
# Trabajo de Fin de Grado

---

# Diseño e implementación de un prototipo de videojuego de Realidad Virtual

*Implementation and design of a Virtual Reality video game prototype*

Jorge Octavio Blanchard Cruz

---

La Laguna, 3 de marzo de 2016

D. **Pedro Antonio Toledo Delgado**, con N.I.F. 45.725.874-B profesor ayudante adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

**C E R T I F I C A**

Que la presente memoria titulada:

*"Diseño e implementación de un prototipo de videojuego de Realidad Virtual."*

ha sido realizada bajo su dirección por D. **Jorge Octavio Blanchard Cruz**, con N.I.F. 78.713.338-R.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 3 de marzo de 2016.

# Agradecimientos

En primer lugar, quería mostrar mi agradecimiento al tutor de este Trabajo de Fin de Grado, D. Pedro Toledo, quien ha sido mi guía. Además, quiero destacar también la calidad humana y paciencia demostrada en su apoyo.

Gracias de todo corazón por la ayuda prestada, ha sido una experiencia muy agradable y enriquecedora que ya forma parte de mi vida.

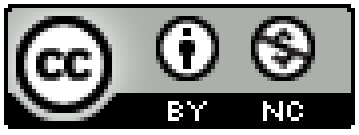
En segundo lugar, me gustaría agradecer a mi familia y a mis amigos, por todo el apoyo que he recibido de su parte y por esas largas horas de testeo del juego desarrollado en este proyecto.

Gracias a María, mi pareja, que me ha prestado su ayuda de forma activa en la elaboración de este proyecto y, sobre todo, en las partes traducidas al inglés y en la preparación de la pronunciación de las conclusiones para la lectura de la memoria.

Finalmente, agradezco a la escuela por todos los buenos momentos que he vivido allí y que me ha formado tanto como ingeniero informático como personalmente. Y en especial mención a los profesores Dña. Margarita María Rivero Álvarez, D. Casiano Rodríguez León y Dña. Marisela López Curbelo.

Todo mi cariño para vosotros.

# Licencia



# Resumen

Este proyecto de fin de grado ha tenido como objetivos, por una parte, investigar las nuevas tecnologías de interacción persona-computador con la función de transportar al usuario a una experiencia más profunda y cercana a la realidad en el ámbito jugable. Por otra parte, y llevados por la pasión hacia los videojuegos, crear un prototipo que consiste en un juego en primera persona (First-Person game) con los métodos y herramientas de hoy en día.

En los últimos tiempos, el mercado de hardware ha estado aportando una amplia gama de tecnologías basadas en novedosas formas de interacción del usuario con la computadora. En este proyecto se ha indagado específicamente en el dispositivo de Realidad Virtual Oculus Rift DK2 (Development Kit 2), un prototipo de Head-Mounted Display (HMD). Esta tecnología está siendo desarrollada bajo la dirección de John Carmack (Cofundador de ID Software), por la empresa Oculus VR fundada en 2012 por Palmer Luckey. A pesar de estar en proceso de investigación, ya contamos con versiones beta para los desarrolladores de software.

La implementación de este proyecto se distingue en varias líneas de desarrollo: Siendo aún Oculus Rift una versión previa a la comercial, e investigando sobre los problemas del desarrollo de estas tecnologías, se ha implementado un controlador del personaje jugable en primera persona que añade una aceleración a los movimientos de este, con el objetivo de suavizar dichos movimientos. Con ello se ha mejorado la experiencia y adaptación del usuario al dispositivo. Por otro lado, se ha elaborado lo que podría ser una nueva fórmula de interacción con los videojuegos al separar el foco de visión del jugador de la acción que se efectúa y se ha integrado un avatar jugable junto con las animaciones básicas adaptadas a la tecnología de Realidad Virtual.

Concluyendo con la implementación, los escenarios y objetos interactuables del videojuego han sido diseñados específicamente para lograr mayor interacción entre el jugador y el controlador Oculus Rift.

**Palabras clave**: videojuegos, interacción persona-computador, Realidad Virtual, Head-Mounted Display, HMD, Oculus Rift, Oculus VR, First-Person game.

# Abstract

This End of Degree Project has had as a goal, firstly, to investigate new human-machine interaction technologies with the purpose of bringing the user to a deeper, closer to reality experience within a playable environment. Secondly, and being driven by the passion for video games, to create a first person game prototype involving nowadays new methods and tools.

Over the last years, hardware market has been providing with a wide range of technologies, based on novel ways of human-computer interaction. This project has specifically looked into the Virtual Reality device Oculus Rift DK2 (Development Kit 2), a Head-Mounted Display (HMD) prototype. This technology is being developed under the guidance of John Carmack (ID Software co-founder) by Oculus VR, a company founded in 2012 by Palmer Luckey, and despite this device being still in research, there are already beta versions for developers.

There have been defined different lines of development for the implementation of this project:

Since Oculus Rift is an early version of the device and with the intention of investigating some of the development problems, it has been implemented a first person playable character controller, which adds an acceleration to the character's movements in order to soften them. This has improved the user's adaptation to the device and therefore achieved a better experience.

In addition, it has been created a new form of video game interaction by the way the focus of the player's vision may or may not differ from that of the action, and it has been integrated a playable avatar alongside basic animations adapted to the Virtual Reality technology.

Concluding with the implementation, the video game's settings and interactive objects have been specifically designed to accomplish a greater interplay between the user and the Oculus Rift controller.

**Key words**: video games, human-machine interaction, Virtual Reality, Head-Mounted Display, HMD, Oculus Rift, Oculus VR, first-person game.

# Índice General

# Índice de figuras

# Índice de tablas

# Capítulo 1. Introducción

## 1.1 Contexto

### 1.1.1 Interfaces persona-computador

Para entender la actualidad de los interfaces persona-computador tenemos que repasar brevemente su historia. Desde la invención de la primera computadora ENIAC en 1945, los ingenieros se han percatado de la importancia de poder comunicarse con las máquinas de forma fácil y eficaz. Con ENIAC, para efectuar diferentes operaciones era necesario conectar y desconectar miles de cables, que hacía que se demorara en semanas.

Durante años, los sistemas evolucionaron muy poco: tarjetas perforadas, impresoras, etc. Solo hubo un gran salto con la aparición del teclado y el monitor, lo que iba a marcar el inicio de la informática tal como la entendemos hoy en día. En 1972 se inventaría el ratón con el primer personal-computer (PC) y en la década de los 80 nacería el primer ordenador con interfaz gráfica, el Xerox Star. Así comenzaría la industria de los videojuegos. Desde ese momento dichas interfaces, junto con el monitor como salida, teclado y ratón como entrada, han sido los más habituales en cuanto a comunicación persona-computador.

#### Tipos de interfaces (1)

Con la expansión industrial tanto del PC como de los videojuegos se fueron desarrollando múltiples tipos de interfaces:

a) Las interfaces de voz han evolucionado desde los antiguos contestadores automáticos hasta las últimas tecnologías de asistentes virtuales e inteligencia artificial como Cortana (Microsoft), Siri (Apple) y Google Now.

b) El uso de las interfaces táctiles en la actualidad está muy expandido, ya que todos tenemos un Smartphone en el bolsillo los cuales están provistos de pantallas táctiles para navegar en su interfaz gráfica. Las primeras máquinas con este concepto fueron las tabletas digitalizadoras, que constaban de una placa y un lápiz para introducir datos como si fuera papel y lápiz.

c) También existen interfaces destinados a discapacitados. Hoy en día las empresas tienen más en cuenta estos colectivos y desarrollan sus aplicaciones con interfaces accesibles. Existen entre otros, teclados con Braille, aviso por vibración o sistemas táctiles sensibles a zonas con poca movilidad con un software predictivo que ayuda a la entrada de datos, como el utilizado por el Profesor Stephen Hawking, desarrollado por Intel.

d) En cuanto a interfaces para videojuegos, la más importante es el gamepad. Se trata de un dispositivo que el usuario debe coger con las dos manos y utilizar sus dedos para accionar los botones que contiene. Ha evolucionado desde el más sencillo con apenas dos botones y un stick digital, hasta el de ocho botones, dos gatillos, dos sticks analógicos y un stick digital más común hoy en día, además de sus capacidades ergonómicas.


Figura 2: Sega Master System gamepad.


Figura 1: Microsoft Xbox One gamepad.

En este ámbito podemos encontrar multitud de dispositivos enfocados al tipo de videojuego o a la simulación. Entre ellos, desde el clásico Joystick hasta una cabina de avión completa.

Figura 4: Joystick Thrustmaster T-Flight Stick X.


Figura 3: Volante Logitech Momo.

## Evolución y expansión de los interfaces para juegos

El mercado del entretenimiento ha liderado la innovación en los controladores de interacción persona-computador. Como ejemplo de ello cabe mencionar a Microsoft Kinect (2) que consiste en un sensor que representa un paso adicional hacia la implementación de interfaces completamente naturales en donde el cuerpo humano se transforma en el controlador. El dispositivo permite a los usuarios proporcionar comandos a la máquina por medio de gestos y posturas corporales. Tal que el hardware incluido se encarga de procesar en tiempo real los datos de base recibidos desde una cámara, que obtiene un esquema del esqueleto humano conformado por un conjunto de huesos y articulaciones. El reconocer la posición y orientación de los huesos le permite a este hardware identificar las posturas y gestos, los cuales pueden ser mapeados con comandos para controlar las aplicaciones. Kinect

fue diseñado en principio para los videojuegos, pero sus características tecnológicas son tan amplias que se aprovecha para multitud de usos, como por ejemplo: crear mapas geográficos, manejar robots a distancia, incluso en la medicina y otros campos.

Figura 5: Microsoft Kinect.


Figura 6: Leap Motion.

Los investigadores han propuesto también sensores que pueden realizar el seguimiento de las manos de un jugador. Por ejemplo, Leap Motion (3) puede realizar un seguimiento interactivo de las manos de un usuario al identificar la posición de los dedos y del centro de la palma, y luego computar las articulaciones de los dedos utilizando un procesador cinemático inverso. Algunos fabricantes de automóviles están proponiendo actualmente una alternativa de seguimiento de manos como modalidad de interacción en reemplazo de las pantallas táctiles tradicionales dirigidas a la gestión de las funciones de información y entretenimiento.

Otro de los campos de investigación que actualmente está en auge, es el de la Realidad Virtual (4), el cual es el motivo de este proyecto. En la década de los 80 ya hubo intención de comercializarla por parte de Nintendo, pero se quedó en un mero intento; la tecnología de la época no estaba suficientemente avanzada como para que los productos llegaran al gran público.

El principal objetivo de este concepto es dar al usuario la sensación de estar en un lugar físico y tangible. Y como consecuencia de ello, nació el producto llamado *Head-Mounted Display* (HMD) (5), que consiste en un casco provisto de una o varias pantallas que proyectan las imágenes muy cerca nuestros ojos. Además, constan de sensores para captar el movimiento de nuestra cabeza.

### 1.1.2 Videojuegos

Tratando de dar una perspectiva al prototipo realizado en este proyecto, se hará una introducción histórica a los videojuegos, especialmente a los First-Person que es la categoría en la que se engloba.

El primer juego interactivo (6) (7) data del año 1947 y sus desarrolladores únicamente contaban con una pantalla de tubo de rayos catódicos (Cathode ray tube - CRT) y junto con ella unas manecillas de forma circular que servían para controlarlo. Su jugabilidad consistía en utilizar dichos controles para hacer llegar un misil al objetivo. Aún no se podía calificar como video-juego, ya que el dispositivo no generaba señales de video.

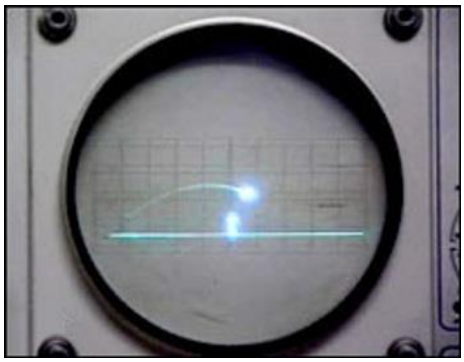


Figura 7: Dispositivo de Entretenimiento de Tubos de Rayos Catódicos.

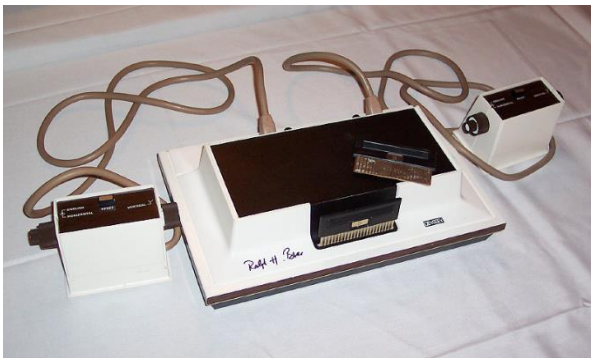


Figura 8: Magnavox Odyssey.

Para llegar a lo que se denomina juego de video o videojuego se tuvo que esperar a 1967 cuando Ralp Baer[1] reanudó su trabajo de desarrollar un juego interactivo en un televisor. Y a 1972 para la primera videoconsola para todos los públicos con el nombre de Odyssey creada por Magnavox (8). Sus dispositivos de interacción todavía eran muy rudimentarios: unas cajas con dos potenciómetros analógicos en sus laterales y un botón en la parte superior que servían para controlar los juegos. La consola funcionaba con los televisores de la época y proyectaba los pixeles dinámicos sobre un fondo estático.

Pronto, en 1977, Atari lanzaría al mercado su consola Atari VCS (Video Computer System)[2] y poco tiempo después, unas cuantas versiones del hardware con características superiores. Uno de sus dispositivos de control aún contenía potenciómetros (en los llamados Paddles), pero ya se podía observar la tendencia futura: los Joysticks. Pong era el juego estrella de esta máquina y consistía en un juego de ping-pong (La '**¡Error! No se encuentra el origen de la referencia.**' lo describe)

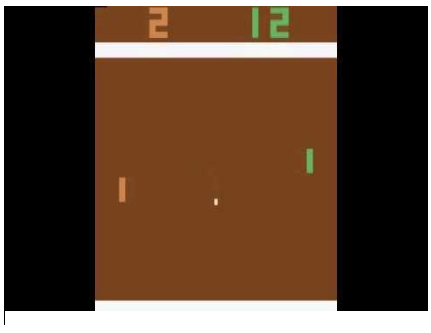



Figura 10: Pong.


Figura 9: Nintendo Virtual Boy.

Poco evolucionó en la siguiente década de los 80. El paradigma venía siendo el mismo, pero como es natural, mejorando el hardware de forma notable. Y la industria seguía creciendo gracias a las compañías niponas Atari, SNK, Sega y Nintendo principalmente y un número cada vez mayor de aficionados a los videojuegos.

[1] **Ralp Baer** fue apodado el padre de los videojuegos.
[2] **Atari VCS** era más conocida como Atari 2600.

En un intento de adelantarse en el tiempo y a sus competidores, Nintendo lanzó en 1995 un novedoso sistema de Realidad Virtual llamado Virtual Boy (9) (10). El concepto ya aparecía en obras de ciencia ficción de la industria del cine y la literatura, pero la tecnología de la época no alcanzaba para que dicha consola funcionase acorde con lo que se esperaba. Los juegos no conseguían esa Realidad Virtual, simplemente consistía en mostrar juegos, que ya eran comunes en el mercado, en dos pantallas para crear un efecto 3D, que además era muy molesto para los ojos. Apenas un par de juegos tenían la opción de cámara en primera persona (11). Existieron otros dispositivos para consolas como por ejemplo armas de fuego y juegos enfocados a ellas, como los que utilizaban dicho dispositivo para disparar apuntando hacia objetivos en la pantalla del televisor.

