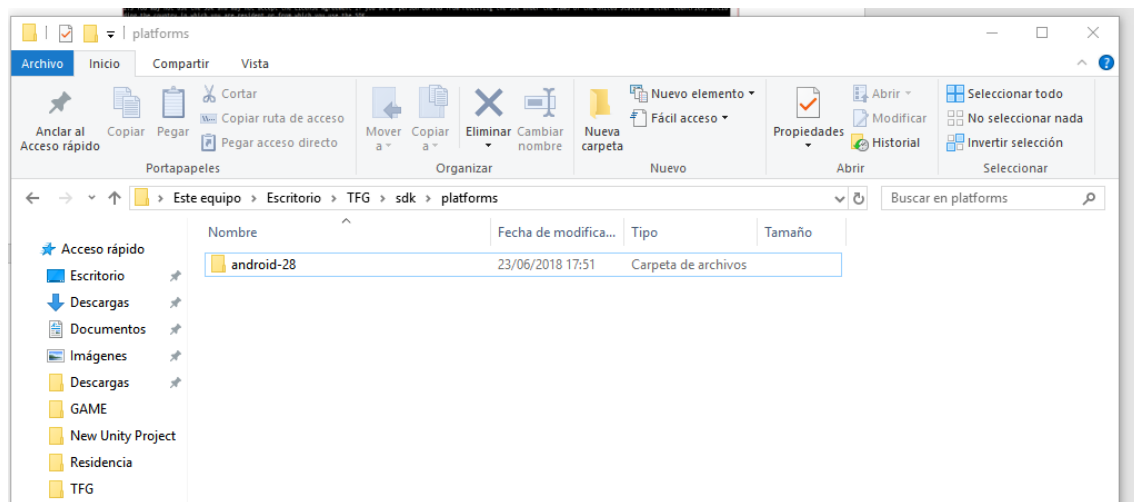
términos y condiciones y una vez finalizada la instalación

2.7- Si está ejecutando en Windows, instale los controladores del dispositivo USB.

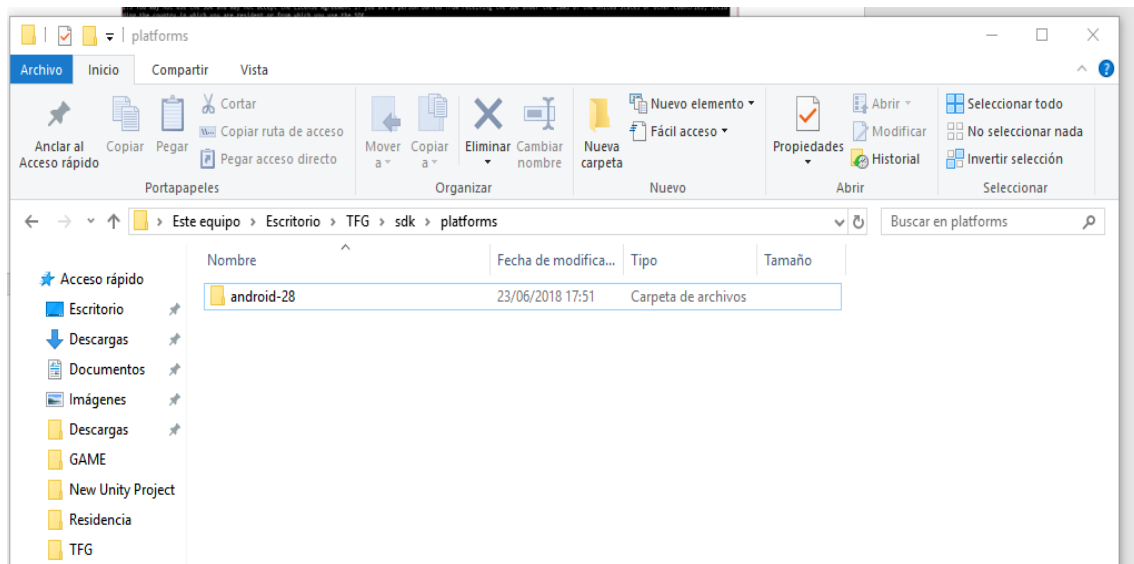
2.8- sdkmanager "extras; google; usb\_driver"

Esto instala el SDK en un directorio llamado "platforms" en el directorio en el que descomprimió la carpeta de herramientas.

Ejemplos: C:\Users\Victo\Desktop\TFG\sdk\platforms.



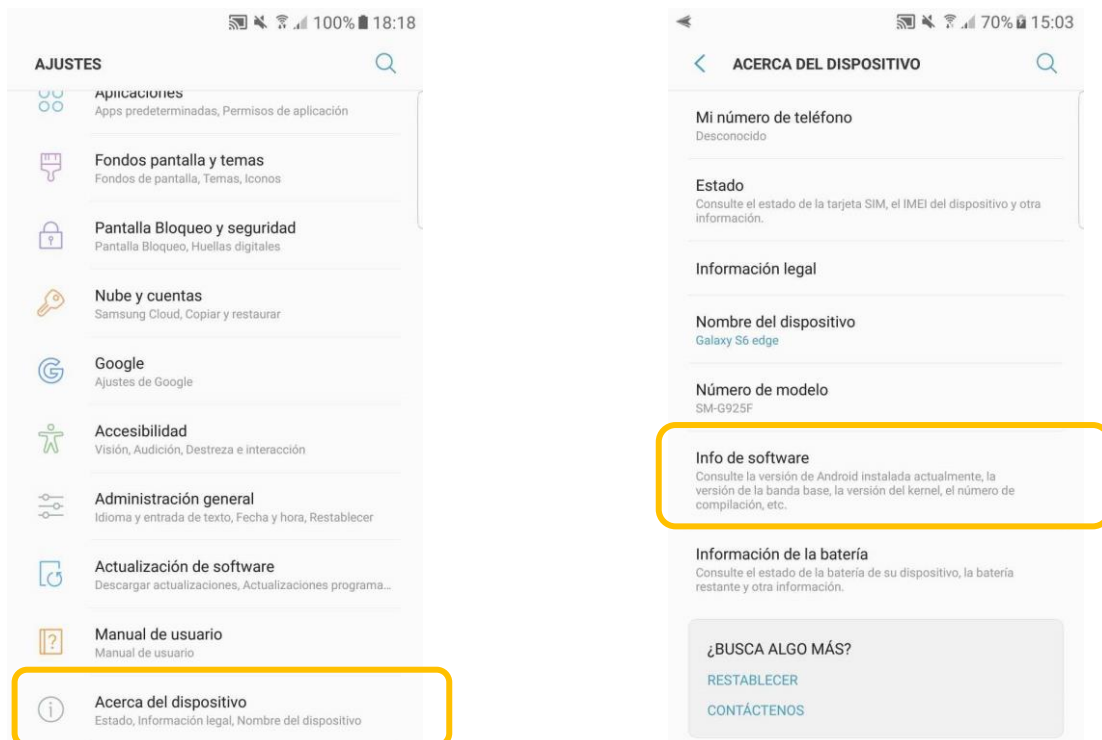
C:\Users\Victo\Desktop\TFG\sdk\extras

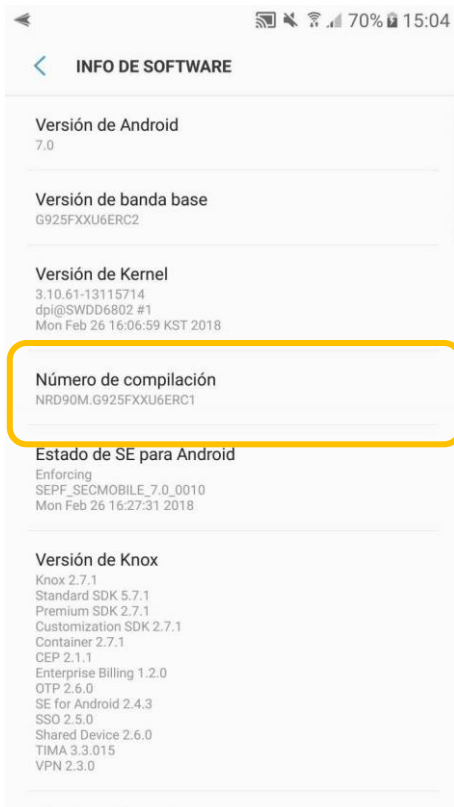


### 3- Habilitar la depuración de USB en el dispositivo.

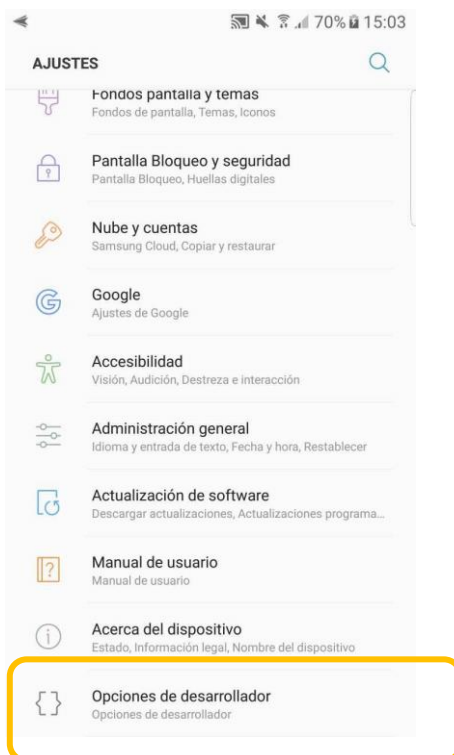
Para habilitar la depuración del dispositivo tendremos que tener activadas las opciones de desarrollador en el terminal. Para ello deberemos de acceder a la siguiente ruta:

Ajustes > Acerca del dispositivo > Información del software > Numero de compilación



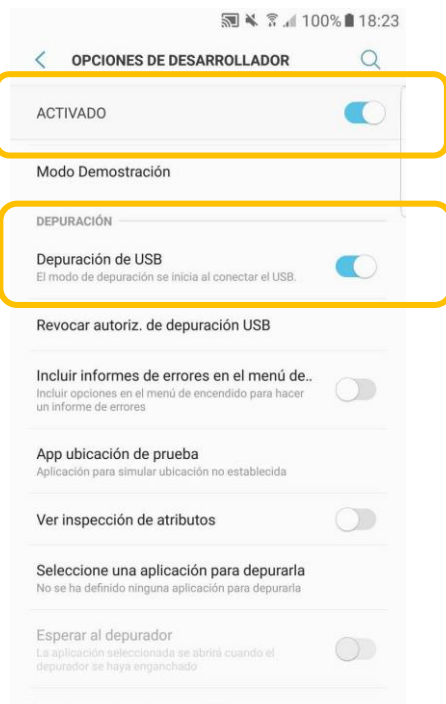


Una vez estemos en esta pantalla si presionamos 7 veces seguidas sobre el número de compilación se activará automáticamente las opciones de desarrollador.