En cuanto a PC, antes de 1992 existían multitud de juegos muy parecidos a los de consola, pero en ese año surgió una gran evolución con Wolfenstein 3D. No era el primer First-Person shooter, pero sí fue un punto de inflexión. El motor gráfico creado por John Carmack[3] (12) era capaz de simular las tres dimensiones con escenarios en dos dimensiones texturizados de solo una altura; hasta entonces la visualización de los escenarios pseudo 3d era en colores sólidos. En lo jugable, los controles eran muy sencillos: rotar, caminar y disparar, no se podía inclinar la cámara hacia arriba ni hacia abajo, ya que esto consumía recursos que en la época no habían. Además, se utilizaba únicamente el teclado. Durante unos años aparecieron unos cuantos títulos más de este tipo de videojuegos, con su evidente evolución gráfica, muy característico del PC junto con las Aventuras Gráficas y los juegos de Estrategia. Pronto se desarrollaría el 3D verdadero y de él surgieron títulos como Quake, que ya podían utilizar el ratón para mover el punto de vista e inclinar la cabeza del personaje.

A partir de ese momento se irían desarrollando infinidad de juegos que utilizaban el teclado y el ratón como dispositivos de entrada. En más de una década apenas cambió ese concepto, pero si evolucionaba en otros aspectos, como por ejemplo el modo de narrar la historia, a la que antaño no se le daba demasiada importancia, pero hoy en día es una de las mayores bazas para la calidad del producto. Igualmente evolucionaban las formas de interactuar con los elementos y escenarios; existía una tendencia de que los entornos fueran cada vez más interactuables.

En las videoconsolas, el uso de dichos controles no es tan extendido y en un comienzo los FPS apenas tenían cabida en ellas hasta la aparición de gamepads con varios Sticks analógicos y gatillos.

La industria tenía intervalos de gran innovación y surgieron sensores como Microsoft Kinect, Nintendo Wiimote y Sony PlayStation Move, con intención de llevar la interacción de los juegos a otro nivel, utilizando tu propio cuerpo para controlarlos.

Hace apenas unos años, en 2012, se decidió volver a darle otra oportunidad a la Realidad Virtual, y fue entonces cuando apareció la empresa Oculus VR (13) (14) la pionera de este segundo intento. A ella se le han incorporado otras grandes compañías como Valve y HTC, unidas para dar nombre al HTC Vive (15) (16) (17), Microsoft con su HoloLens (18) (19) y Sony

[3] **John D. Carmack** es un programador de juegos estadounidense y cofundador de id Software.

con Playstation VR, como las más importantes. El principal controlador o dispositivo es el *Head-Mounted Display* que consiste en un casco con una pantalla y unas lentes que simulan el efecto de tres dimensiones y profundidad. También están provistos de sensores para capturar los movimientos del jugador y transmitirlas al juego. Este hardware vendría a sustituir principalmente al monitor como dispositivo de entrada. La Realidad Virtual podría suponer un gran salto evolutivo para los videojuegos en general y para los First-Person en particular, ya que son ese tipo de juegos los que le podrían sacar el mayor partido por su propio concepto. Actualmente ya se están desarrollando títulos y se han readaptado otros juegos ya existentes para soportar esta tecnología. El principal objetivo es la inmersión, realmente el jugador siente que está en ese mundo virtual con más fuerza que con los métodos anteriores.

## 1.2 Objetivos

Para la correcta realización de este trabajo, se han impuesto una serie de objetivos a cumplir de forma que al visualizar el videojuego al final de su desarrollo, se pueda comprobar si cumple con las expectativas marcadas.

Objetivos generales:

1. Investigar las posibilidades de las tecnologías de Realidad Virtual para el desarrollo de videojuegos.

2. Investigar las diferentes plataformas de desarrollo para la Realidad Virtual.

3. Realizar un prototipo de videojuego en primera persona empleando la tecnología de Realidad Virtual.

4. Implementación y validación de un modelo que mejore la experiencia jugable de la tecnología HMD.

Objetivos específicos de la aplicación:

1. Implementación y validación de un controlador del personaje jugable en primera persona con aceleración de movimientos.

2. Implementación y validación de una nueva fórmula de interacción con el juego, desvinculando el foco de visión con las acciones que efectúa el jugador.

3. Implementación y validación de un avatar jugable con animaciones básicas.

4. Implementación y validación de escenarios y objetos interactuables del videojuego, diseñados específicamente para la Realidad Virtual.

## 1.3 Estado del arte

### 1.3.1 Proyectos similares

En este apartado se describen brevemente los proyectos de mayor importancia con semejanza al proyecto presente. Se subrayan sus características, similitud y diferencias (20).

- **The Climb** (21) (22): Es un videojuego que está siendo desarrollado por Crytek y lo que pretende ofrecer es una experiencia de escalada enfocada a la Realidad Virtual. Crytek explica que, al basarse su jugabilidad en dicho deporte, su principal característica es que el jugador no necesita hacer movimientos bruscos con la cabeza, por tanto, el malestar debido a esto se palia en gran medida. La compañía también destaca que lo que consigue el juego es *presencia,* que consiste en que el jugador sienta que está dentro de un mundo que no existe en realidad. Se comercializará a lo largo de 2016.

  **ADR1FT** (23): Videojuego inspirado en la película "Gravity" que ofrece una experiencia inmersiva en primera persona (First Person Experience: FPX). Como en el largometraje, el jugador se pone en la piel de un astronauta que ha sobrevivido a un accidente espacial y debe buscar suministros entre los restos para sobrevivir. También se recalca la *presencia*. Saldrá a la venta en marzo de 2016.

  **The Witness** (24): Consiste en un juego de mundo abierto (Sandbox) representado como una isla con gráficos minimalistas, donde el jugador debe resolver una gran variedad de puzles y acertijos para llegar a su final. Como muchos otros, The Witness no está desarrollado exclusivamente para Realidad Virtual, aunque su integración con esta tecnología es excelente. Disponible desde enero de 2016.

  **Detached** (25): es una FPX enfocada en la competencia, exploración y supervivencia en un escenario espacial. El juego ha sido diseñado específicamente para la Realidad Virtual, por lo que ofrecerá a los jugadores una inmersión más sofisticada. Además, es un juego en mundo abierto donde se tendrá que resolver puzles mientras se compite con un rival. Su disponibilidad es para otoño de 2016.

- **P.O.L.L.E.N** (26): este juego está siendo desarrollado para los principales HMD de Realidad Virtual, y consiste en un thriller FPX de exploración en el espacio. Sus creadores afirman que el mayor potencial del juego es su gran interactividad con el mundo virtual. Su fecha de salida está aún por concretar, pero será durante 2016.

### 1.3.2 Tecnologías

## 1.3.2.1 Oculus Rift Development Kit 2

Se ha decidido adoptar como dispositivo de juego un controlador en primera persona de Realidad Virtual, específicamente Oculus Rift (27) Development Kit 2 (DK2) (28).


Figura 11: HMD de Oculus Rift Development Kit 2.

#### Características técnicas

El HMD del DK2 contiene una pantalla Samsung Galaxy Note 3 Super AMOLED[4] de 5.7 pulgadas con una resolución 1080p (1920x1080 píxeles, 960x1080 píxeles por ojo) con una tasa de refresco de 75Hz. Ya que consta de una sola pantalla que es adaptada al espacio de cada ojo mediante dos lentes (el kit de desarrollo también incluye lentes intercambiables que permiten la sencilla corrección de distrofias), hace falta una transformación óptica en la imagen para que la visualización sea correcta. Es el software el que se encarga tanto de este proceso, como de realizar el ajuste de la distancia interpupilar[5] que cada usuario debe configurar para poder mostrar la imagen estereoscópica correctamente. El visor dispone de un dial de cada lado que permite ajustar la distancia de la pantalla a los ojos. El campo de visión es de 100º en horizontal y 110º en vertical. Su peso es de 440 g (29).

[4] Un dispositivo **OLED de matriz activa** (AMOLED), consiste en un conjunto de píxeles OLED que se depositan o integran en una serie de transistores de película fina (TFT) para formar una matriz de píxeles, que se iluminan cuando han sido activados eléctricamente, controlados por los interruptores que regulan el flujo de corriente que se dirige a cada uno de los píxeles. El TFT continuamente regula la corriente que fluye por cada uno de los píxeles, para así caracterizar el píxel con el nivel de brillo que mostrará.

[5] Distancia entre pupilas.

Para rastrear el movimiento de la cabeza del usuario con una tasa de actualización de 1000 Hz y 6 grados de libertad de baja latencia utiliza una combinación de un giroscopio de 3 ejes, un acelerómetro y un magnetómetro. Además, posee una cámara con sensor CMOS[6] de infrarrojos que ayuda a capturar la posición global en relación a la tierra.

Figura 12: Cámara de infrarrojos de Oculus Rift DK2.

### Software

El software que distribuye Oculus junto con su HMD es principalmente de tres tipos:

- **Runtime:** Controladores que dan el servicio del HMD como otro monitor hacia el sistema operativo. Tiene una interfaz gráfica llamada *Oculus Configuration Utility* donde se configura el tipo de lentes, la medición del espacio interpupilar, la altura del jugador y una demo donde probar el *Traking* y el HMD.
- **Integración con motores gráficos:** Oculus da soporte para Unity, Unreal Engine y CryEngine. Consiste en un software que se integra en el IDE del motor y da acceso al SDK.
- **SDK (software development kit):** Librerías para el desarrollo de software de Oculus Rift.

### Aportación del controlador

El dispositivo de Realidad Virtual Oculus Rift nos proporciona datos sobre el movimiento de la cabeza del jugador, así como su posición global. Con esta información podemos recoger estadísticas más precisas sobre las entidades que observa el jugador, además de desarrollar objetivos y funciones que conlleven el uso de estos datos.

### Alternativas

A continuación, se muestra una comparativa de Oculus Rift con las alternativas más populares para PC:

[6] Un sensor de píxeles activos (*active pixel sensor* cuyo acrónimo es *APS*), es un sensor que detecta la luz. Está basado en tecnología CMOS (semiconductor complementario de óxido metálico) y por ello es más conocido como Sensor CMOS.

| | Oculus Rift DK2 | HTC Vive | Totem VR | Avegant Glyph | ANTVR | OSVR |
|---|---|---|---|---|---|---|
| **Resolución por ojo** | 960x1080 | 1200x1080 | 1280 x 1440 | 1280 x 720 | 960 x 1080 | 960x1080 |
| **Tasa de refresco** | 75 Hz | 90 Hz | 75 Hz | 120 Hz | | 120 Hz |
| **Panel** | AMOLED | OLED | OLED | Láser, proyecta imágenes en ojos | LCD | OLED |
| **Tamaño de panel** | 5.7” | 5.7” | 5.7” | | 6” | 5.5” |
| **Ángulo de visión** | 100º | 96º | 90º | 45º | 90º | 90º |
| **Seguimiento de posición** | Frontal | Completo | Completo | Ninguno | En desarrollo | Completo |
| **Lanzamiento comercial** | 2016 | 2016 | 2016 | 2015 | 2016 | 2016 |

Tabla 1: Alternativas a Oculus Rift.

Cabe destacar que en el momento de la adquisición del hardware HMD no existían las alternativas nombradas. Oculus VR es pionero en esta segunda oportunidad a la Realidad Virtual.


Figura 13: HTC Vive.

Figura 15: Totem VR.

Figura 14: Avegant Glyph.

Figura 16: ANTVR.


Figura 17: OSVR.

### 1.3.2.2 Unity 3D

Unity (30) (31) es un motor gráfico para videojuegos, en 2 y 3 dimensiones, creado por Unity Technologies. Es una aplicación multiplataforma que puede ejecutarse en Apple OS X, Linux y Microsoft Windows y permite desarrollar juegos, aplicaciones interactivas, visualizaciones y animaciones en tiempo real[7] para los sistemas operativos Windows, Mac, Linux, Android, BlackBerry, para las videoconsolas Xbox 360, Xbox One, Playstation 3, Playstation 4, Playstation Vita, y para WebGL[8] y Samsung Tv actualmente.

El IDE[9] de Unity es el centro de la línea de desarrollo que consiste en un editor visual con una serie de potentes herramientas para la creación de aplicaciones con entorno gráfico. Este contenido gráfico puede ser realizado desde el propio editor o ampliado gracias a la compatibilidad con aplicaciones de creación de gráficos y animación 3D como son 3D Studio Max y Maya entre otros. Además, cuenta con la integración con Visual Studio 2015 mediante una extensión.

La jugabilidad o *gameplay* se programa mediante lenguajes de scripts C#[10], Javascript y Boo[11]. El núcleo del motor nos provee de una serie de eventos, que son fácilmente programables para cada script, asociados a los objetos[12] o elementos, y de esa manera define los comportamientos de éstos. Para este prototipo se ha utilizado C#.

La estructura de las aplicaciones en Unity son separadas por escenas que constan de un conjunto de objetos gráficos, disparadores y scripts. Dichos objetos pueden ser jerarquizados de forma visual y cómoda que facilita la edición y relación entre objetos y elementos de la aplicación.

Unity tiene un gran apoyo de la comunidad de desarrolladores profesionales y noveles. En su tienda (Asset Store) que viene integrada en el propio IDE, podemos encontrar gran cantidad de elementos así como proyectos de ejemplo completos. Como muestra de esta actividad de la comunidad, los foros y canales de *youtube* son muy activos.

Este motor gráfico está en continuo desarrollo y prueba de ello es que cada año preparan una nueva versión con multitud de mejoras.

La licencia de Unity es gratuita, pero cuenta con una versión *Pro* que incluye varias características y herramientas que no se encuentran en la versión gratuita. Con la versión *Pro* el desarrollador puede comercializar su producto.

---

[7] Un sistema en **tiempo real** (STR) es aquel sistema digital que interactúa activamente con un entorno de forma dinámica y sus procesos de ejecución se realizan en el momento en que recibe un parámetro de entrada o una interacción con el usuario.

[8] **WebGL** es una especificación estándar que está siendo desarrollada actualmente para mostrar gráficos en 3D en navegadores web. El WebGL permite mostrar gráficos en 3D acelerados por hardware (GPU) en páginas web, sin la necesidad de plug-ins en cualquier plataforma que soporte OpenGL 2.0 u OpenGL ES 2.0. Técnicamente es un API para javascript que permite usar la implementación nativa de OpenGL ES 2.0 que será incorporada en los navegadores.

[9] **Integrated Development Environment**: ambiente de desarrollo integrado o entorno de desarrollo integrado.

[10] **C#** o **C Sharp** es un lenguaje de programación orientado a objetos desarrollado por Microsoft y basado en C/C++ para su plataforma .NET.

[11] **Boo** es un lenguaje de programación orientado a objetos, de tipos estáticos con una sintaxis inspirada en Python.

[12] Un **objeto** o ***GameObject*** en Unity es un contenedor que puede tener objetos gráficos, elementos de física, fuentes de sonido, cámaras y scripts.

Como alternativas a Unity más conocidas y potentes se ha de destacar Unreal Engine, desarrollado en su primera versión para el juego multijugador Unreal Tournament y que actualmente es un referente en cuanto a iluminación realista en tiempo real, y CryEngine desarrollado por la empresa alemana Crytek y que ha dado títulos de gran calidad gráfica realizados por la propia compañía y terceros.

La elección de Unity como motor gráfico para la creación de este prototipo estriba en lo siguiente:

- En momento de comenzar el proyecto, Unity era el único motor para el que Oculus VR proporcionaba herramientas completas para el desarrollo de aplicaciones con su dispositivo, por lo tanto, facilitaba su programación.
- La curva de aprendizaje del propio Unity es más accesible que su competencia.
- Los scripts de Unity pueden ser escritos en C#, siendo este un lenguaje que ya se utilizaba.
- Cuenta con gran cantidad de contenido ya creado por la propia Unity Technologies y la comunidad que lo respalda.

### 1.3.3 Tipo de juego

La base de la jugabilidad del prototipo es un juego de cámara en primera persona (*First-Person game*), es decir, la perspectiva o punto de visión está a la altura de la cabeza del personaje. La particularidad es que se ha incluido el HMD, por lo tanto entra también en la subcategoría de experiencia inmersiva en primera persona (First Person Experience: FPX). La ventaja que proporciona ésta vista es que lo dota de mayor realismo y sensación de presencia. Sin embargo, la desventaja es que el jugador debe rotar el personaje para observar lo que le rodea. Además, el juego podría englobarse en la categoría de puzles y plataformas.

Otra característica jugable y que está vinculada principalmente con la Realidad Virtual es la *presencia* que consigue el juego. Como se ha dicho anteriormente, consiste en que el jugador perciba que está dentro de un mundo que no existe en realidad.

### 1.3.4 Argumento

Es una historia de ciencia ficción cuyo argumento es el siguiente:

Hace milenios una especie extraterrestre modificó la genética de nuestros antepasados. Ahora, ellos abducen a los especímenes humanos que han estado estudiando de forma clandestina y que hayan demostrado ciertas capacidades y actitudes. En ese momento, los teleportan a algún lugar de la galaxia de Andrómeda, donde tienen preparado un artefacto que consiste en una esfera transparente que provoca una baja gravedad y que contiene una serie de pruebas. Los abducidos son obligados a superar estos puzles con el objetivo de obtener datos sobre su memoria, percepción, atención, capacidad emocional e inteligencia. Lo que ellos pretenden es medir cómo ha evolucionado su gen.

## 1.4 Arquitectura de la aplicación

Para poder entender cómo funciona la aplicación hay que pasar desde la perspectiva del usuario hasta el software del juego (donde se han desarrollado los módulos). En la siguiente tabla se muestra la arquitectura de la aplicación y la interacción entre apartados y el flujo de datos:

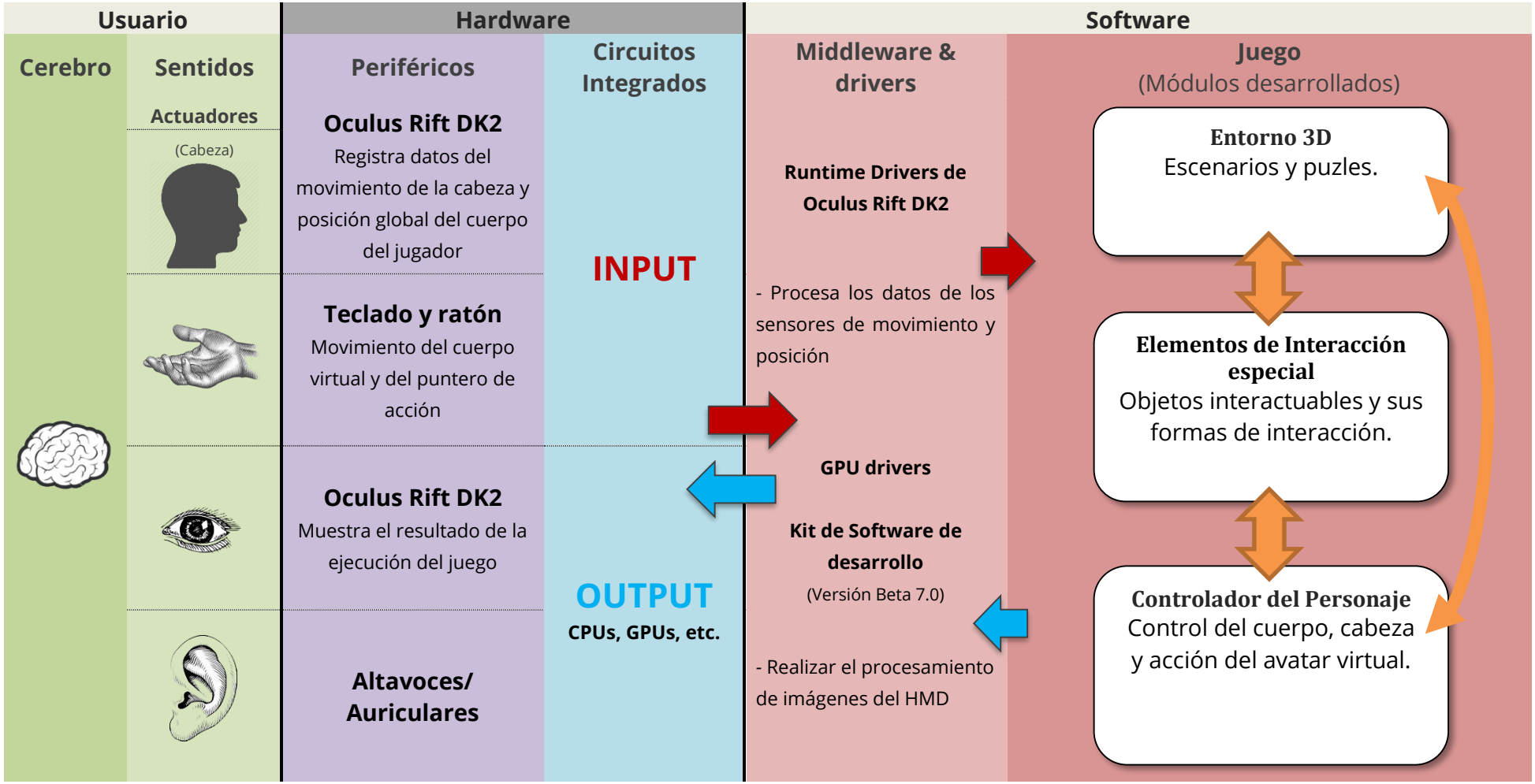



Tabla 2: Arquitectura de la aplicación.

El usuario (o jugador) de la aplicación recibe datos a través de sus sentidos vista y oído. En reacción a este estímulo, utiliza su cabeza y manos para enviar órdenes al juego, recogidos respectivamente por el HMD, teclado y ratón. El hardware recibe los datos de los sensores de movimiento y posicionamiento global que posteriormente son enviados al controlador *Runtime*. En el módulo de *Controlador de Personaje* dichos datos se procesan para mejorar la respuesta al suavizar los movimientos y realizar un control específico para este hardware. El personaje se encuentra en un *Entorno 3D* que contiene objetos interactuables con sus respectivos *Elementos de interacción especial*. Los tres módulos envían datos del personaje, objetos y mundo 3D a las librerías del *Kit Software de desarrollo* y *Runtime* que se encargan de aplicar los algoritmos para la correcta visualización de las imágenes en el HMD.

Siendo más específicos, y en el apartado que se ha desarrollado; El módulo de *Elementos de interacción especial* es el que contiene los métodos, eventos y atributos de los objetos con los que puede interactuar el personaje. Estos objetos han sido diseñados para que el uso del HMD sea evidente. Los movimientos y las acciones del personaje son descritos en el módulo *Controlador de Personaje*; ambos son enviados a los objetos para lograr su funcionamiento. A su vez, el entorno es el que contiene al personaje y a los objetos, y recibe datos de ambos para ejecutar el procedimiento de los puzles.

En los siguientes capítulos se describirán en detalle los tres módulos desarrollados.

# Capítulo 2.
# Controlador del personaje

Como se ha dicho anteriormente, se ha adoptado la Realidad Virtual para este prototipo. Lo que principalmente logra esta tecnología es una mayor inmersión en el juego. La forma en la que se realiza esto es proyectando para cada ojo una imagen que distan de una diferencia de punto focal, simulando la visión con profundidad humana. Asimismo, proporciona otra referencia de profundidad llamada paralaje. En conjunto ofrecen un efecto de inmersión mejorado con respecto a la de un monitor estándar. El rango de visión ocupa prácticamente todo lo que ve el jugador al tener la pantalla muy cercana a los ojos. Todo esto nos permite desarrollar situaciones que logren un determinado impacto emocional.

Oculus Rift hace posible el uso de acciones relacionadas con la función de posicionamiento global, que consisten en que el jugador deba colocar su cuerpo de maneras determinadas, para cumplir unos objetivos que se describirán en capítulos posteriores.

En los juegos clásicos de First-Person, la rotación del punto de vista, rotación del cuerpo y control de la acción se realiza conjuntamente con el movimiento del ratón. Para este prototipo, estas tres acciones se han separado; en cuestiones del personaje jugable, el movimiento del cuerpo se controla con el teclado, el control de la acción se realiza con el ratón y el punto de vista se maneja con el HMD.

La jugabilidad ha sido diseñada pensando en una fácil asimilación de los jugadores experimentados en este tipo de videojuegos, con la intención de que la adopción de la tecnología no conlleve un cambio drástico en la forma de jugarlos, pero sí en la intensidad y en la cantidad de acciones que es posible realizar. Ya de por sí, la adaptación al controlador es complicada, por esta razón se ha optado por desarrollar un híbrido con los controles y dispositivos clásicos. Por otro lado, se ha tenido en cuenta el costo del sistema.

La aceleración en las acciones se ha implementado para mitigar, en lo máximo posible con la tecnología tomada, la sensación de mareo que produce el controlador en el jugador con los movimientos bruscos y repentinos.

Separar la acción del punto de vista nos permite desfocalizar la propiamente dicha acción, con lo que se consigue un control más realista y profundo además de aportar otra forma de jugar.

## 2.1 Cuerpo del personaje

Para este prototipo se ha escogido un personaje incluido en las bibliotecas estándar del editor de *Unity*. Este cuerpo virtual viene provisto de animaciones básicas a las que, además, se le han agregado otras ajenas a dicho cuerpo para completar el rango total de las animaciones que se querían integrar. Todas ellas se han adaptado para su correcta

visualización en primera persona y para Realidad Virtual, dado que de base estaban implementadas para una visualización en 3ª persona. Además, se ha desarrollado el funcionamiento y el accionamiento de dichas animaciones.

Figura 18: Cuerpo del personaje.

Hay varios parámetros, definidos por el editor, que se han modificado para hacer funcionar correctamente las animaciones. Asimismo, se ha agregado un nuevo parámetro en el Animador llamado Velocidad, así como varios flags para implementar la funcionalidad: avanzar, retroceder, movimiento lateral derecho e izquierdo, rotación derecha e izquierda, saltar y agacharse. El Animador es un módulo de *Unity* que sirve para crear el funcionamiento de las animaciones y los enlaces entre ellas.

Para cada animación se han ajustado los dos tipos de parámetros nombrados y los flags, habiendo realizado lo siguiente:

a) Construcción de los enlaces de transacciones con el resto de animaciones.
b) Ajuste del parámetro correspondiente para dar paso de una a otra. Por ejemplo, tenemos la animación de caminar que transita a correr si la velocidad del personaje llega a un umbral.
c) Determinación del tiempo de inicio y final, para que las transacciones entre animaciones resultaran más cómodas en la vista de primera persona.
d) Ajuste de la orientación del cuerpo en el momento de realizarse la animación.
e) Ajuste de los ejes en los que debe desplazarse.

A continuación, se enumeran las 27 animaciones implementadas en el personaje: Cuando el jugador no realiza ninguna acción con el teclado se ejecuta la animación de reposo u ocioso, incluso si mueve la cabeza o el ratón únicamente. Se han implementado las animaciones de caminar y correr hacia adelante, igualmente que caminar y correr hacia atrás, y en el movimiento lateral izquierdo y derecho, normal y corriendo. Cuando el personaje rota hacia la izquierda o la derecha también se distinguen las dos velocidades de movimiento, haciendo que dé los pasos en menor o mayor ángulo. Si el jugador salta estando en reposo se ejecuta la animación correspondiente, a la par que al saltar con desplazamiento donde la animación accionada es diferente. También se ha ajustado la animación de agacharse y levantarse si el personaje ya está agachado. Encontrándose en ésta última posición, se diferencian las animaciones de caminar y correr hacia adelante agachado, lo mismo que caminar y correr hacia atrás agachado, y los movimientos laterales de izquierda y derecha normal y corriendo. Por último, el personaje también puede rotar hacia la izquierda o la derecha mientras está agachado.

## 2.2 Modo de funcionamiento

El objetivo del modo de funcionamiento es dar un control claro y debidamente separado de las posibles acciones que se pueden realizar sobre el personaje. Así, como ya se ha descrito, el control del cuerpo del personaje se realiza con el teclado, el del punto vista o cámara se realiza directamente con el HMD, y el del cursor de la acción se realiza con el ratón. De esta manera se consigue un control intuitivo, dado que están separados cada uno por un dispositivo diferente.
Los tres tipos de control se describen en detalle a continuación:

### 2.2.1 Control del cuerpo del personaje

Debido a que la sensación de mareo es notable en la versión temprana del HMD utilizado, la principal funcionalidad es suavizar los movimientos del personaje con una aceleración, como se ha dicho anteriormente.

El movimiento del cuerpo del personaje tiene como base la jugabilidad de los videojuegos clásicos de *First-Person*. Se realiza con el **teclado** y la asignación de teclas se ha elaborado mediante el proceso de ensayo y error, teniendo en cuenta que el resultado debía ser un control lo más intuitivo posible con el dispositivo nombrado.

El mapeo por defecto es con las siguientes teclas:

| Tecla | Función |
|---|---|
| **W** | Avanzar |
| **A** | Rotación hacia la izquierda |
| **S** | Retroceder |
| **D** | Rotación hacia la derecha |
| **Q** | Avance lateral hacia la izquierda |
| **E** | Avance lateral hacia la derecha |
| **Espacio** | Saltar |
| **Mayus** | Aumento de velocidad de desplazamiento |
| **Teclas especiales** | |
| **F2** | Reseteo de la posición del cuerpo respecto el HMD |
| **Esc** | Salir del juego |
| **Tab** | Menú debug |

Tabla 3: Mapeo de teclas de movimiento del personaje.

Esta configuración está parametrizada desde el editor de *Unity*. Se ha optado por estos controles por ser de costumbre para los jugadores experimentados de *First-Person*, con la diferencia de que los movimientos laterales se realizan con las teclas **'A'** y **'D'**. La manera descrita resulta más intuitiva debido a que para la rotación del cuerpo no es posible utilizar el ratón, como se hace en los clásicos, dado que a este dispositivo se le da otro uso. Además, la posición en el teclado de dichas teclas es ergonómica junto con el resto de asignaciones.

Para mayor comodidad del usuario, se ha asignado la tecla 'F2' con la que podrá colocar el cuerpo virtual respecto a su punto de visión. En la versión del SDK utilizada, en ocasiones la posición global no sitúa correctamente el HMD en la escena, e incluso se da el caso de tener que reiniciar la aplicación.

### 2.2.2 Control del punto de vista o cámara

El control del punto de vista (o cámara) se realiza con el mismo **HMD**, que ya de por sí ha de ir en la cabeza del usuario. Al moverla también estará activando los sensores del dispositivo. Con esto, los datos recibidos por la aplicación son utilizados para mover también la cabeza del personaje y con ello las cámaras.

Hay una cámara virtual para cada ojo, dado que necesitan su propia proyección para simular la profundidad en la pantalla del HMD.

La finalidad de este funcionamiento es dotar al personaje de un cuerpo lo más realista posible para que el usuario se sienta dentro de él. El movimiento de la cabeza del personaje no se puede distinguir directamente cuando se produce, puesto que queda en ángulo muerto para las cámaras, pero si en la sombra que proyecta. Esto es importante para dar más intensidad a la inmersión jugable.