Conecte su dispositivo móvil mediante un cable USB a su computador. Puede que deba de instalar los controladores adecuados a su dispositivo para que se reconozca en su PC.

En el dispositivo móvil, dentro de las opciones de desarrollador deberemos de buscar y activar la opción de depuración de USB para activar el modo de depuración cuando el dispositivo esté conectado a la computadora.



Como ultimo paso a tener en cuenta para la preparación del dispositivo Android es configurar la ruta del SDK de Android en Unity.

La primera vez que crea un Proyecto para Android (o si Unity luego no puede ubicar el SDK), Unity le pide que busque la carpeta en la que instaló el SDK de Android.

Como la instalación del SDK se realizó usando *sdkmanager*, puede encontrar la carpeta en <ubicación de instalación de las herramientas android> \ platforms \ <android sdk folder>.

En nuestro caso:

**c: \ <ubicación de instalación de herramientas de Android> \ platforms \ android-28**

## Google VR en Unity3D

Google posee una serie de herramientas para poder desarrollar mediante su tecnología de realidad virtual en diferentes motores como son *Android Studio*, *Unreal Engine*, *Xcode*(iOS) o *Unity3D*, siendo este último nuestra elección ya que nos proporciona una mayor versatilidad en cuanto al desarrollo de proyectos.



Una vez instalado Unity3D y configurado tanto nuestro dispositivo móvil como el motor gráfico podemos introducirnos en la descarga y estudio de *Google VR*.

Desde el enlace descargaremos la última versión del SDK de Google VR para Unity `GoogleVRForUnity_*.unitypackage`

<https://github.com/googlevr/gvr-unity-sdk/releases>

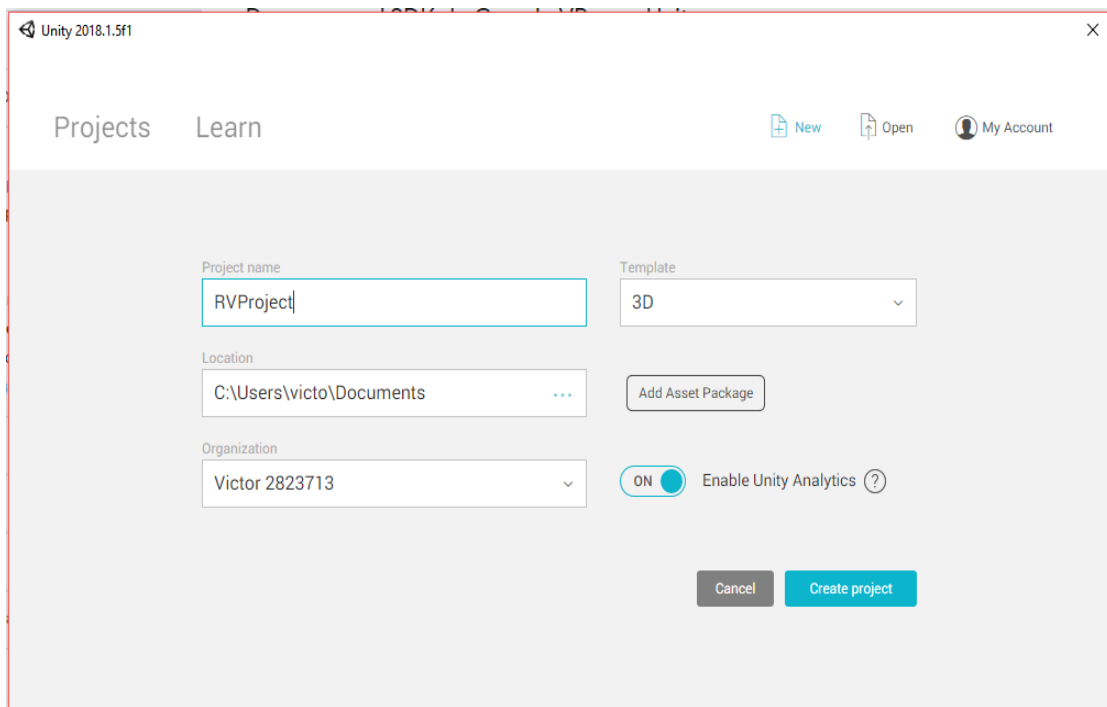
El SDK incluye demostraciones con pequeños ejemplos de *Daydream* y *CardBoard*.

Algunas funcionalidades de la librería de *Daydream* estarán limitadas debido a que el dispositivo Android en el cual estamos desarrollando no soporta algunas de estas funcionalidades como el control de movimiento mediante mandos con giroscopios.

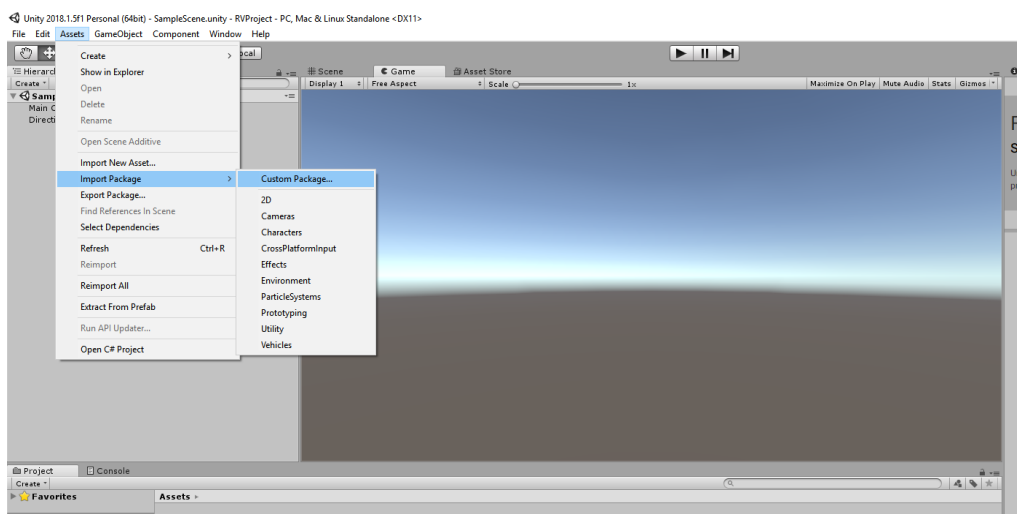
Una vez descargado el SDK deberemos de importar desde Unity el SDK para poder empezar a trabajar con *Google VR*.

## Configuración del proyecto de Unity con Google VR

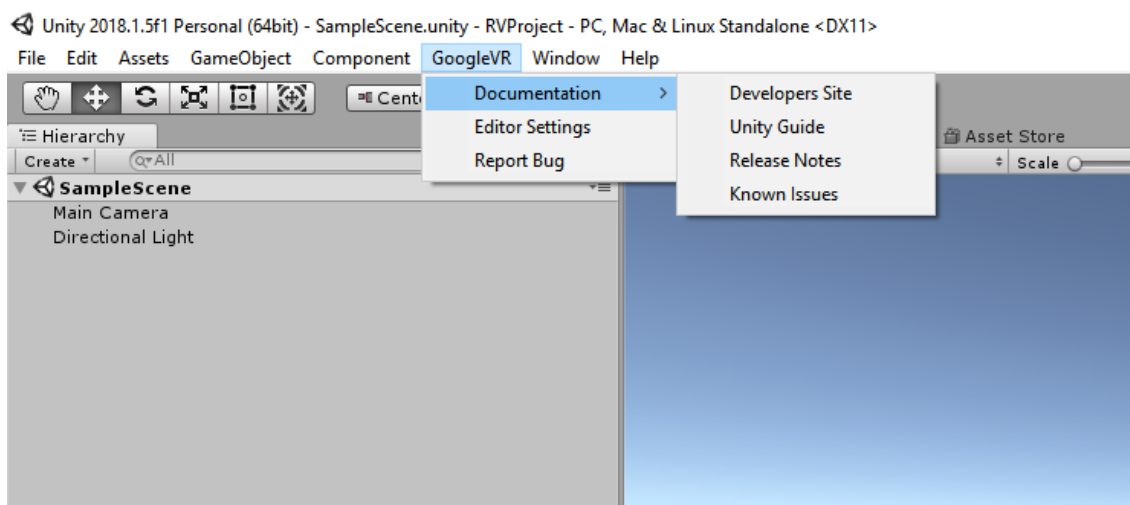
### 1- Abrir un proyecto nuevo 3D



## 2- Seleccionamos **Assets > Import Package > Custom Package.**



Importamos el SDK y una vez instalado veremos como nos aparece en el menú una nueva pestaña de Google VR.

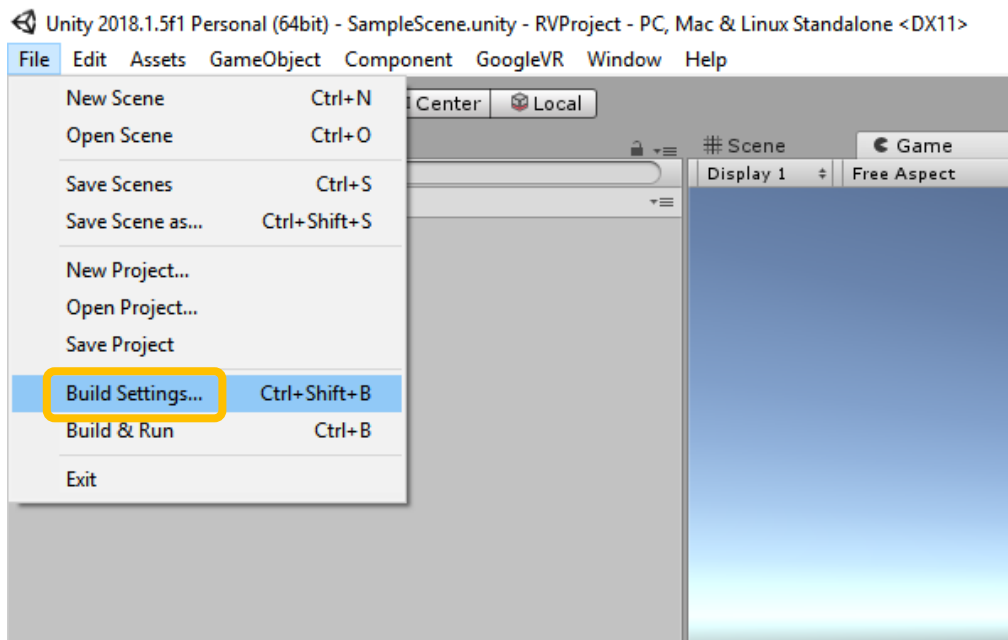


Dentro de este SDK podremos ver algunas demos donde se puede ver funcionalidades que la realidad virtual nos puede aportar.

Una de ellas se llama *HelloVR*, en la cual nos basaremos para nuestro desarrollo ya que esta demo nos proporciona la base de la realidad virtual con el movimiento de cabeza del usuario y la interacción con objetos del entorno virtual.

Configuración de opciones de construcción y opciones de jugador.


1- Selecciona **File > Build Settings**.



2- Selecciona **Android** y haz click en **Switch Platform**

3- En la ventana de **Build Settings** haz click en **Player Settings**.

Una vez pulsado **Player Settings** deberemos de modificar algunos aspectos desde el menú de la derecha **Inspector** indicado en la siguiente tabla:

Setting	Value
Player Settings > XR Settings > Virtual Reality Supported	Enabled
Player Settings > XR Settings > Virtual Reality SDKs	Click + y selecciona CardBoard y Deydreams
Player Settings > Android > XR Settings > Virtual Reality SDKs > Daydream > Positional Head Tracking	Click en el icono  y selecciona <b>Supported</b>
Player Settings > Other Settings > Minimum API Level	Android 4.4 'KitKat' (API level 19) o mayor.

Una vez configurados estos parámetros ya tendremos preparado el proyecto con todo lo básico para empezar con nuestro desarrollo.



## Conceptos sobre el desarrollo del proyecto

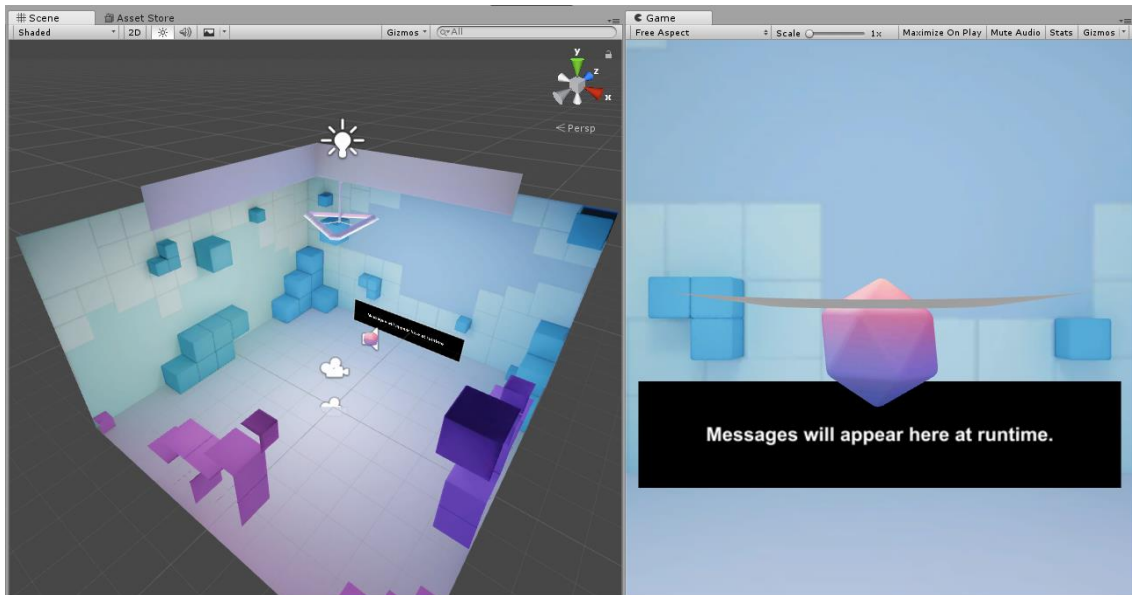
Como se ha mencionado con anterioridad el desarrollo del videojuego se implementará utilizando como base una demo de *Google VR* llamada *HelloVR* implementada para las gafas de realidad virtual de Google *Daydreams*. En esta demo se puede ver cómo se puede hacer uso del entorno con el movimiento de cabeza y haciendo *click* en las esferas que se van generando de forma aleatoria.

En este desarrollo se emula la interacción con el entorno con otro dispositivo Android que nos hará de un mando de las gafas *Daydreams* y nos moveremos por el entorno como si se tratasen de unas básicas *CartonGlass* con *CardBoard*.

## Desarrollo

### Estudio de la demostración básica

Empezaremos el desarrollo de nuestro proyecto viendo y analizando las principales funciones de la demostración *HelloVR*.



El desarrollo de la acción es dentro del un cubo con formas geométricas diferentes.

En dicha escena si ejecutamos el proyecto dándole al Play podremos ver como el usuario permanece estático en el centro de la sala mientras delante de el aparece una forma geométrica.

Presionando la tecla ALT mientras estamos en ejecución podremos movernos libremente por el entorno de varias formas. Si tenemos nuestro dispositivo conectado e *Instant View* abierto, con nuestra cabeza podremos ver el entorno. Otra opción es mover el ratón para observar el cubículo.

Si fijamos la retícula sobre alguna de las esferas veremos como esta se abre y la esfera cambia de color, esto nos indicará que se puede realizar una acción con esta forma.

Por otra parte, presionando el botón del ratón sobre las esferas veremos cómo estas se moverán de forma aleatoria por el cubículo.

Nos centraremos en los apartados de *CubeRoom*, *Treasure* y *Player* donde realizaremos los principales cambios y funciones.

## Cambios para realizar en la demostración

Los cambios que vamos a realizar en la demo serán los siguientes y por orden:

1. Creación de un HUD sencillo.
2. Movimiento del usuario por el entorno.
3. Interacción con el entorno.
4. Sistema de puntuación.
5. Compilación y prueba en nuestro dispositivo móvil.

### Creación de HUD base.

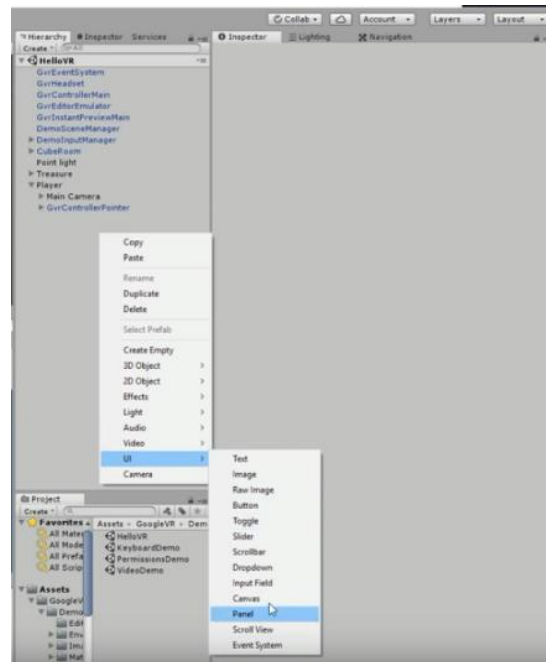
El primer punto que abordamos será la retícula que nos aparece en nuestro dispositivo y nos indica la interacción con el entorno.

Al tener que moverse junto al movimiento del jugador crearemos un nuevo *Canvas*:

**Click derecho -> UI -> Canvas**

Este *canvas* será utilizado como nuestro HUD que almacenará aquellas imágenes o textos que utilicemos par nuestro HUD.