La cabeza y el cuello del personaje pueden rotar sobre los tres ejes (32):

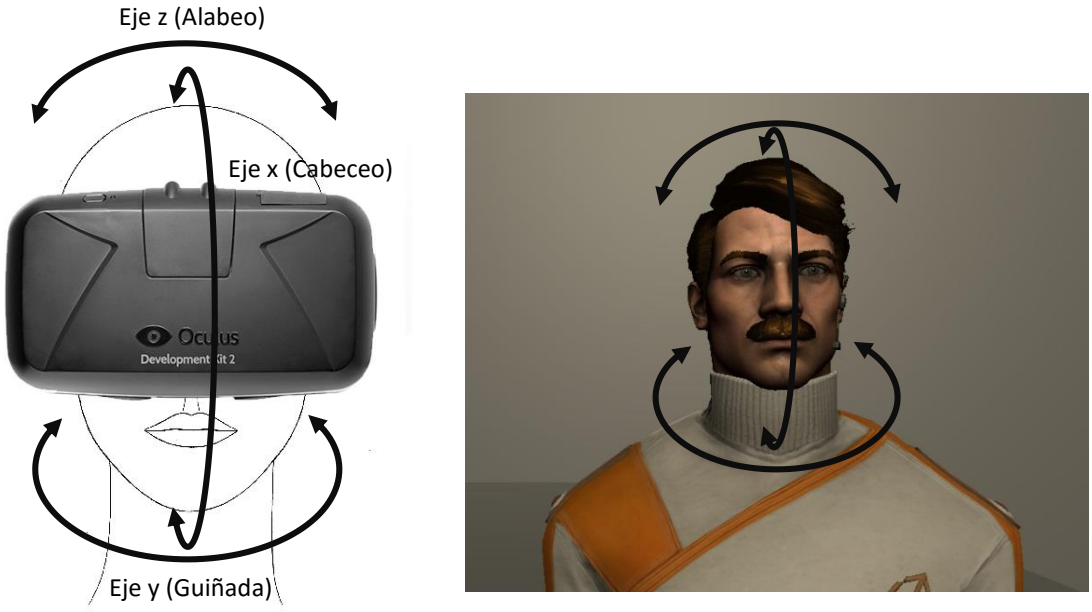



Figura 19: Descripción del movimiento del HMD con la cabeza del personaje.

### 2.2.3 Control de la acción

El control de la acción es la activación de determinados objetos interactuables que serán descritos con posterioridad. En los juegos clásicos la vista y el control de la acción se manejan con el ratón conjuntamente. Debido a que contamos con el HMD que realiza el control de la vista directamente, el control de la acción recae en el **ratón** de forma independiente. Gracias a esta separación de acciones, es posible realizar mayor cantidad de tareas.

Dicho con otras palabras, el propósito de esta función es desvincular hacia donde mira el jugador de donde realiza sus acciones. Así, por ejemplo, podrá observar un objeto, mientras activa otro que no se encuentra dentro de su rango de visión.

En pantalla se muestra un *Crosshair*[13] que, como se ha dicho, puede controlarse de forma distinguida de la cámara. Contiene una función de *Raycasting* que sirve para ajustar la distancia del *Crosshair* al objeto al que apunta con un máximo 1.2 metros. Esta es la distancia máxima, que se ha establecido, en la que se puede activar un objeto interactuable con dicha función, aunque existen otras maneras descritas con posterioridad.

El *Crosshair* adapta su distancia según al objeto más próximo al que este apuntando.

[13] **Crosshair** o retícula. Muestra la posición del cursor del ratón principalmente en videojuegos.

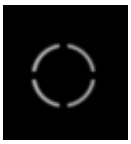


Figura 20: Crosshair.

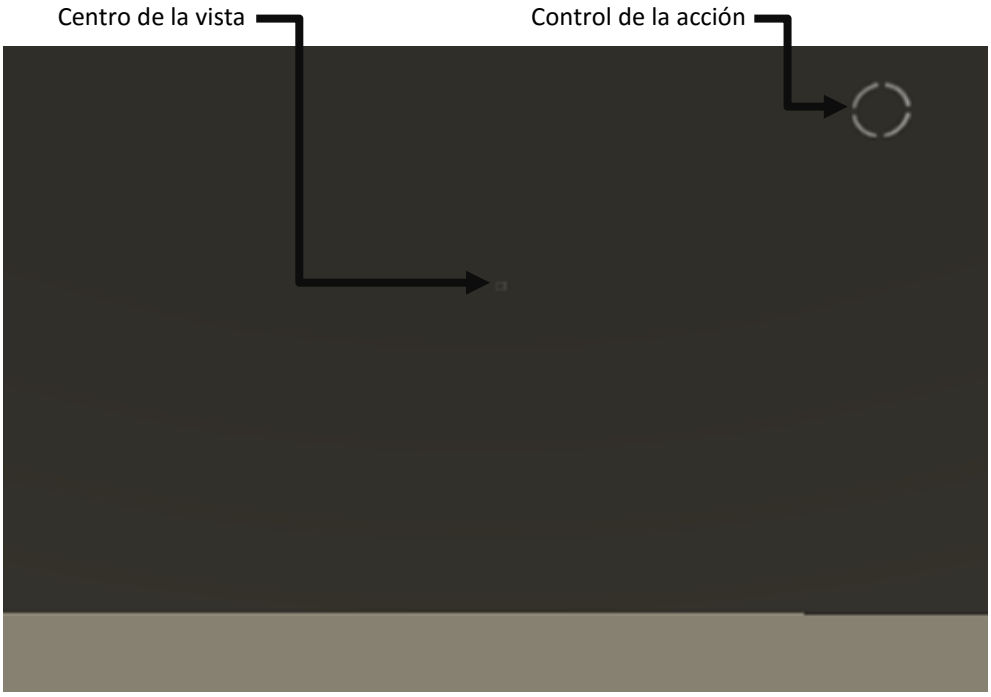



Figura 21: Descripción de la separación de centro de la vista y control de la acción.

### 2.2.3.1 Determinación de posibilidad de interactuar

Para dotar de mayor comodidad la interacción del usuario con el juego, se ha desarrollado otra función que consiste en mover el brazo izquierdo del personaje de forma automática, de la manera mostrada en 'Figura 23: Brazo izquierdo activado.', ante la posibilidad de interactuar con determinados objetos cuando son observados directamente por el jugador.

En una versión anterior de esta funcionalidad, el jugador solo debía mirar al elemento interactuable para desencadenar su activación. Esto provocaba la activación de dichos objetos de forma involuntaria. La función fue modificada de forma que el jugador debía hacer clic sobre el reloj que lleva en el brazo izquierdo. Sin embargo, esto se descartó por ser complicado para el jugador acertar con el ratón sobre él en circunstancias en las que no quedaba a la vista, ya que los objetos que utilizan esta función necesitan fijar la mirada en ellos para acceder a una fase de su activación.

La funcionalidad ha sido evolucionada modificando la necesidad de hacer clic sobre el reloj, con hacerlo simplemente con el botón derecho del ratón sin tener que apuntar a ningún objeto.

Su utilidad es para mostrar al jugador que puede realizar una acción sobre determinados elementos que se describirán en capítulos posteriores.

Figura 22: Brazo izquierdo desactivado.

Figura 23: Brazo izquierdo activado.

### 2.2.4 Control del cuerpo del personaje

- **Movimiento del cuerpo del personaje** (`PlayerControl`): se han utilizado las siguientes variables en la clase desarrollada:

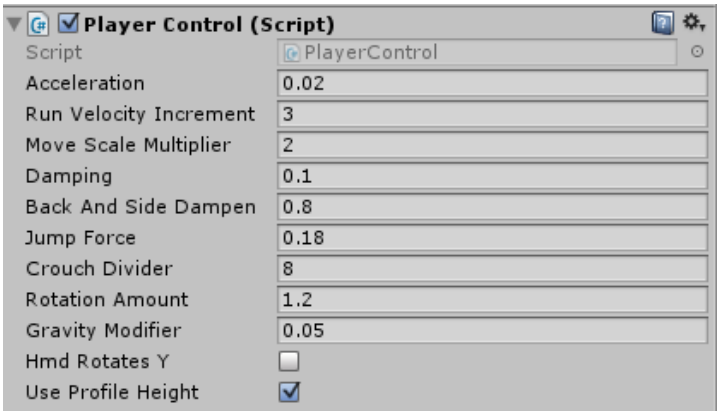



Figura 24: Parámetros del Player Control.

Donde:

| Propiedad | Función |
|---|---|
| **Acceleration** | Aceleración de los movimientos de avanzar, retroceder, lateral y rotación. |
| **Run Velocity Increment** | Incremento de la velocidad al correr. |
| **Move Scale Multiplier** | Multiplica la influencia del movimiento para darle una escala según la preferencia del desarrollador. |
| **Back and Side Dampen** | Multiplicador cuando el personaje se mueve hacia atrás o lateralmente. |
| **Jump Force** | Fuerza del salto. Tendrá en cuenta el modificador de gravedad. |
| **Crouch Divider** | El divisor de la altura del cuerpo cuando el personaje se agacha. |
| **Rotation Amount** | El ratio de rotación. |
| **Gravity Modifier** | Modificador de la gravedad. La gravedad de la tierra se ajustaría aproximadamente en `0.137`. |
| **Use Profile Height** | Si es *true* se ajustará la altura del personaje configurada en el software Runtime de Oculus. |

Tabla 4: Descripción de propiedades del controlador de Player Control.

- **Movimiento del brazo izquierdo** (`Switches`, `SwitchWatch`, `PlayerAnimator`): previamente se ha guardado la configuración de las articulaciones del brazo en la posición ideada para mostrarse activado. Mediante el *Raycasting del Crosshair* (se describirá con posterioridad en este mismo capítulo) se determina si el objeto en su forma de interacción es correcto y si puede ser accionado en ese momento. En caso positivo, asigna a las articulaciones del brazo a su posición activado. En otro caso, el brazo vuelve a su posición original en reposo.

### 2.2.5 Control del punto de vista o cámara

Contiene una función de *Raycasting* para determinar a qué objeto se está mirando céntricamente y a que distancia está. El máximo al que se ha ajustado por defecto es a 20 metros.

- **Movimiento de la cabeza** (`PlayerMoveHead`): simplemente se asigna la rotación de las cámaras a los objetos del cuello y cabeza. Es importante la posición de las dos cámaras respecto a la cabeza y el cuello, ya que podría causar que éstos se antepusieran produciendo un defecto en la visualización.

- ***Raycasting de las cámaras*** *(`Raycasting` type camera)*: se proyecta un rayo sobre el 'eje z' positivo de la cámara hacia 20 metros. Recibe el colisionador más próximo. Existen dos tipos de objetos en la escena en cuanto a colisiones; los que reciben el *Raycast* y los que lo ignoran. La justificación de la utilización de estos dos tipos de objetos se realizará posteriormente.


Figura 25: Parámetros del Raycasting para las cámaras.

### 2.2.6 Control de la acción

- (`CrosshairControl`) **Control del *Crosshair*:** muestra el movimiento del ratón con unos límites por defecto de -90º y 90º en el '*eje x*' y -75º y 75º en el '*eje y*'.

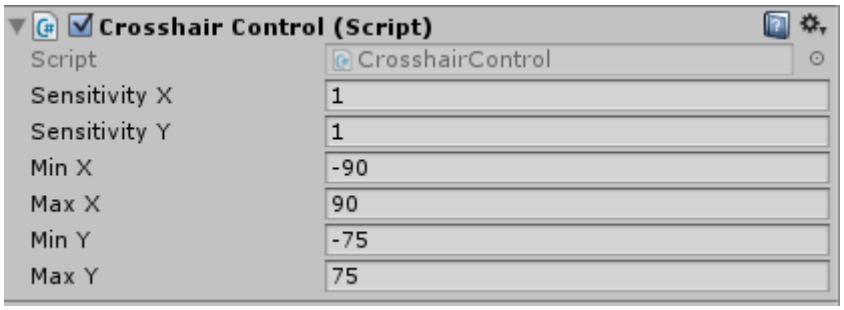



Figura 26: Parámetros del Crosshair Control.

- ***Raycasting* del *Crosshair*:** se proyecta un rayo sobre el 'eje z' positivo respecto al personaje hacia 1.2 metros. Recibe el colisionador más próximo.


Figura 27: Parámetros del Raycasting para el Crosshair.

# Capítulo 3.
# Elementos de interacción especial

En el presente capítulo se describirán los objetos interactuables del juego, además de otros elementos lógicos imprescindibles para la implementación.

## 3.1 Interruptores

Se han desarrollado cuatro tipos de interruptores: *SwitchA*, *SwitchB*, *SwitchWatch* y *SwitchTeleport*.

### 3.1.1 SwitchA

El interruptor *SwitchA* es el primero que encontrará el usuario en el juego y da una primera instancia de cuál es la dinámica de activación del resto de interruptores. El jugador deberá experimentar con él para descubrir su proceso de activación, que aunque es sencillo, no es el convencional; le obliga a utilizar tanto el movimiento de su cabeza como el de la acción desvinculada (ratón).

Los tres estados del interruptor se identifican por colores:

- ***Desactivado***: color azul, estado de reposo del interruptor.
- ***Observado***: colores naranja y verde, el interruptor está siendo observado directamente por el usuario. Cambia a este color indicando que está a la espera de ser activado.
- ***Activado***: de color verde.

Su proceso de activación es el siguiente: por defecto, se encuentra en estado ***desactivado***. Cuando el jugador lo mira directamente cambia a su estado ***observado***. Estando en este último estado y no en otro, haciendo clic sobre su superficie, se **activará** quedando en ese estado. Si se vuelve a hacer clic sobre él, siendo *observado* o no, se **desactivará**.

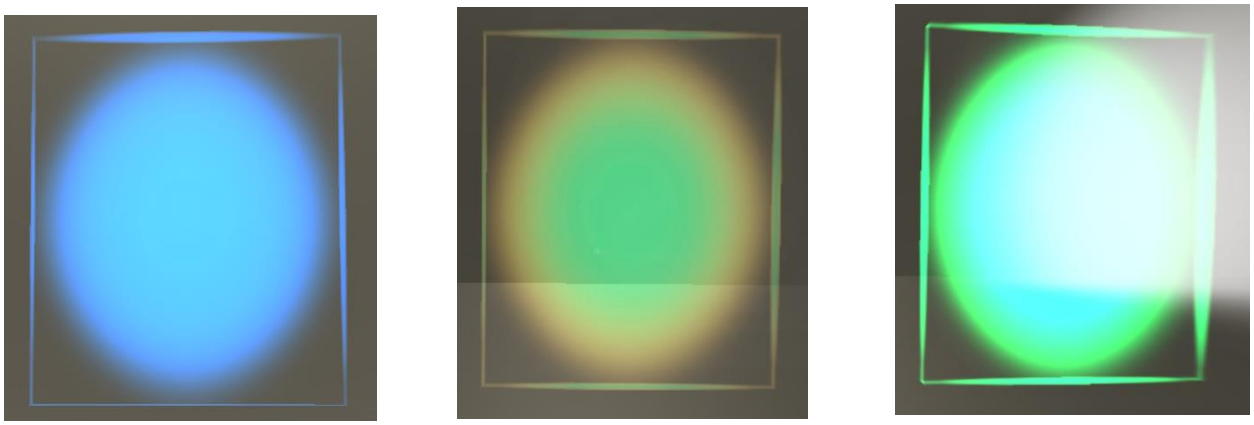


Figura 28: SwitchA estados: desactivado, observado y activado.

### 3.1.2 SwitchB

La característica de este interruptor es que obliga al jugador a utilizar tanto el movimiento de su cabeza, como el de la acción desvinculada (ratón). Los elementos que le dan forma son *el Activador por cámara* y *el Activador por acción*: el primero, *el Activador por cámara,* es un interruptor que se activa al ser mirado desde cerca. Este hace visible un *Haz de luz* que debe focalizarse sobre un *Receptor* para ser activado. El segundo, *el Activador por acción,* es un botón que debe ser pulsado simultáneamente por el usuario.