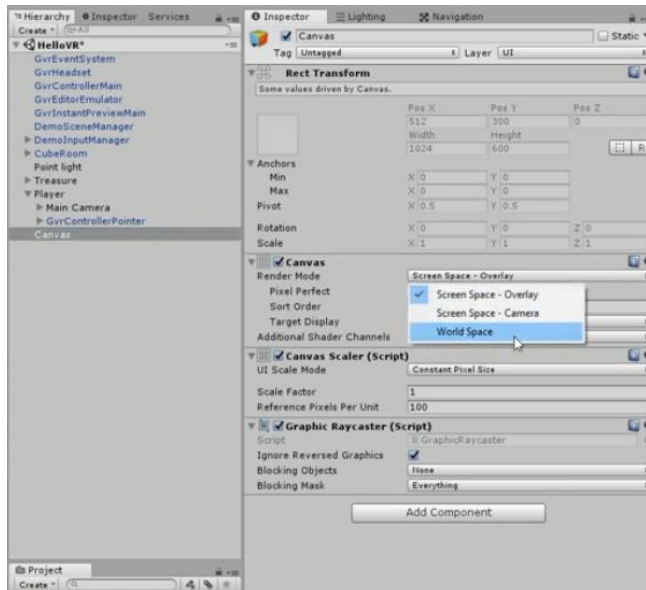
Podemos deseleccionar la opción de *Graphic Raycaster* si deseamos seguir teniendo la funcionalidad de *crosshair* original de apertura y cierre



Ponemos el *canvas* junto al sistema de coordenadas de la *main camera* par un movimiento sincronizado. Para ello cambiamos dentro de las opciones de *canvas* a *WorldSpace*:

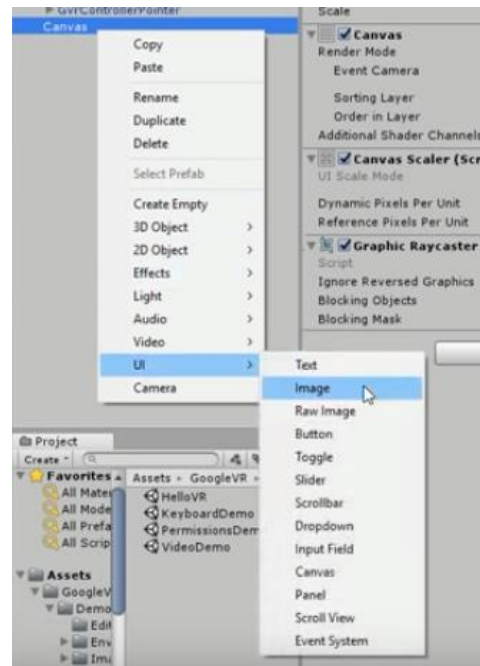
**Canvas -> Render Mode -> World Space**

Reubicamos el *canvas* por delante de la cámara principal por delante de las coordenadas (0,0,0) en este caso y escalamos a una medida adecuada para nuestro entorno.



Seguidamente insertaremos nuestra *crosshair* que actuará como punto de referencia y hará una interfaz mas amigable. Para ello creamos un objeto dentro del *Canvas*:

**UI -> Image**



Importamos una imagen que actúe como *crosshair*. En nuestro caso elegiremos esta:

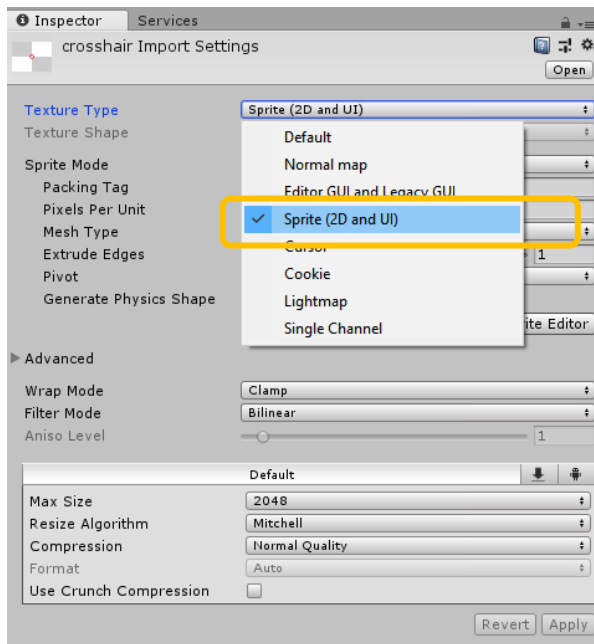


Esta imagen la insertaremos de nuestro proyecto:

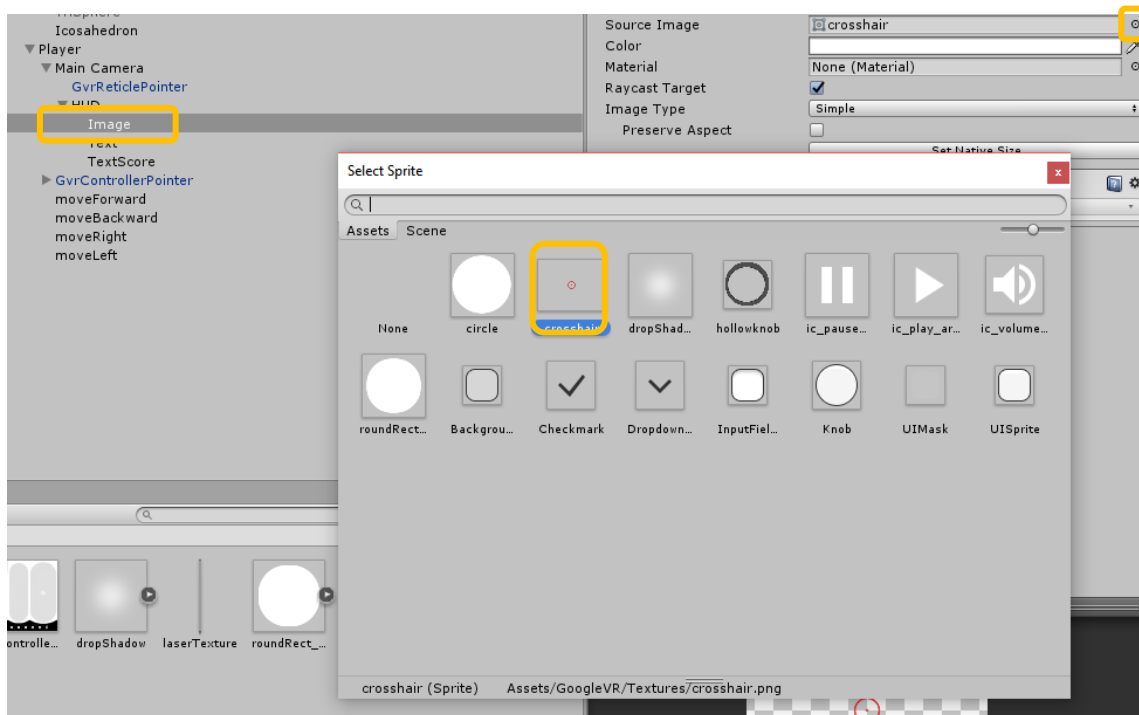
`<ruta de creación del proyecto>\ <proyecto>\ Assets\GoogleVR\Textures`

Dentro de nuestro proyecto transformar el tipo de textura a Sprite (2D and UI) para poder asignar al *canvas* dicha mira.





Con esto nuestra imagen adquiere transparencia en nuestra vista. Ahora seleccionamos la imagen dentro de **Image** → **source Image**

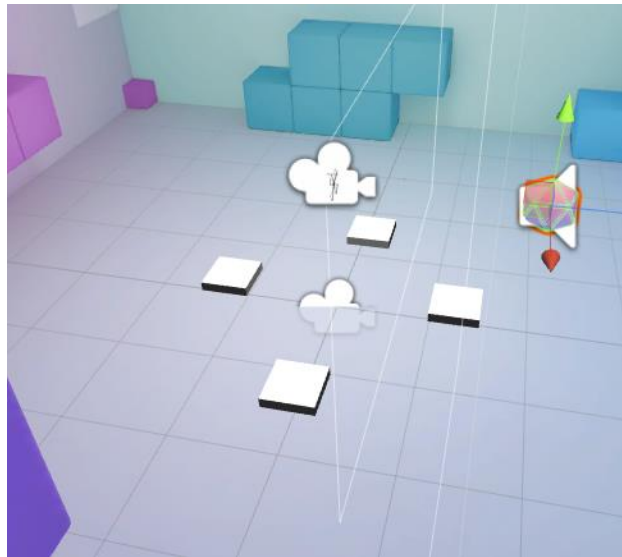


Con esto debemos de cambiar el *canvas* creado dentro de la *main camera* para que el movimiento sea en conjunto.

## Movimiento del usuario

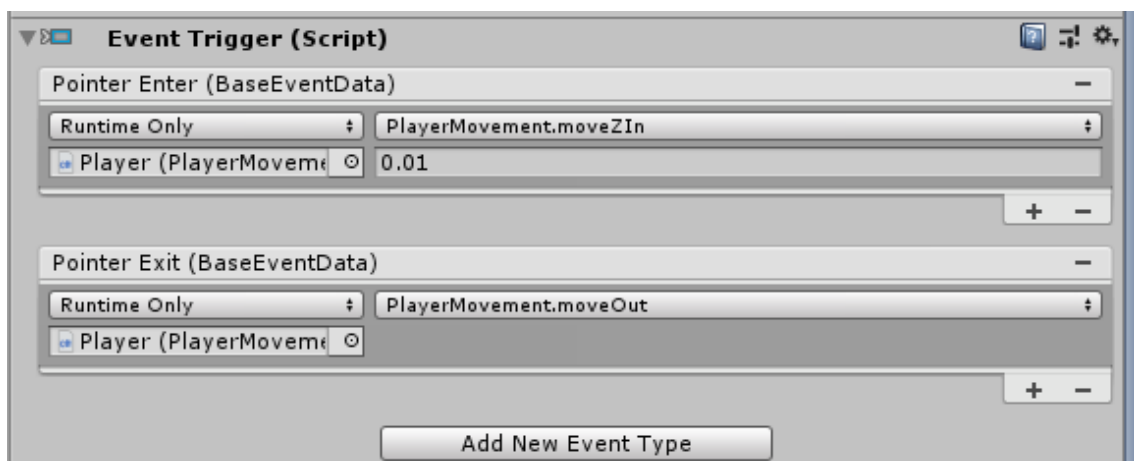
En este apartado se implementa un sistema de movimiento por el entorno sin interacción física con ningún dispositivo externo. Únicamente con nuestra mirada en el entorno virtual.

Para el movimiento crearemos 4 plataformas alrededor del usuario que representarán las 4 principales direcciones de movimiento.



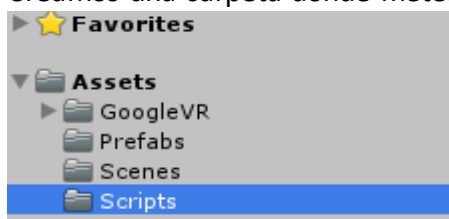
Para ello crearemos los nuevos objetos *Cube* que redimensionaremos y colocaremos a la altura del suelo.