*El Activador por cámara* y *Activador por acción* no están dentro del rango de visión cuando se interactúa con el interruptor, el jugador se verá obligado a mirar al *Activador* por acción perdiendo de esa manera el foco del *Receptor*. Igualmente, deberá experimentar con él, ya que su activación no es tan obvia ni sencilla como la de *SwitchA*. Sin embargo, en la escena y en el propio interruptor se encuentran pistas de cómo realizarla: por la cultura de activación del resto de interruptores y por los elementos que le rodean.

Este interruptor consta de cuatro partes diferenciadas:

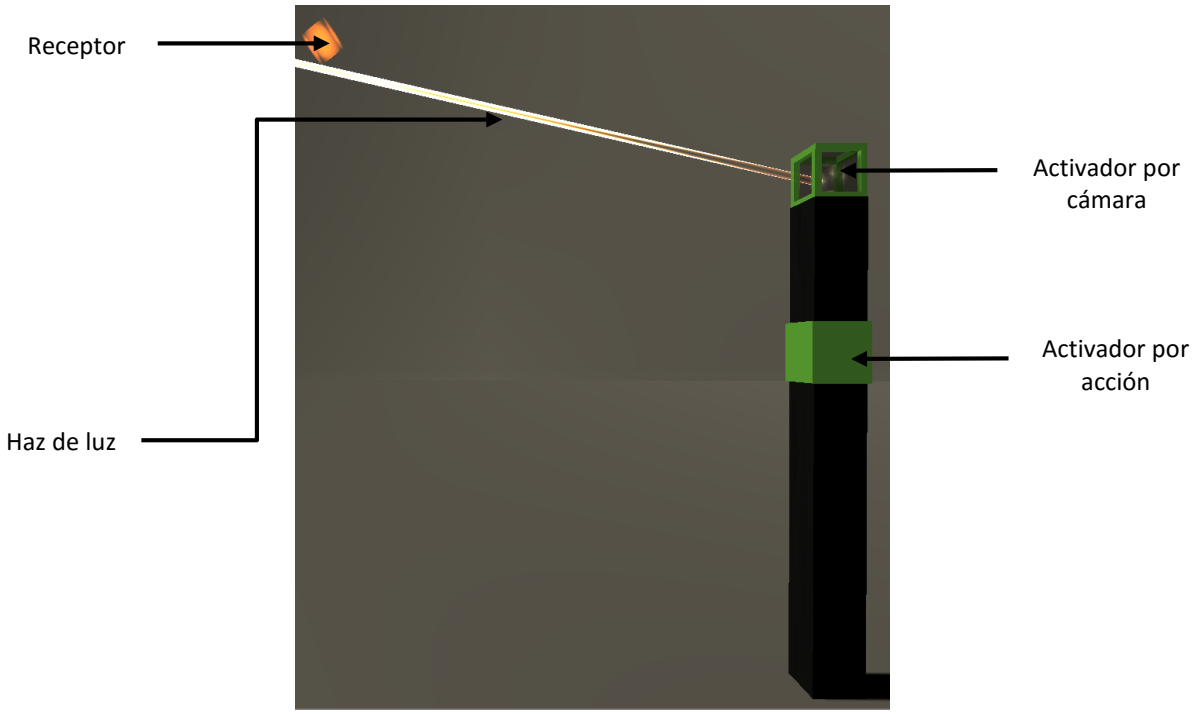



Figura 29: Partes de SwitchB.

Además de cuatro estados:

- ***Desactivado***: estado de reposo del interruptor.
- ***Observado***: el interruptor está siendo observado directamente por el usuario. Aparecerá el *Haz de Luz* de color amarillo que permanecía invisible hasta ese momento.
- ***Apuntado***: el *Haz de Luz* está incidiendo con el *Receptor*. El *Haz de Luz* cambiará a color naranja.
- ***Activado***: El *Haz de Luz* cambiará a color verde.

Su proceso de activación es el siguiente: por defecto, se encuentra en estado ***desactivado***. Cuando el jugador mira directamente al *Activador de cámara*, cambia a su estado ***observado***. Para ello es necesario que el personaje se sitúe lo más cerca posible del interruptor y se incline hacia adelante. Moviendo la cabeza, el jugador podrá hacer incidir el *Haz de Luz* con el *Receptor*, en ese momento pasará al estado ***apuntado***. Partiendo de este estado y no de otro, para concluir la **activación**, se debe hacer clic sobre la superficie del *Activador por acción*. Si se vuelve a hacer clic sobre él, quedará en ***apuntado***.

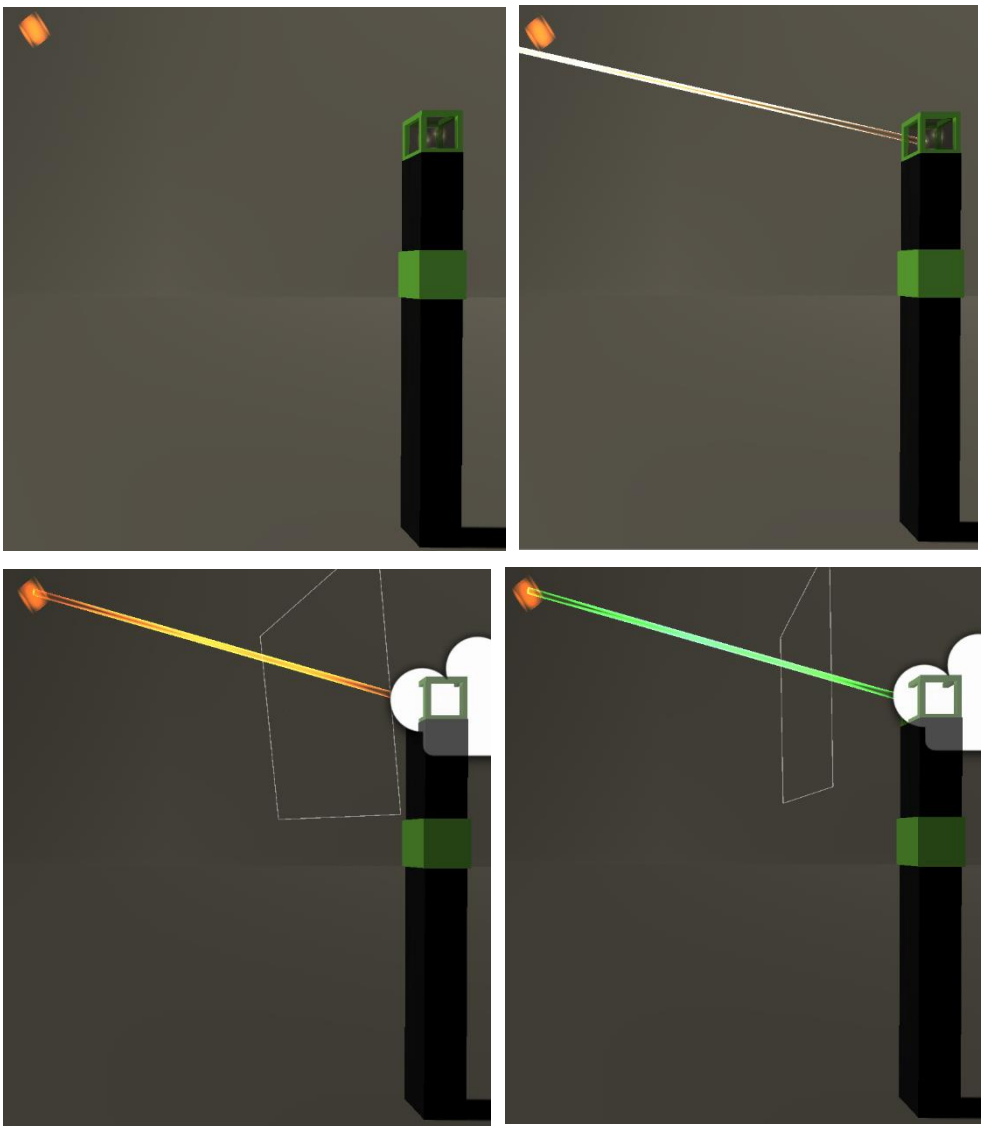


Figura 30: SwitchB estados: desactivado, observado, apuntado y activado.

### 3.1.3 SwitchWatch

*SwitchWatch* es un interruptor intrínseco al personaje, pero solo se puede utilizar si el jugador puede realizar una acción a distancia, como mover una plataforma con la vista. Es decir, como se ha visto en capítulos anteriores, cuando el *brazo izquierdo* está activado. Si en ese momento, se hace clic derecho, sin importar donde se encuentre el puntero del ratón, se activará el movimiento de la plataforma observada.

La ventaja de este interruptor radica en que el usuario puede activar ciertos objetos desde lejos.

Figura 31: Indicador de utilización de SwitchWatch.

### 3.1.4 SwitchTeleport

Este interruptor desencadena el final de la demo del prototipo y solo es necesario hacer clic sobre él para activarlo. Sin embargo, para hacerlo visible es necesario que el jugador haya recogido un objeto designado al efecto en la escena. Si se hace clic sobre el interruptor, accionará la animación del *Teleportador*.

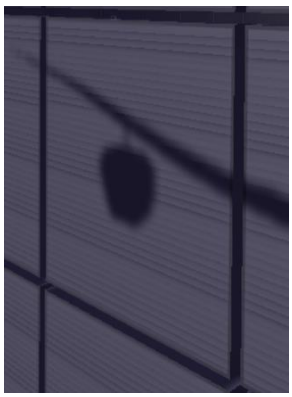


Figura 32: SwitchTeleport.

## 3.2 Plataformas

### 3.2.1 PlatformSwitchA, PlatformSwitchB y PlatformSwitchC

Estas plataformas sirven para alcanzar lugares inaccesibles de la escena, tanto como en el desplazamiento del personaje, como por el desbloqueo de zonas producido por el propio desplazamiento. Las características de este tipo de interruptor son que obliga al jugador a utilizar el movimiento de su cabeza y además es posible activarlo desde lejos.

Sus ejes de desplazamiento son el 'X' y el 'Z'.

Para activar su movimiento el jugador debe mirar a las zonas de color verde de las plataformas y hacer uso de *SwitchWatch*, es decir, hacer clic izquierdo mientras se sigue mirando. Una vez ha alcanzado su tope, se debe volver a pulsar el botón izquierdo para

provocar el desplazamiento en sentido contrario, si procede. Se puede colocar en el punto deseado simplemente dejando de mirarlas o dejando de hacer clic.

*PlatformSwitchA* tiene accionadores en las dos caras, mientras que *PlatformSwitchC* solo en una, esto provoca que su activación, por parte del jugador, solo pueda realizarse con rango de visión a dicha cara. Por la situación de sus activadores, estas dos plataformas son utilizadas para bloquear el acceso y no para desplazar al personaje. *PlatformSwitchB* contiene un activador a mayor altura, cuya utilidad es desplazar al personaje cuando está sobre la plataforma, por ser de activación más accesible sobre ella.

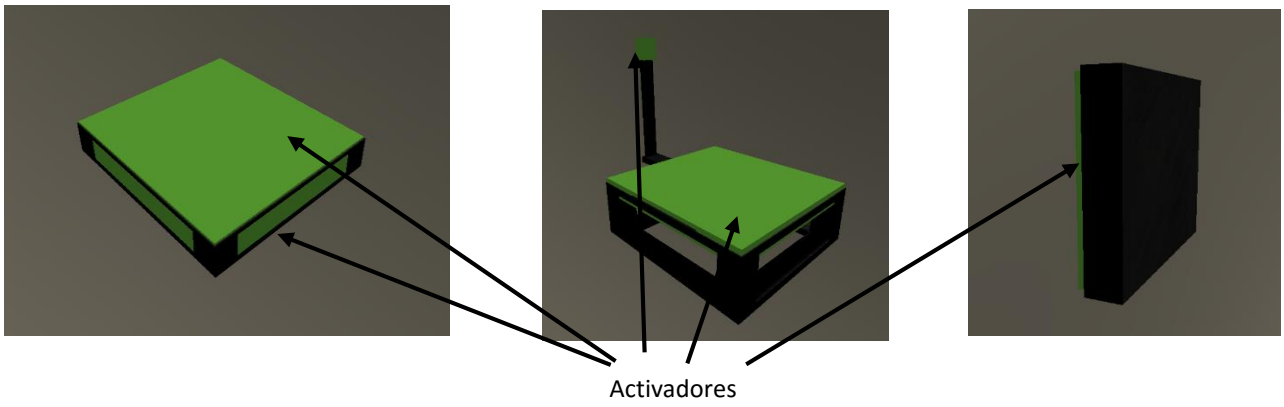



Figura 33: PlatformSwitchA, PlatformSwitchB y PlatformSwitchC.

### 3.2.2 ElevatorA y ElevatorB

Los ascensores sirven para alcanzar lugares inaccesibles. Contienen una *PlatformSwitchB* por tanto su activación es de la misma manera que esa plataforma.

La diferencia con las plataformas es que contienen elementos para que el jugador pueda colocarse sobre ellos de forma más sencilla y que su eje de movimiento es el 'Y'.

*ElevatorB* contiene paredes que provocan que el jugador solo pueda activarlo desde determinadas zonas. Es la diferencia entre *ElevatorA* y *ElevatorB*.

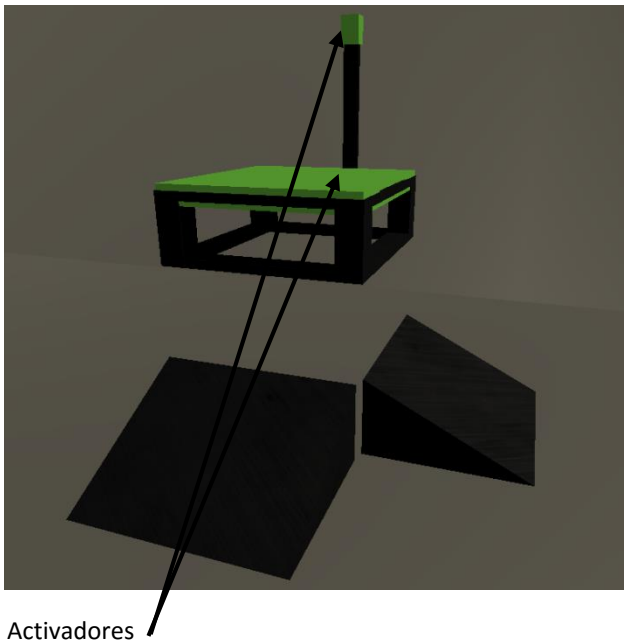



Figura 35: ElevatorA.

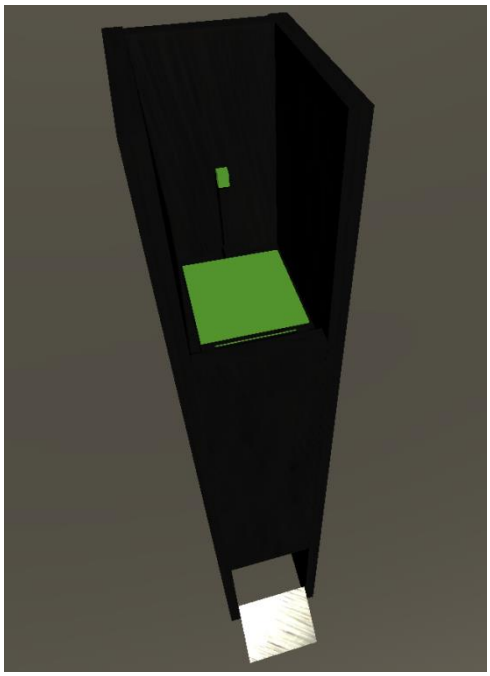


Figura 34: ElevatorB.

### 3.2.3 ObstacleA

Este objeto es utilizado para dificultar el paso al jugador por determinadas zonas. Se trata del clásico bloque que se desplaza continuamente a través de un recorrido prefijado. No requiere de activación ni desactivación por parte del jugador

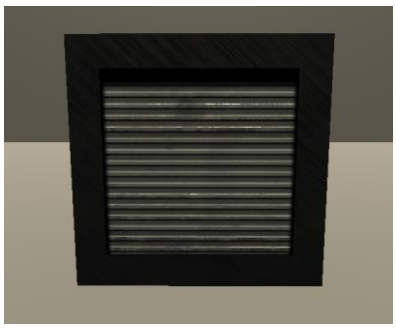


Figura 36: ObstacleA.

### 3.2.4 PlatformA

Consiste en la clásica plataforma móvil que el jugador tendrá que utilizar para llegar a un punto inaccesible de la escena. Se desplaza continuamente a través de un recorrido prefijado. Tampoco requiere de activación ni desactivación por parte del jugador.

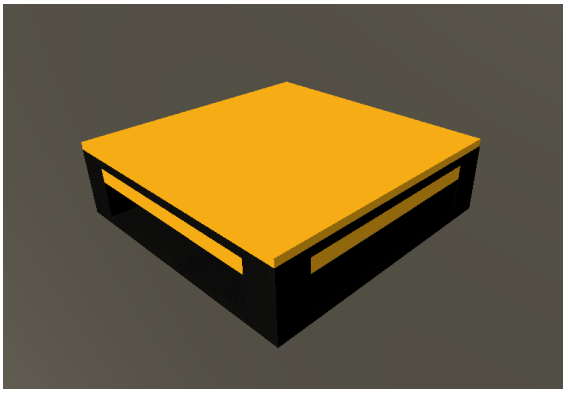


Figura 37: PlatformA.

## 3.3 Otros elementos

### 3.3.1 GateA

Esta puerta depende de un interruptor que debe estar activado para abrirla.

En la siguiente ilustración se muestra un ejemplo con el interruptor *SwitchA*.

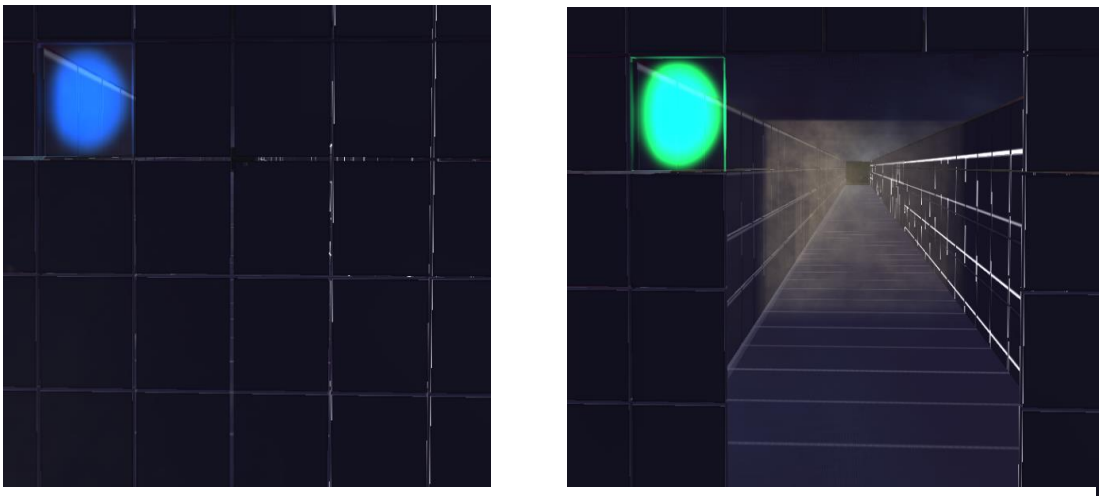


Figura 38: GateA estados: cerrada y abierta.

### 3.3.2 Inventario

El inventario es intrínseco al personaje y su funcionalidad es muy sencilla, pero se ha desarrollado con la intención de aumentar sus posibilidades en el futuro. En estos momentos lo único que se puede almacenar en él es un elemento que en el próximo capítulo se describirá.

# Capítulo 4. Entorno 3D

La escena que se ha diseñado para el juego consta de tres partes principales: Contenedor de Inicio, Contenedor Pirámide y Contenedor del Árbol.

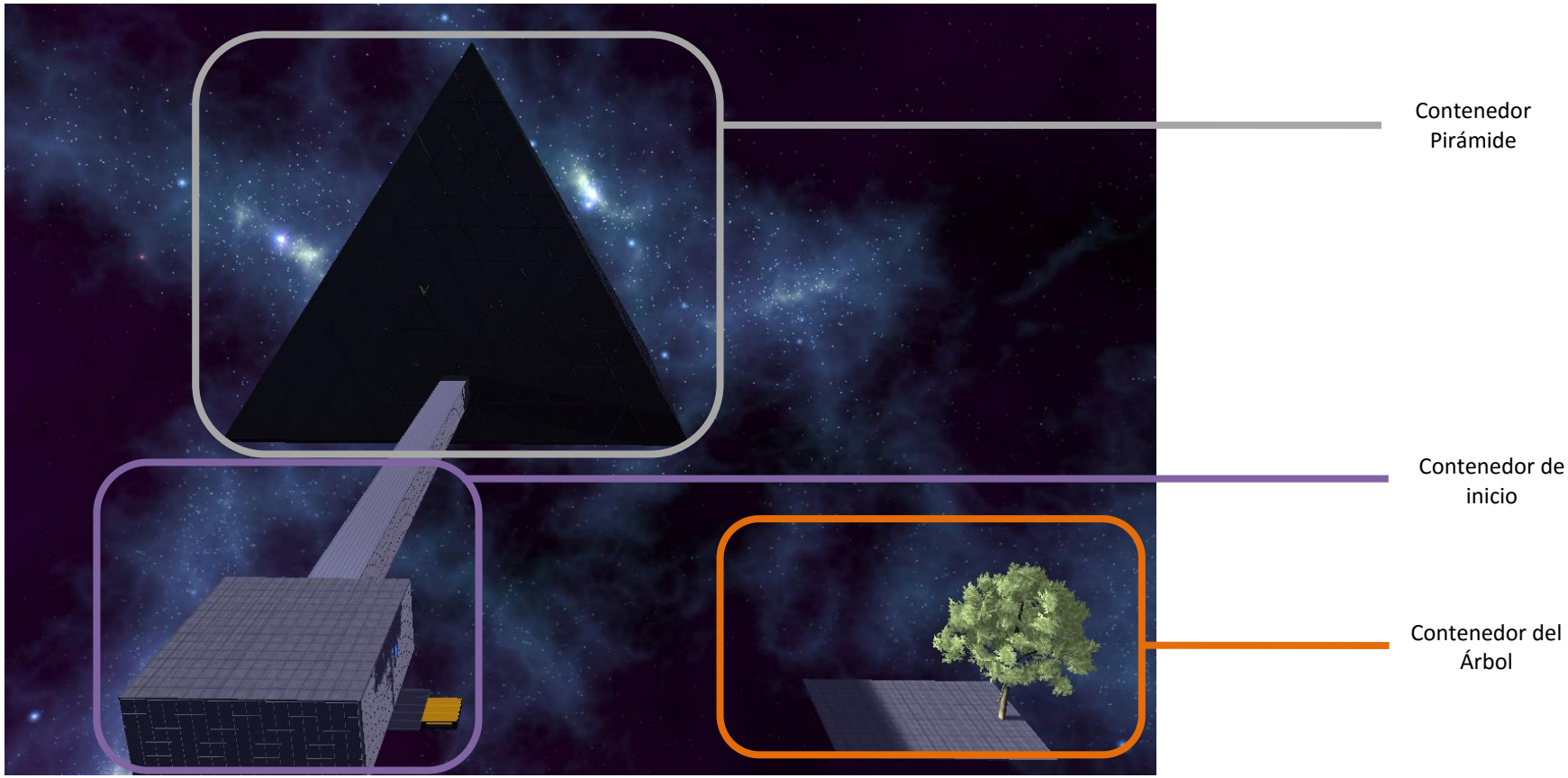



Figura 39: Captura del juego con la diferenciación de partes principales.

Cada contenedor está compuesto por puzles y elementos que se describirán por orden de activación, aunque no por ello es el orden en el que se lo encontrará el jugador. La escena en sí es un gran puzle compuesto por otros más pequeños que deben ser resueltos para tener éxito en la partida.

Los contenedores son los elementos más importantes de la escena, aunque también existen otros objetos decorativos diseñados con el objetivo de darle un aspecto espacial al juego. Entre ellos se encuentra, como fondo, una galaxia que se observa lejana, mientras que como elementos cercanos se hallan un asteroide y dos estrellas de donde se obtiene la luz; una está colocada al norte de la escena y la otra, de menor tamaño, al sur.

> *Para una descripción más clara, en algunas figuras se han aislado los elementos señalados del resto de la escena.*

## 4.1 Contenedor de inicio

Este contenedor consta de los siguientes objetos interactuables: dos *SwitchA* y dos *GateA*. También contiene un *Teleportador*.

El personaje se encuentra en el centro de esta habitación, justo debajo del *Teleportador,* que al ser activado mostrará una animación y activará su *trigger* para el final del juego cuando se hayan resuelto todos los puzles de la escena. Cada *SwitchA* activa la *GateA* más cercana. Al abrir *GateA 1* se encontrará un pasillo que lleva hasta *Contenedor Pirámide*. Desde *GateA 2* el jugador puede acceder a *Contenedor del Árbol*.

Las paredes de esta estancia están construidas por un conjunto de cubos con una pequeña separación entre ellos dejando pasar haces de luz y sus efectos, de manera que proyecta sombras en el suelo y en las demás paredes.

En la siguiente ilustración se muestra la vista de planta del contenedor inicial:

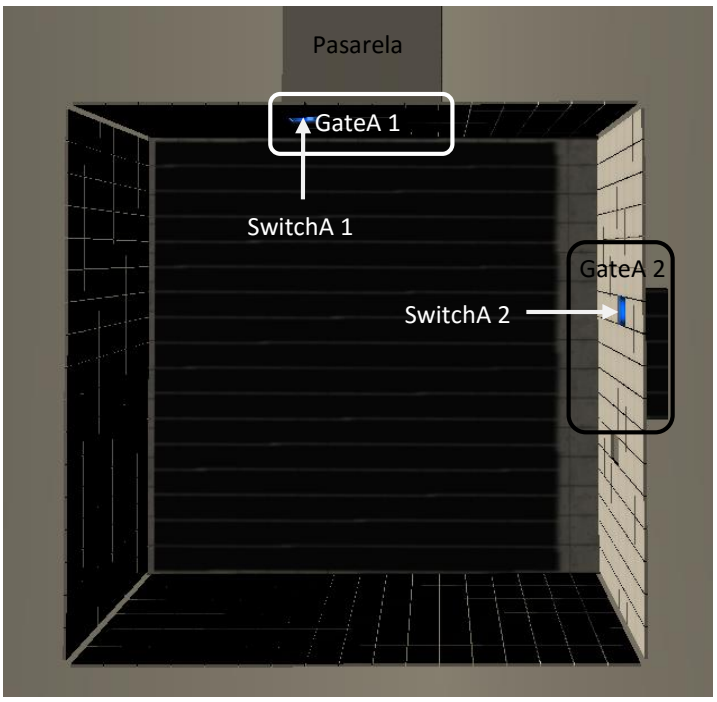



Figura 40: Vista de planta del contenedor de inicio.

## 4.2 Contenedor de Pirámide

Es el mayor contenedor del juego y consta de un *ElevatorB*, tres *ElevatorA*, tres *ObstacleA*, cuatro *SwitchB*, cinco *PlatformA*, y cuarenta y dos *PlatformSwitchC*. También contiene un obelisco, cuatros estatuas de dioses egipcios, entre otros elementos decorativos, pero que a la vez sirven de pistas para resolver los acertijos.

El objetivo del jugador en este contenedor es alcanzar una manzana que se encuentra en la cima del obelisco. La manzana no se encuentra visible ni accesible desde el comienzo de la partida. Para poder cumplir el objetivo, se ha de resolver cuatro puzles y superar una serie de plataformas que serán descritos a continuación.

Las paredes de esta estancia están construidas de la misma manera y con el mismo objetivo que las del contenedor de inicio, pero en vez de utilizar cubos se han utilizado prismas de base triangular.

A continuación, se muestra un esquema general del interior de la pirámide con los puzles y series de plataformas:

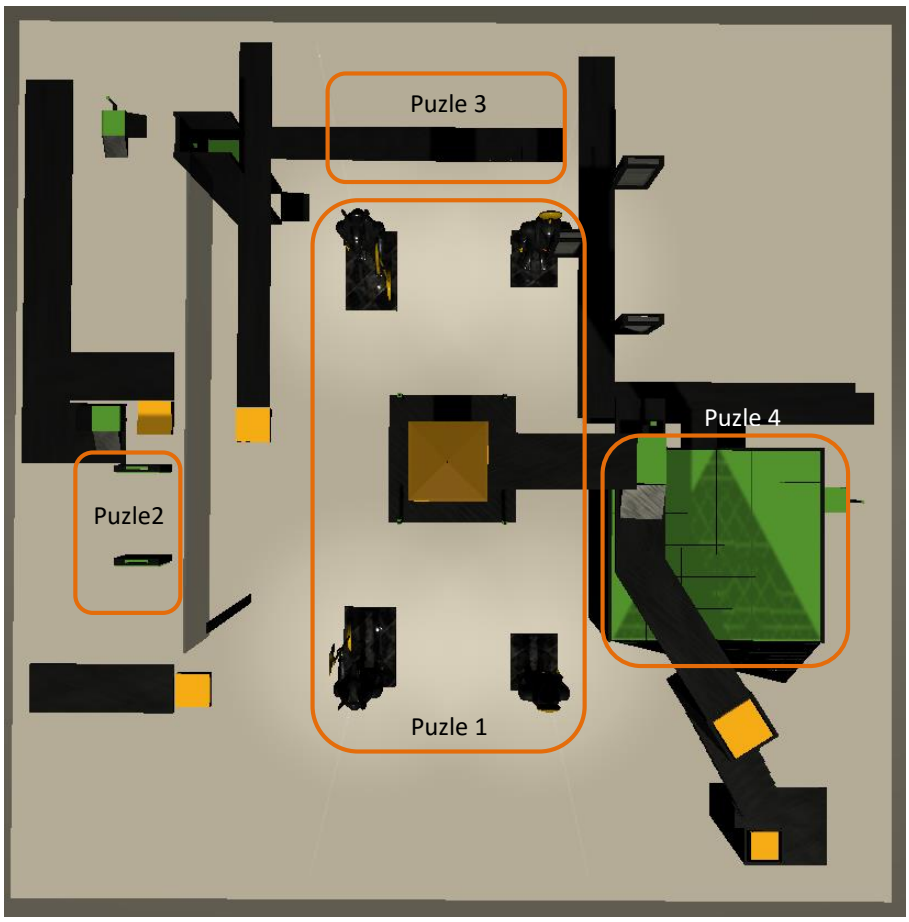



Figura 41: Vista de planta de la pirámide.

El acceso a la pirámide se hace desde una pasarela que la conecta con el *Contenedor de Inicio*. A continuación, el jugador deberá descender hasta la base de la pirámide por medio de un *ElevatorA*.

### 4.2.1 Puzle 1

En la base de la pirámide se encuentra el primer puzle. Consta principalmente de cuatro *SwitchB*, donde sus *Receptores* están posicionados a la altura del pecho de la estatua más cercana al interruptor. Un *haz de luz* que llega desde el exterior de la pirámide se proyecta sobre cada estatua. Esto sirve de pista para resolver el acertijo.