Inicialmente obtendremos una serie de cubos que no tendrán una interacción con nuestro *crosshair* creado anteriormente. Si deseamos dicha interacción visual deberemos de añadir un *Event Trigger* con un *Pointer Enter* y otro *Pointer Exit* que harán que según nuestra mirada en dicho cubo nos desplazemos en la dirección adecuada.



Cada uno de estos eventos dentro del *Event Trigger* tienen asociado un *script* de movimiento.

Creamos una carpeta donde meteremos todos los scripts generados.



Dentro de la carpeta creamos un *script* con nombre *PlayerMovement*:

**Click botón derecho ratón -> Create -> C# Script**

El editor de *scripts* puede ser el que viene por defecto con Unity o aquel que os resulte mas cómodo para el desarrollador.

```
PlayerMovement.cs

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMovement : MonoBehaviour {

    // Use this for initialization

    public Transform playerTf;

    Vector3 playerVector;

    void Start () {

        }

    // Update is called once per frame
    void FixedUpdate () {
        playerTf.position += playerVector;
    }

    public void moveXIn(float x) {

        playerVector = new Vector3(x,0,0);

    }

    public void moveZIn(float z)

    {
        playerVector = new Vector3(0, 0, z);

    }

    public void moveOut() {

        playerVector = new Vector3(0, 0, 0);

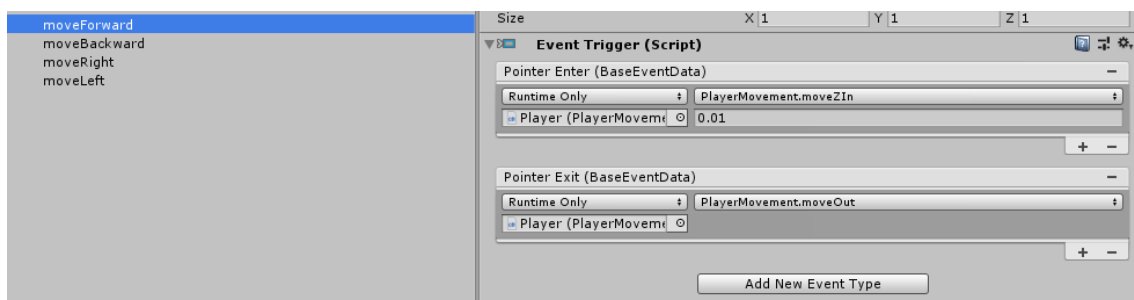
    }

}
```

Como se puede observar se han creado 3 métodos diferentes para el movimiento en función de la posición del usuario donde:

- **FixedUpdate():** En este método la posición del usuario cambiará cada *frame* en función al vector *playerVector*.
- **moveXIn(float x):** Como el propio método indica el vector de movimiento se verá afectado por la posición del eje X cuando la mira esté sobre la plataforma.
- **moveZIn(float Z):** Debido a que trabajamos sobre un eje de coordenadas donde los ejes X y Z determinarán el movimiento delante/detrás e izquierda/derecha este método hará el mismo funcionamiento que *moveInX()* pero en el eje Z.
- **moveOut():** Cuando salgamos de nuestra plataforma se activará este método que hace que el usuario quede quieto en la posición actual.

Una vez creado el script deberemos designarlo a nuestro objeto y asignar a los eventos creados de *Event Trigger* de cada cubo los métodos apropiados con su variable de entrada correspondiente que nos indica la distancia a recorrer en cada instante.



Finalmente, si tenemos instalado *Instant Preview* en nuestro dispositivo móvil y conectado a nuestro PC una vez le demos al *play* podremos probar desde nuestro *smartphone* todos los cambios que hemos realizado y haremos.

### *Interacción con el entorno (Disparo del jugador)*

El siguiente paso que implantaremos será un sistema de disparos que accionaremos desde el otro dispositivo móvil que utilizaremos como mando de las gafas de realidad virtual *Daydreams*.

Primero instalaremos la aplicación móvil *The Controller Emulator* en nuestro *smartphone* que emulará el mando de interacción descargando desde este enlace oficial de *Google* el *apk*.

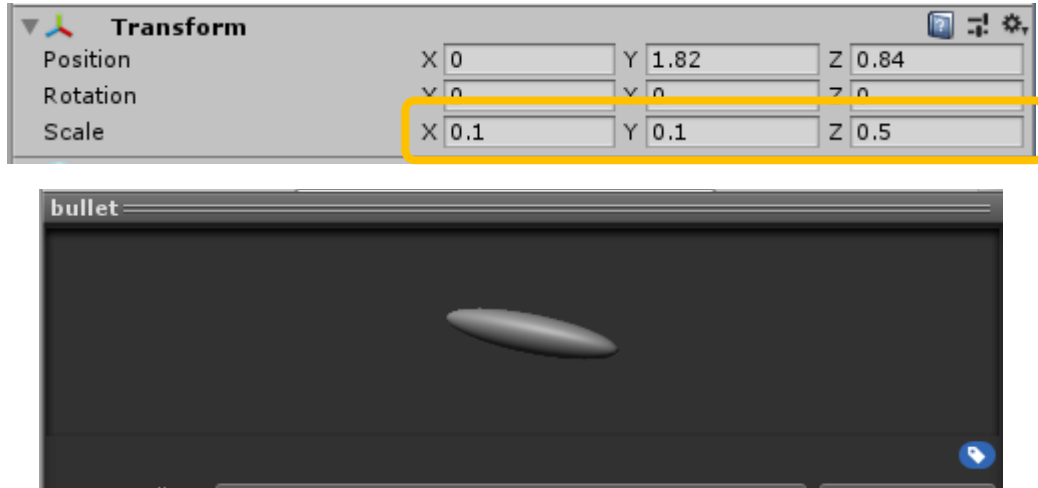
<https://developers.google.com/vr/daydream/controller-emulator>

La conexión con nuestro dispositivo móvil que ejecutará la aplicación se hace automáticamente mediante la conexión *bluetooth*.

## Creación de la bala

Volviendo a Unity deberemos de crear una instancia para hacer que nuestro objeto que haga de bala se mueva de forma uniforme una vez accionemos la interacción.

Primero crearemos una esfera y modificaremos para darle un aspecto de bala.



Crearemos un nuevo script que haga que nuestra bala se mueva.

```

                    BulletMovement.cs

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

    public class BulletMovement : MonoBehaviour {

public float speed = 20.0f;
float elapsed;
    // Use this for initialization
    void Start () {
        elapsed = 0;
    }

    // Update is called once per frame
    void FixedUpdate() {
        elapsed += Time.fixedDeltaTime;

        transform.position += transform.forward * speed * Time.fixedDeltaTime;

        if (elapsed >= 10)
        {
            GameObject.Destroy(gameObject);
        }
    }
}
    
```



Como podemos observar en nuestro script nuestra principal función vendrá por el movimiento de la bala:

```
transform.position += transform.forward * speed * Time.fixedDeltaTime;
```

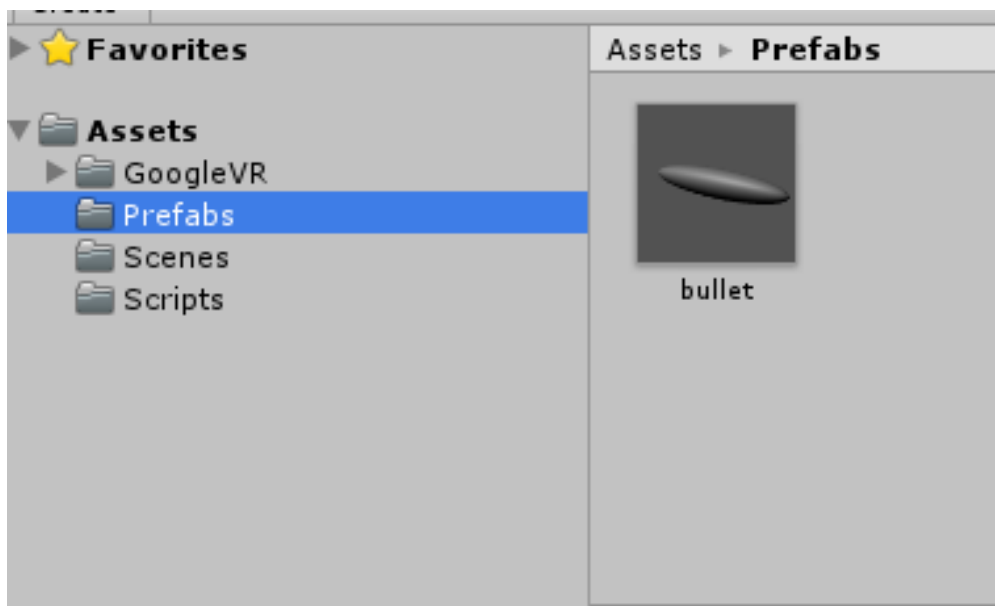
Cambiaremos la posición de la bala en función del vector de dirección (`transform.forward`), la velocidad (`speed`) y el tiempo (`Time.fixedDeltaTime`)

Otra de las características que se han desarrollado es el tiempo que nuestra bala está en escena:

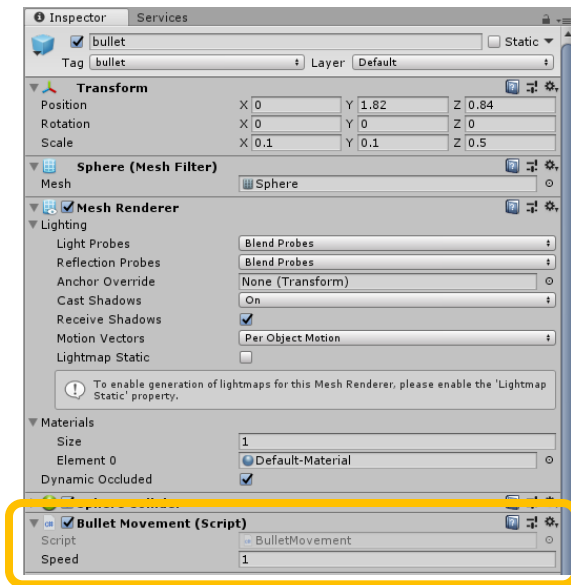
```
elapsed += Time.fixedDeltaTime;  
  
if (elapsed >= 10)  
{  
    GameObject.Destroy(gameObject);  
}
```

Trabaja con una variable que actuará de medidor de tiempo. Cuando esta variable supere los 10 segundos la bala se destruirá.