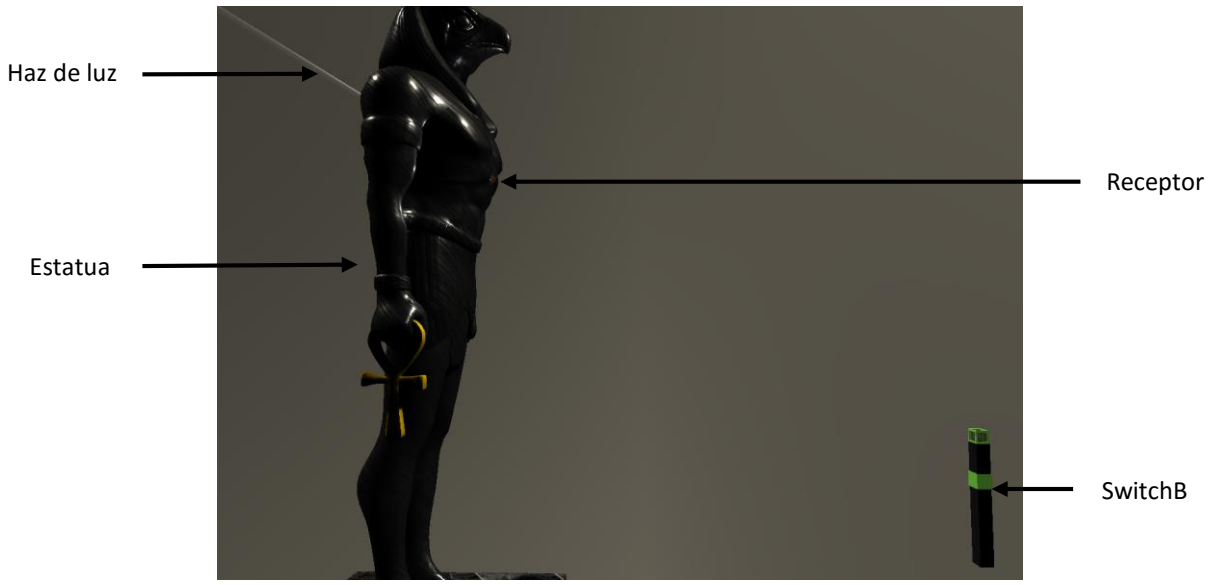



Figura 42: Puzle 1: SwitchB y decorado.

En el centro del puzle se encuentra el obelisco y en lo alto de este la manzana, que solo será visible si el jugador resuelve el puzle, y con ello activa un haz de luz que se proyecta sobre ella.

Figura 43: Elementos del Puzle 1.


Figura 44: Captura Puzle 1, cima del obelisco.

### 4.2.2 Puzle 2

Su resolución debe realizarse desde la base de la pirámide y consta de dos *PlatformSwitchC*.

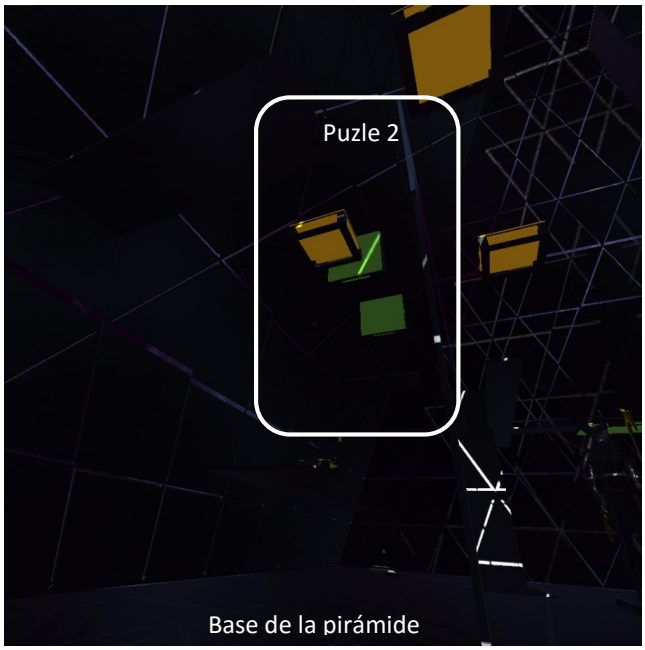



Figura 45: Captura vista del jugador del puzle 2.

El jugador debe desplazar cada una de las *PlatformSwitchC* hasta su límite máximo de altura, de la manera que ha sido descrita en el capítulo anterior en el apartado de este elemento. Esto provoca que el personaje pueda moverse por debajo de ellas y llegar al otro extremo mediante una *PlatformA*.

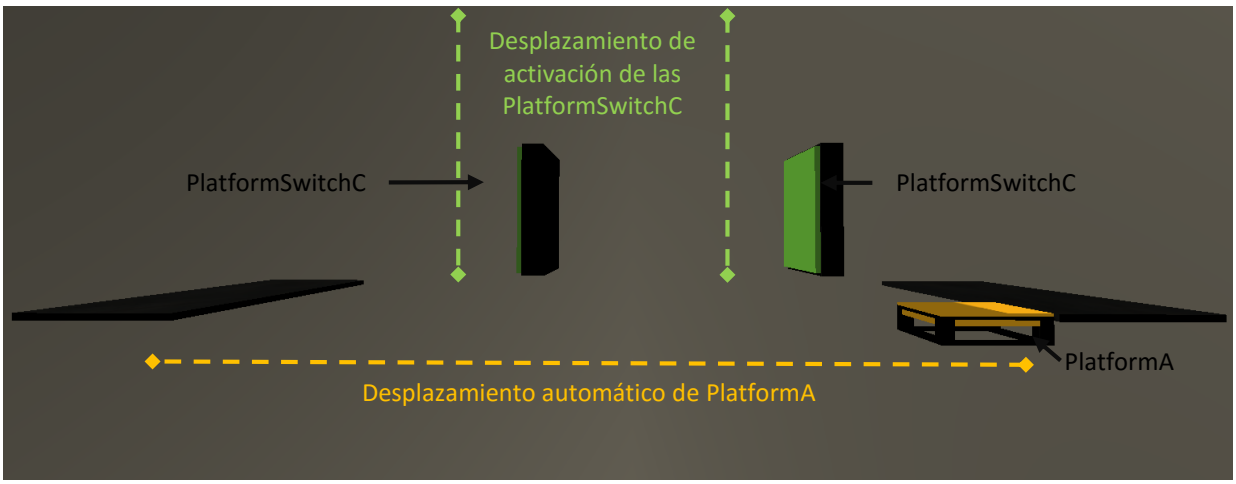



Figura 46: Esquema de funcionamiento del puzle 2.

### 4.2.3 Puzle 3

Su resolución debe realizarse desde la base de la pirámide y consta de cinco *PlatformSwitchC*.


Figura 47: Captura vista del jugador del puzle 3.

El jugador debe desplazar las *PlatformSwitchC* ajustando la altura de cada una de manera que formen una escalera por la que pueda subir.

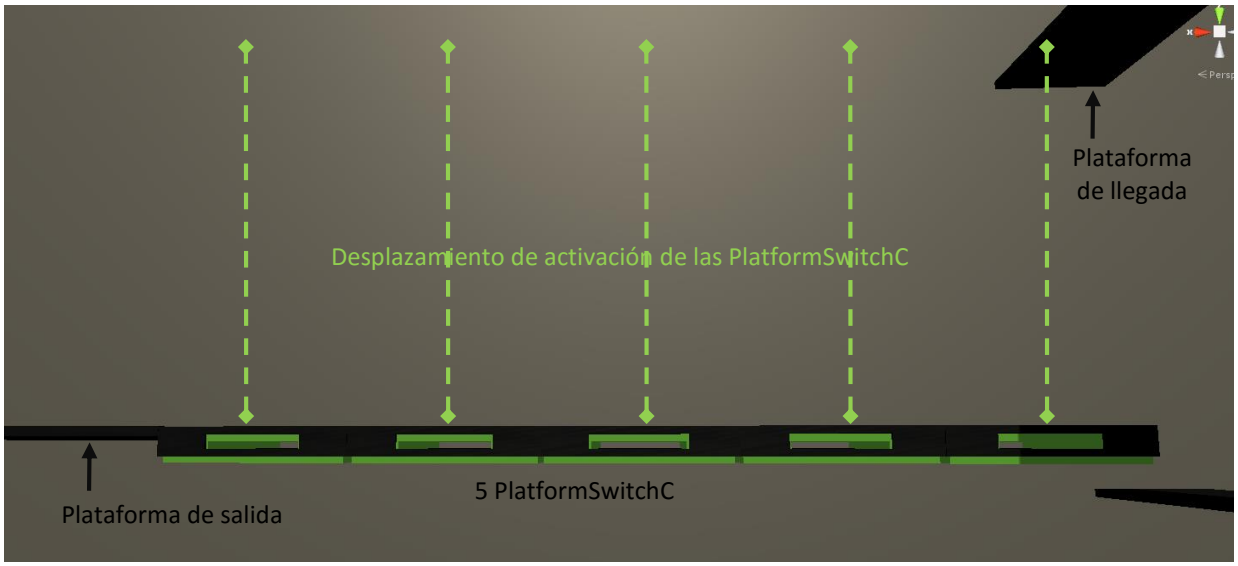



Figura 48: Esquema de funcionamiento del puzle 3.

Una de las formas de resolución del puzle es la que se muestra en la siguiente ilustración:

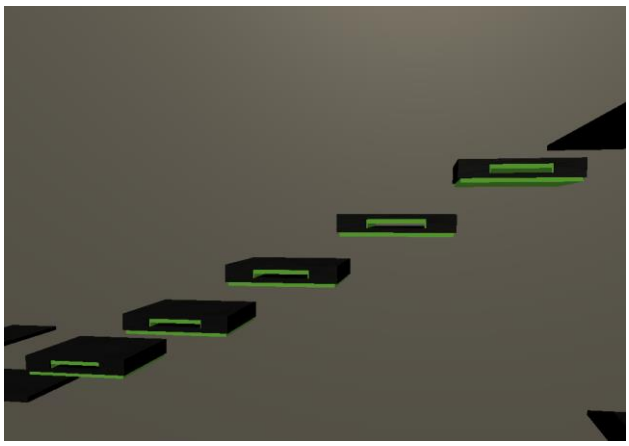


Figura 49: Puzle 3 resuelto.

### 4.2.4 Serie de plataformas 1

Consta de dos *ElevatorA*, una *PlatformA*, del P*uzle 2* y otras plataformas estáticas. El primer *ElevatorA* se encuentra en la base de la pirámide y el final de la serie conecta con el principio de la serie de plataformas 2. El jugador deberá tomar el camino que se muestra en la siguiente ilustración para alcanzar la siguiente serie de plataformas.

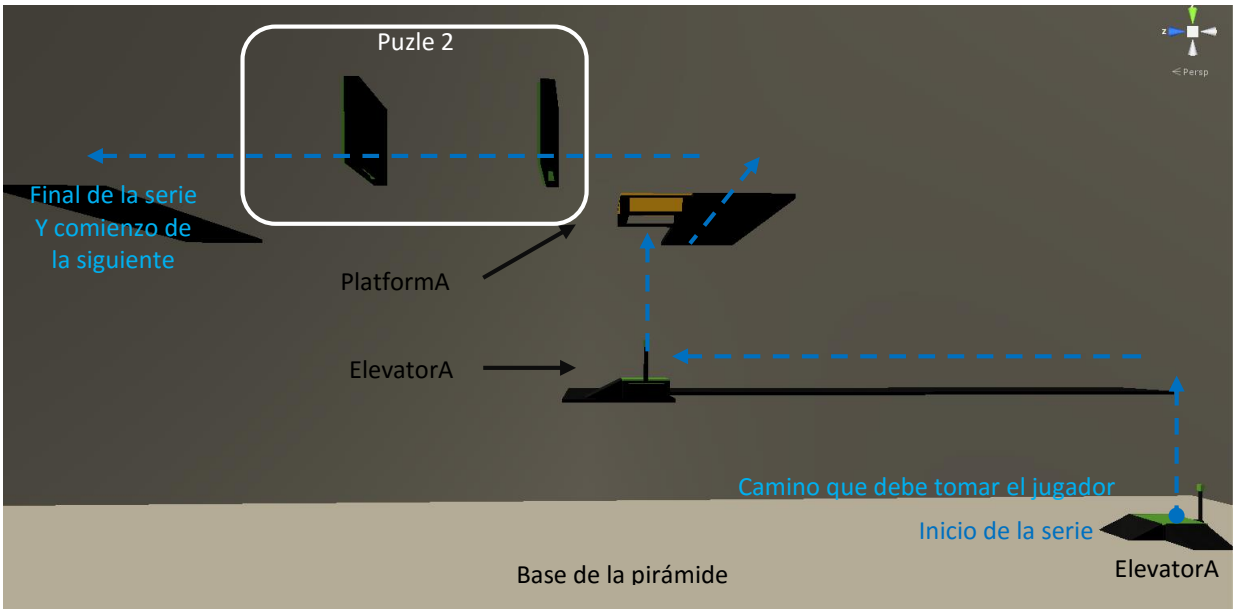



Figura 50: Esquema de la serie de plataformas 1.

### 4.2.5 Serie de plataformas 2

Consta de dos *PlatformA*, un *ElevatorB*, del *Puzle 3*, además de otras plataformas estáticas. El *ElevatorB* conecta con la base de la pirámide y solo puede ser activado (por su morfología) una vez se haya llegado a esta serie de plataformas. El comienzo de la serie conecta con la *Serie de plataformas 1* y al final de la trayectoria se encuentra la *Serie de plataformas 3*.

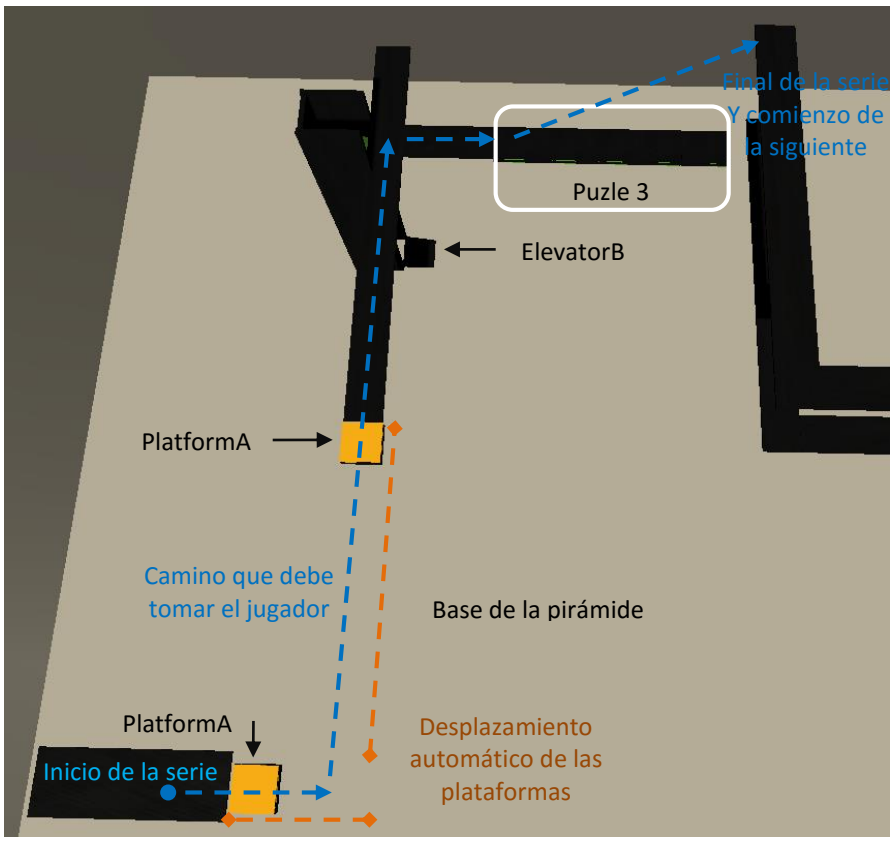



Figura 51: Esquema de la serie de plataformas 2.