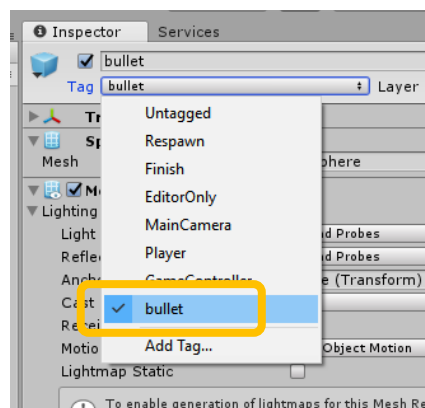
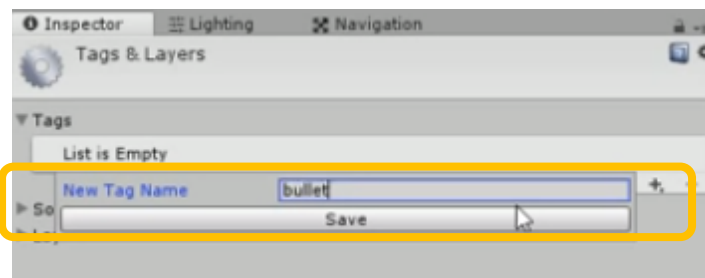
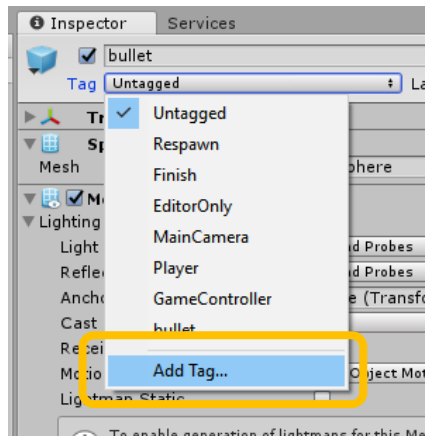
Siguiendo con el funcionamiento de la bala, crearemos una nueva carpeta llamada *prefabs* donde añadiremos nuestra bala.



A la cual asignaremos nuestro script de *BulletMovement.cs*



Además, crearemos un nuevo *Tag* y se lo asignaremos a nuestra bala para poder llamarla posteriormente al interactuar con el mando controlador.



## Sistema de disparos del jugador

Por otra parte, crearemos otro script que actuará como accionador de la bala por el usuario con el controlador que tengamos.

```
PlayerShoot.cs  
  
using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class PlayerShoot : MonoBehaviour {  
  
    public GameObject prefab;  
    public Transform head;  
  
    // Use this for initialization  
    void Start () {  
  
    }  
  
    // Update is called once per frame  
    void Update () {  
        if (GvrControllerInput.ClickButton)  
        {  
            GameObject.Instantiate(prefab, head.position, head.rotation);  
        }  
    }  
}
```

En este script simplemente se efectuará la acción de disparo mediante:

```
GvrControllerInput.ClickButton
```

Esta instrucción esta creada para los controladores de las gafas de realidad virtual de *Daydreams*, esta instrucción la ejecutará nuestro smartphone que haga de mando.

Seguidamente se ejecutará la instrucción:

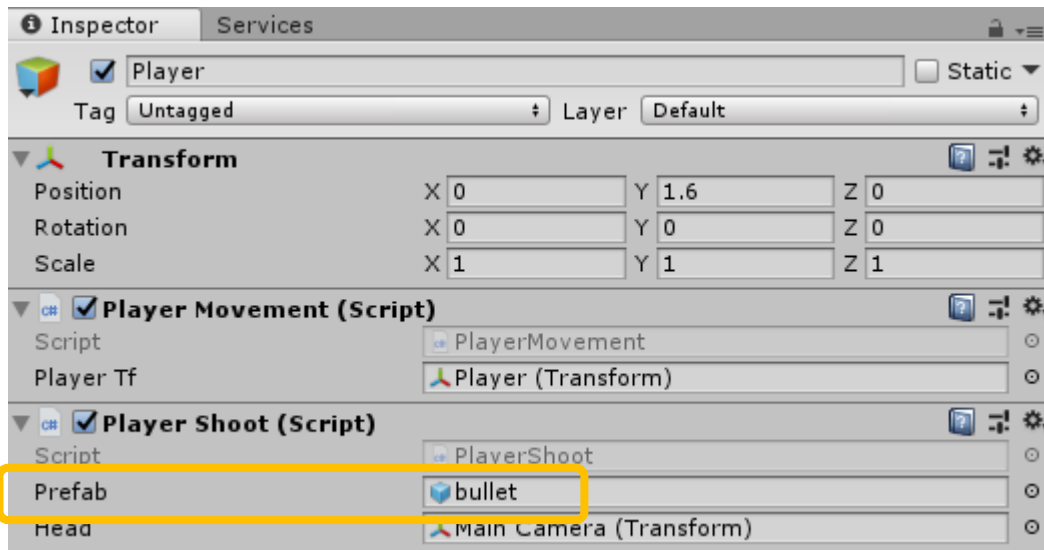
```
GameObject.Instantiate(prefab, head.position, head.rotation);
```

Que se compondrá del *GameObject prefab* y la posición y rotación de nuestra cabeza.

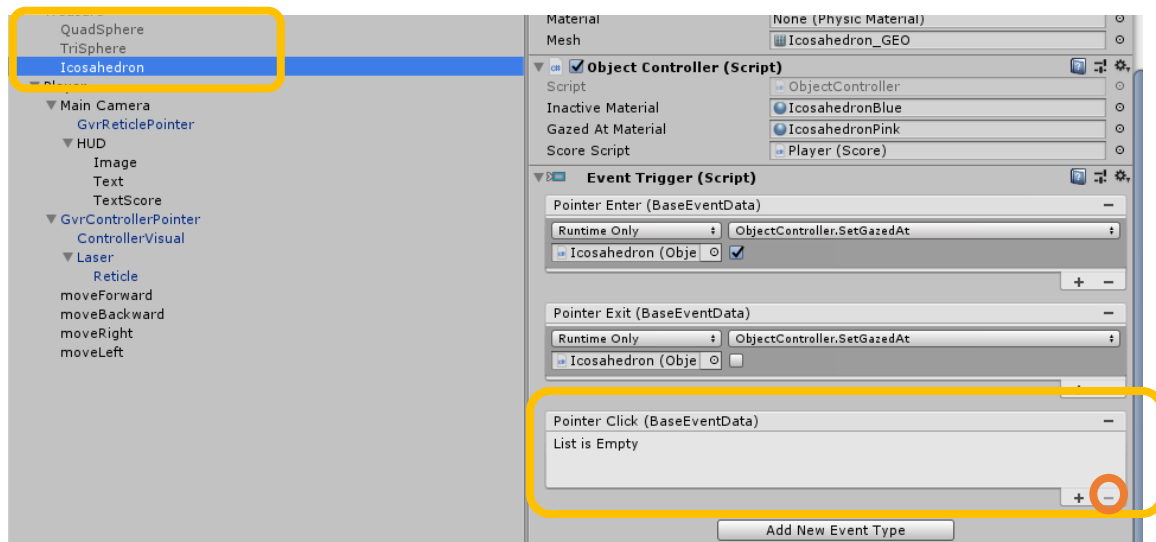
Si opcionalmente queremos probar la interacción desde nuestro computador podemos añadir:

```
if (GvrControllerInput.ClickButton || Input.GetMouseButtonDown(0))  
{  
    GameObject.Instantiate(prefab, head.position, head.rotation);  
}
```

Una vez terminado el *script* se lo asignaremos a nuestro *Player* y dentro del *Prefab* asignamos nuestra bala que anteriormente creamos.



Originalmente las formas geométricas que aparecen al interactuar con nuestro *click* cambian de posición de forma aleatoria. Para cambiar esta funcionalidad deberemos de modificar los polígonos quitando el *EventTrigger* de *PointerClick* y dejándolo vacío:



La función que queremos asignar al hacer *click* o interactuar con nuestro mando será que la forma geométrica se mueva de forma aleatoria al impactar nuestra bala. La creación anterior del *tag bullet* sobre nuestra bala nos ayudará a dicho propósito.

Para ejecutar dicha acción modificaremos el script *ObjectController.cs* creando un nuevo método que respalde dicha función:

```
ObjectController.cs  
  
[...]  
  
private void OnCollisionEnter(Collision collision)  
    {  
        if (collision.gameObject.tag == "bullet") {  
            TeleportRandomly();  
            GameObject.Destroy(collision.gameObject);  
        }  
    }  
  
    public void TeleportRandomly() {  
        // Pick a random sibling, move them somewhere random, activate them,  
        // deactivate ourself.  
        int sibIdx = transform.GetSiblingIndex();  
        int numSibs = transform.parent.childCount;  
        sibIdx = (sibIdx + Random.Range(1, numSibs)) % numSibs;  
        GameObject randomSib = transform.parent.GetChild(sibIdx).gameObject;  
  
        // Move to random new location ±100º horizontal.  
        Vector3 direction = Quaternion.Euler(  
            0,  
            Random.Range(-90, 90),  
            0) * Vector3.forward;  
        // New location between 1.5m and 3.5m.  
        float distance = 2 * Random.value + 1.5f;  
        Vector3 newPos = direction * distance;  
        // Limit vertical position to be fully in the room.  
        newPos.y = Mathf.Clamp(newPos.y, -1.2f, 4f);  
        randomSib.transform.localPosition = newPos;  
  
        randomSib.SetActive(true);  
        gameObject.SetActive(false);  
        SetGazedAt(false);  
    }  
}
```

El método creado *OnCollisionEnter* tendrá en cuenta la colisión entre nuestra bala y el objeto al que asignemos el script. Para ello si la colisión se efectúa con un objeto con el tag <<bullet>> se lanzará la función *TeleportRandomly()* que mueve las formas geométricas por de forma aleatoria dentro de un espacio limitado.

```
if (collision.gameObject.tag == "bullet") {  
    TeleportRandomly();  
}
```

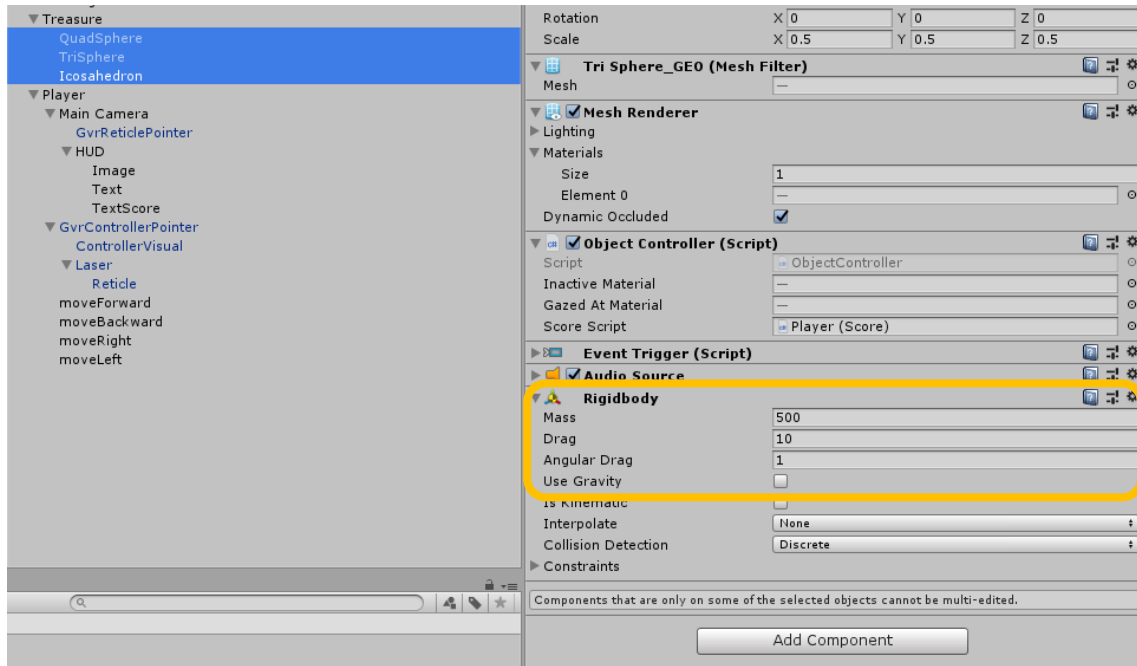
Además, con la llamada posterior la bala desaparecerá al colisionar con el otro objeto:

```
GameObject.Destroy(collision.gameObject);
```

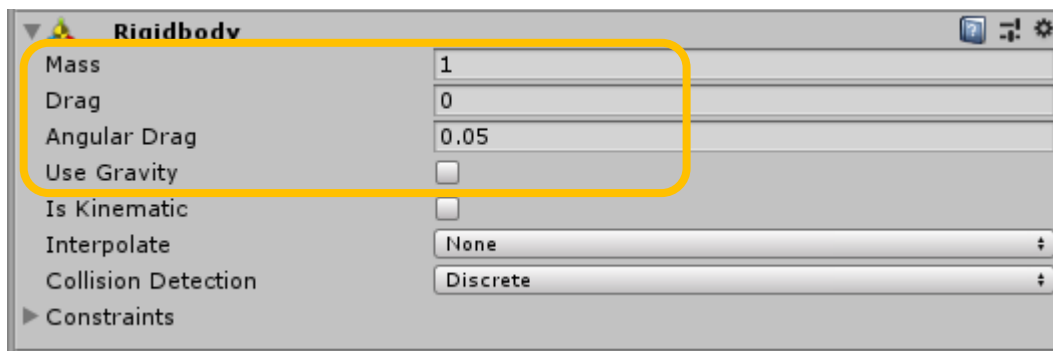


Antes de probar nuestros scripts deberemos de modificar los cuerpos geométricos y nuestra bala para que las colisiones se efectúen de forma eficiente.

Unity trabaja las colisiones con los cuerpos rígidos así que seleccionando las tres figuras añadiremos un *RigidBody* donde asignaremos una masa y un rozamiento altos y deshabilitando la opción de *Use Gravity*.



A nuestra bala le asignaremos otro *RigidBody* pudiendo dejar las características por defecto deshabilitando también la opción *Use Gravity*:



Una vez realizados estos pasos nuestro sistema de colisiones estará realizado y podremos probarlo de forma adecuada en nuestro entorno.

## Sistema de puntuaciones

Este apartado añadiremos a nuestro HUD un sistema de puntuación que nos servirá de referencia para observar que al impactar una bala sobre un cuerpo geométrico este desaparece junto a la bala y aumenta una puntuación en nuestra vista de usuario.

Primero crearemos un nuevo script llamado *Score.cs* para el sistema de puntuación:

```


Score.cs


using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Score : MonoBehaviour {

    // Use this for initialization

    public Text txt;
    public int score;
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        txt.text = "" + score;
    }

    public void UpdateScore(int deltaScore) {
        score += deltaScore;
    }
}

```

Añadimos `using UnityEngine.UI;` ya que utilizaremos elementos gráficos del *canvas*. El script se compone de un método *UpdateScore(int deltaScore)* que irá actualizando la puntuación por cada interacción entre objetos:

```
score += deltaScore;
```

y el *Update()* que reflejará esos cambios en nuestra pantalla.

```
txt.text = "" + score;
```

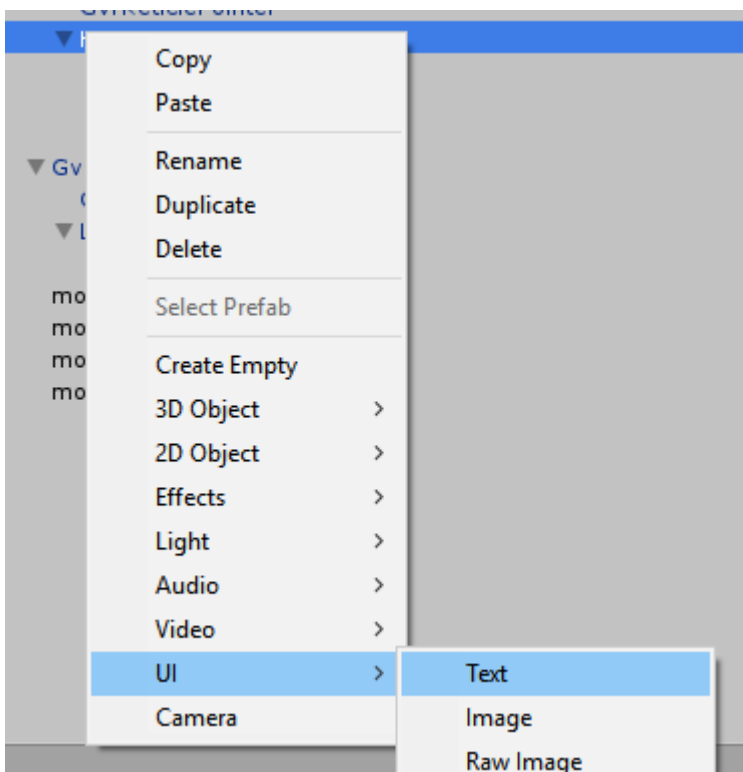
Ahora volvemos al script *ObjectController.cs* para que registre los cambios de la puntuación.

Modificamos el anterior método *OnCollisionEnter* para que registre el cambio de puntuación:

```
ObjectController.cs  
  
public Score scoreScript;  
[...]  
private void OnCollisionEnter(Collision collision)  
{  
    if (collision.gameObject.tag == "bullet") {  
        TeleportRandomly();  
        GameObject.Destroy(collision.gameObject);  
        scoreScript.UpdateScore(100);  
    }  
}
```

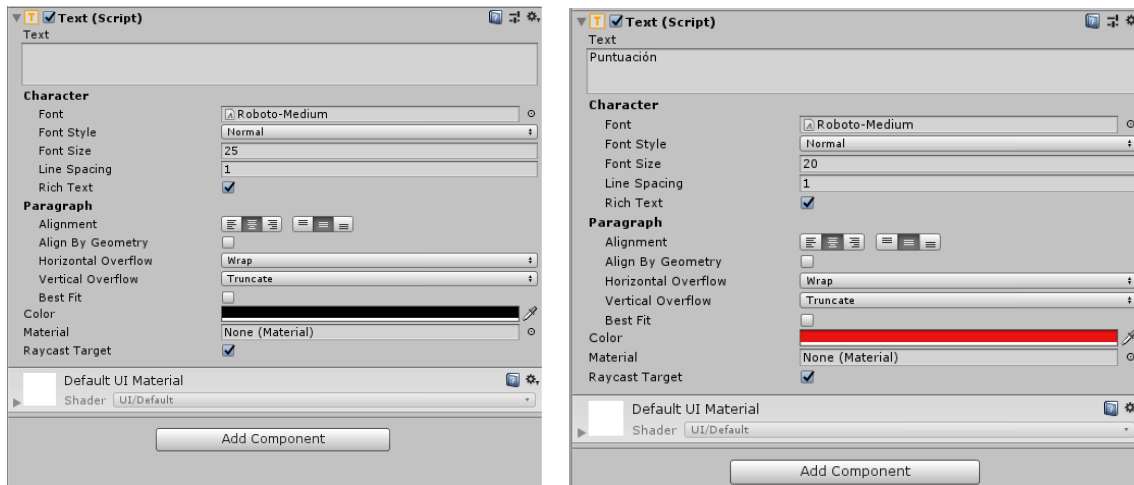
Llamamos a un *scoreScript* de tipo *Score* que anteriormente hemos creado. Una vez hecho esto simplemente llamaremos a la función *UpdateScore* para que aumente dicha puntuación.

Terminado nuestros *scripts* y antes de asignar a nuestro Player el script del *Score* crearemos un par de objetos gráficos de tipo *Text* que albergaran la puntuación.

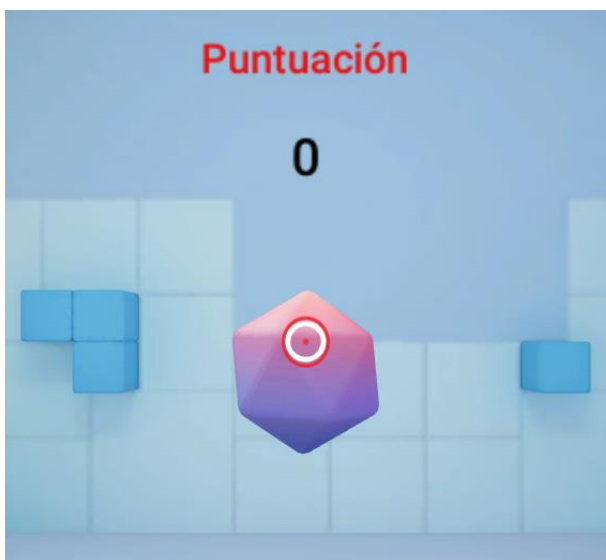




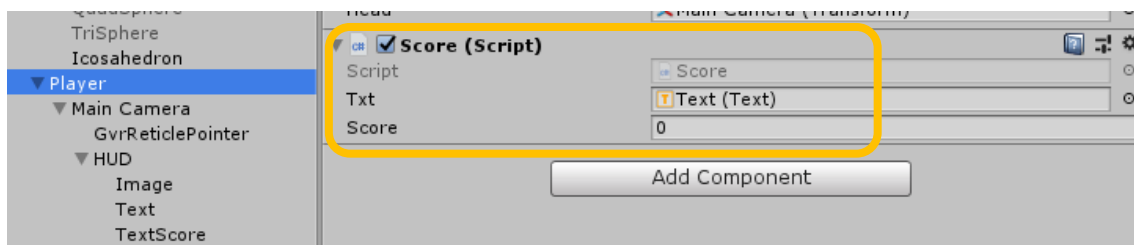
Esta puntuación la podemos regular y modificar a nuestro gusto para colocarla en nuestro entorno.



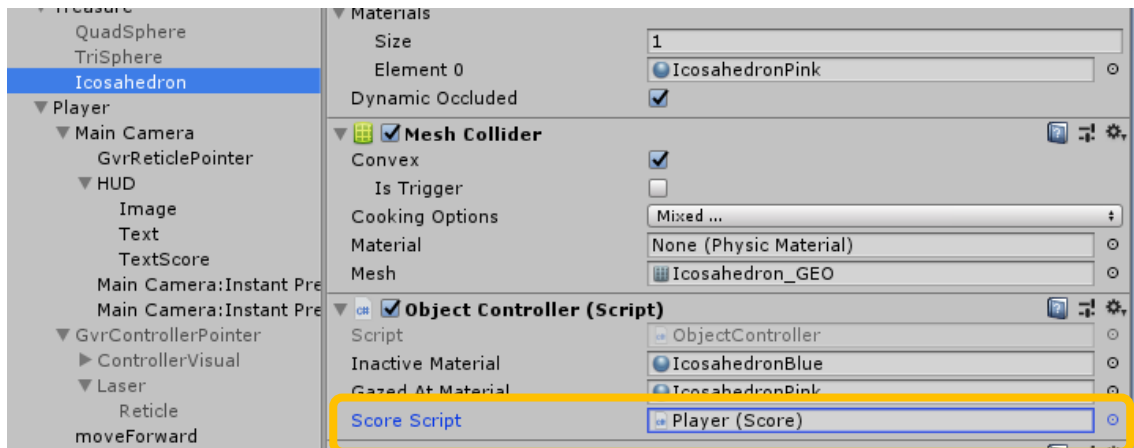
Con esta configuración nos quedaría una vista tal como esta:



Asignamos el script a nuestro *Player* y dentro de este *Score (script)* metemos nuestro *Txt* de puntuación.



Seguidamente en cada una de nuestras formas geométricas asignaremos nuestro *Player* dentro del *Score Script*.

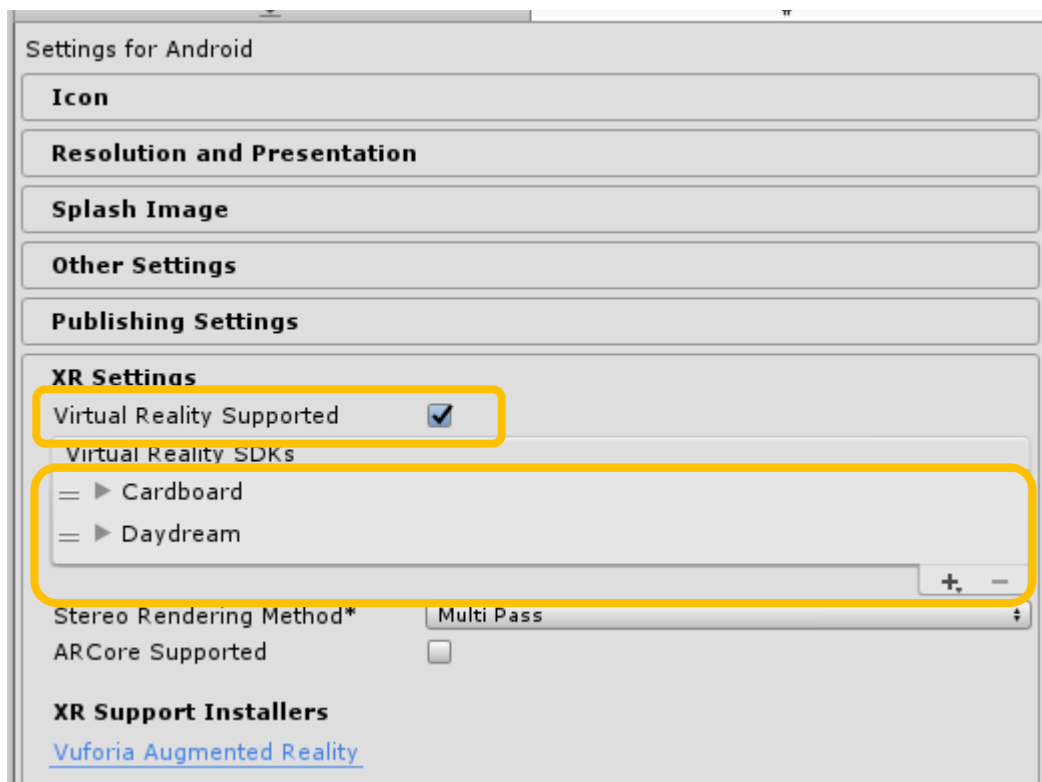


Con esto habremos terminado nuestro sistema de puntuación.

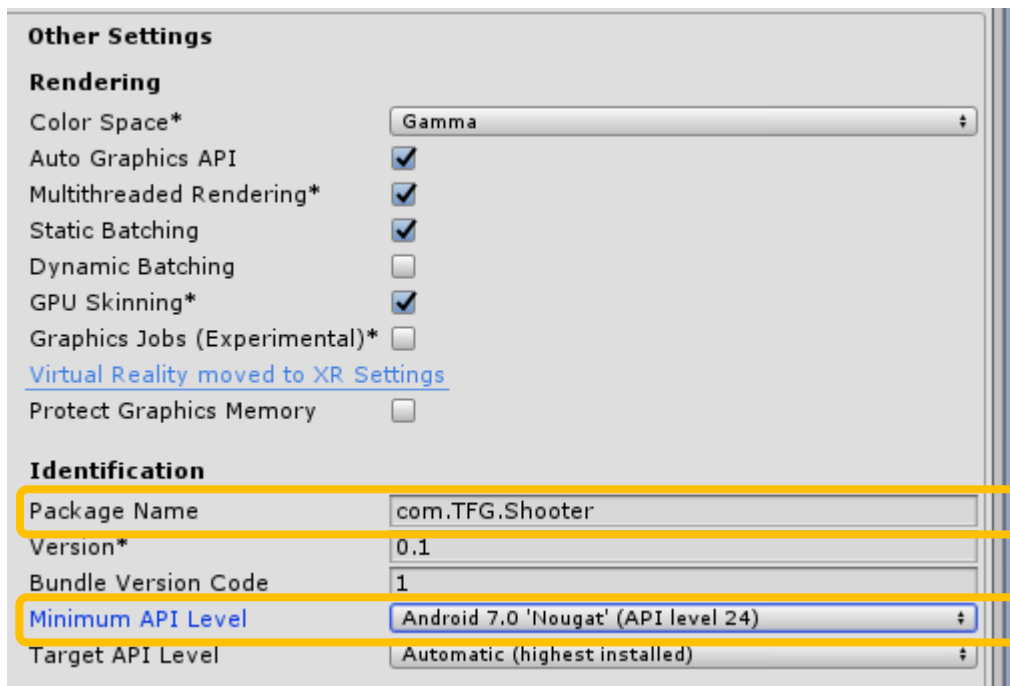
## Exportación en nuestro dispositivo móvil

Para poder exportar de forma adecuada tendremos que tener en cuenta una serie de características.

Dentro de: **File > Build Setting... > Player Settings > XR Settings** nos aseguraremos que tenemos las marcadas la opción de *Virtual Reality Supported* y añadidas las opciones de *CardBoard* y *Daydreams*.



En *Other Settings* tener en *Minimum API Level* una versión de *Android 7.0* o mayor y darle un nombre a nuestro desarrollo.



Con esto ya podemos darle a *Build* y pasarnos a nuestro dispositivo el *apk* generada por Unity.

Si todo ha funcionado bien, pasada la aplicación al *smartphone* y una vez instalada el *apk* en el dispositivo, si ejecutamos la aplicación y tenemos abierto *The Controller Emulator* en el otro *smartphone* se conectará de forma automática y podremos efectuar nuestros disparos.