### 4.2.6 Serie de plataformas 3

Consta de tres *ObstacleA* que dificultan el paso por las plataformas al jugador. Cada uno de los obstáculos se desplaza a mayor velocidad que el anterior. El comienzo conecta con la *Serie de plataformas 2* y el final con el *Puzle 4*.

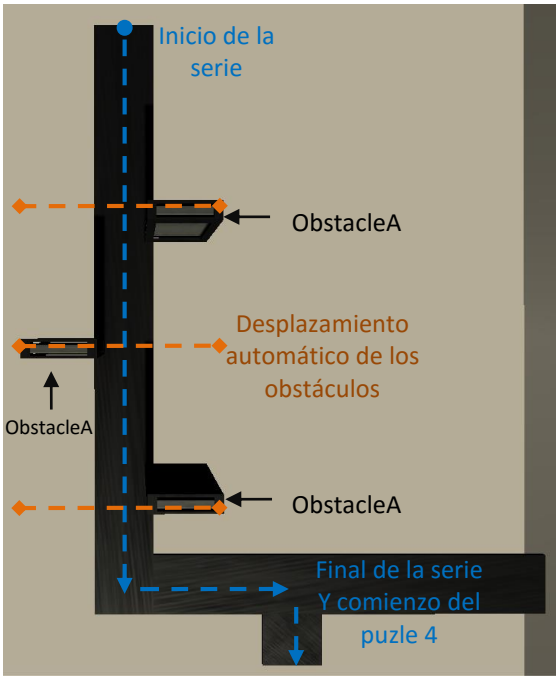



Figura 52: Esquema de la serie de plataformas 3.

### 4.2.7 Puzle 4

Consta de treinta y seis *PlatformSwitchC* y plataformas y paredes estáticas. Cada uno de los *PlatformSwitchC* tiene unos límites de movimiento diferentes. El objetivo en este puzle es acceder a la *PlatformSwitchB* que se encuentra oculta debajo de las *PlatformSwitchC*. Para ello, el jugador ha de desplazarlas hasta que la plataforma móvil quede al descubierto. El inicio del puzle conecta con la *Serie de plataformas 3* y el final con la *Serie de plataformas 4*.

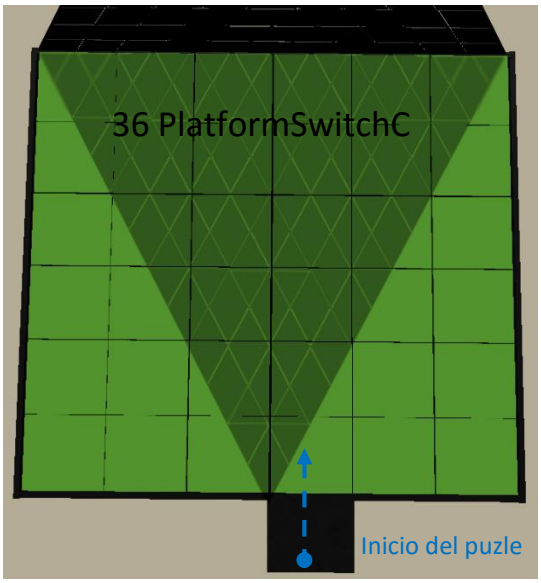



Figura 54: Esquema del puzle 4.

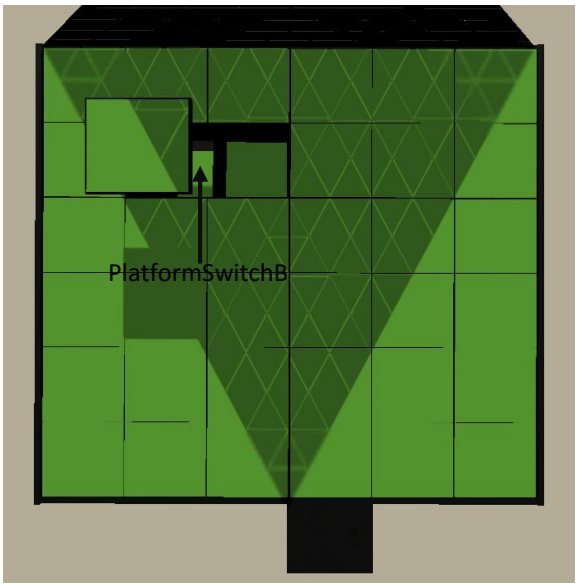



Figura 53: Puzle 4 resuelto.

### 4.2.8 Serie de plataformas 4

Consta de una *PlatformSwitchB*, dos *PlatformA* y un *ElevatorA*. El inicio conecta con el *Puzle 4* en una caída del personaje y el final es donde se encuentra la cima del obelisco con la manzana que el jugador debe recoger.

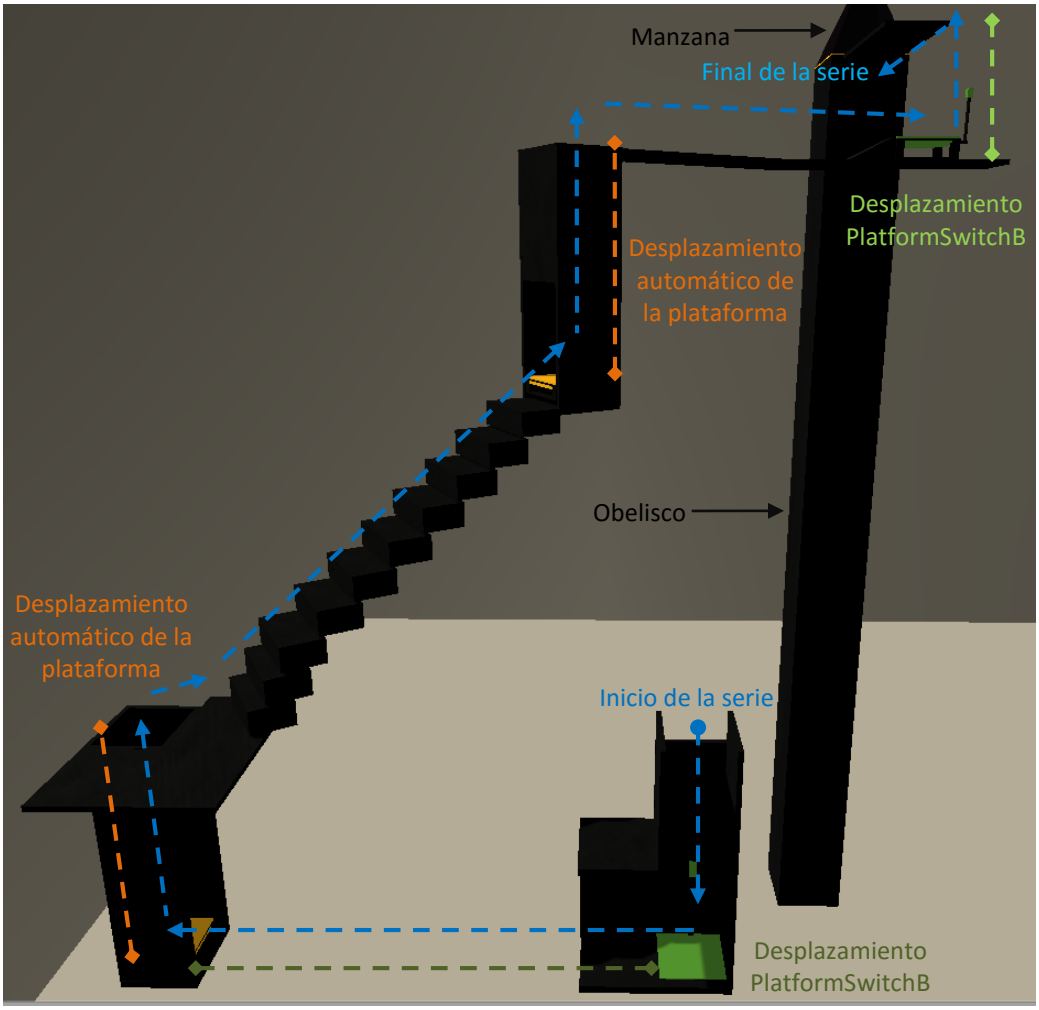



Figura 55: Esquema de la serie de plataformas 4.

Cuando el jugador tenga la manzana en su inventario debe salir de la pirámide por el mismo lugar que por donde entró y dirigirse al **Contenedor del Árbol.**

## 4.3 Contenedor del Árbol

Este contenedor consta de una *PlatformA* que lo conecta con el contenedor de inicio, un árbol y varias plataformas estáticas. Para desenlazar el final de este juego, el jugador debe tener la manzana en su poder y hacer clic sobre una determinada rama del árbol. De esa manera colocará la manzana pendiendo de la rama y haciendo sombra sobre un interruptor (*SwitchTeleport*) en el contenedor de inicio. De nuevo, haciendo clic sobre *SwitchTeleport,* el Teleportador se activará y si el jugador sitúa al personaje debajo de este, terminará el juego.

Una vista de planta de este contenedor es la que se muestra en la figura siguiente:

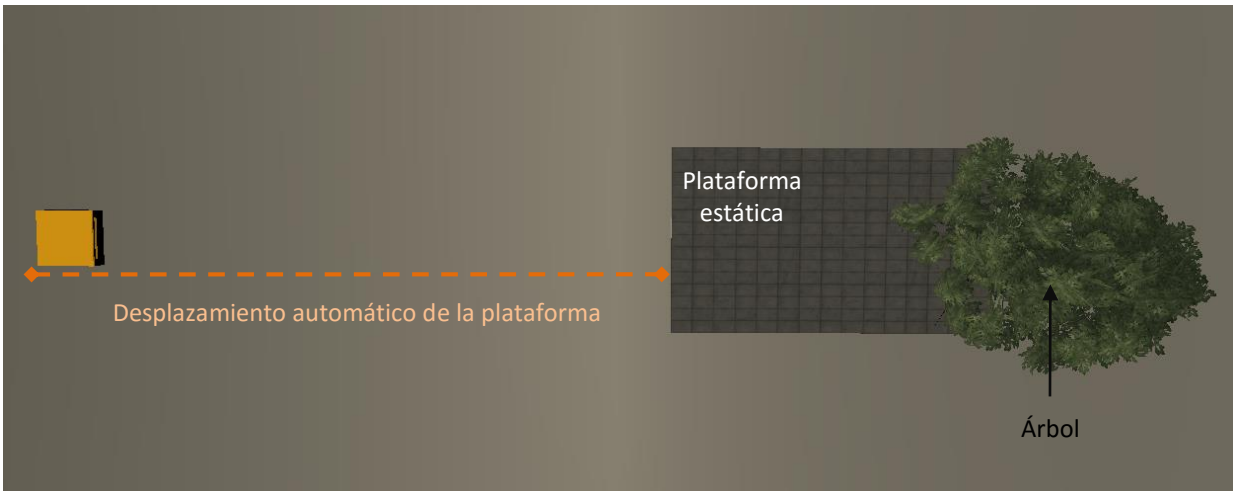



Figura 56: Vista de planta del contenedor del Árbol.

A modo de pista, se ha colocado el tallo de la manzana en la rama donde el jugador debe hacer clic. La siguiente ilustración lo muestra:

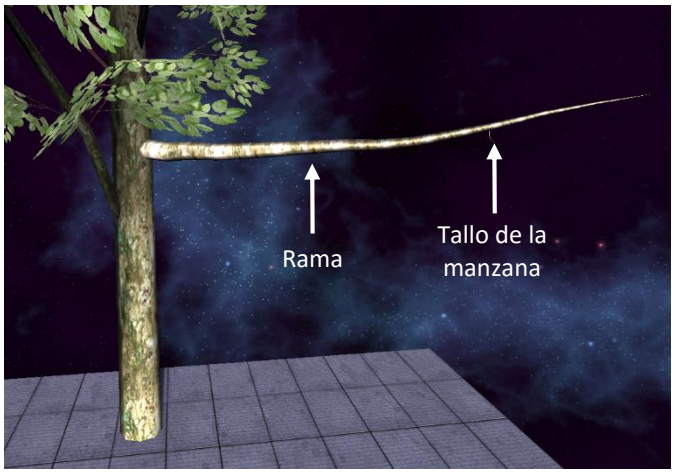



Figura 57: Captura de la rama del árbol.

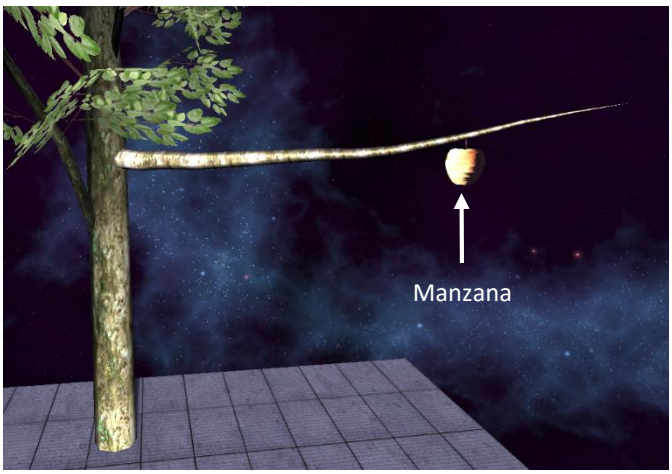



Figura 58: Captura de la rama del árbol y la manzana.

# Capítulo 5.
# Conclusiones y líneas futuras

## 5.1 Conclusiones

En las fases iniciales de este proyecto se definieron dos objetivos principales.

El primero de ellos era investigar nuevas tecnologías de interacción persona-computador en el ámbito de los videojuegos. A día de hoy, la tecnología de la Realidad Virtual aún no está suficientemente evolucionada. Por ejemplo, el tamaño de los pixeles en el HMD de *Oculus* se deja notar demasiado ante nuestros ojos, es aún demasiado pesado para un tiempo prolongado de juego, la respuesta de los sensores no son los suficientemente rápidos, y además se necesita una computadora potente para poder ejecutarlo con suavidad. Como jugador, pueden ser realmente frustrantes los mareos provocados ante sesiones de apenas 20 minutos y se tarda meses en acostumbrarse. Sin embargo, hay multitud de empresas que actualmente están invirtiendo una cantidad considerable de recursos en desarrollarla y ya se pueden encontrar grandes avances en los métodos y tecnologías. Durante 2016 se verá que la mayoría comercializan sus versiones finales y es cuando se podrá empezar a medir su verdadero potencial.

El segundo objetivo consistía en crear un prototipo de videojuego en primera persona. Y dentro de este, tres líneas de desarrollo diferentes: un controlador del personaje, un modo de interacción con los elementos del juego y un entorno 3D.

En primer lugar, se ha implementado un controlador del personaje lo más realista y cómodo posible para el jugador. Esta línea se divide en dos apartados:

Por un lado, se ha elaborado un cuerpo para el personaje virtual, que incluye sus animaciones. En ese sentido, donde se encuentran las verdaderas dificultades es al hacer coincidir el cuerpo virtual con el físico, ya que los sensores que contiene Oculus Rift no son suficientes, tanto en cantidad como en su naturaleza. Aunque para este proyecto se ha integrado un cuerpo virtual, no resulta del todo perfecto por lo dicho anteriormente.

Por otro lado, se ha desarrollado el modo de funcionamiento, que consiste principalmente en el manejo del personaje procurando suavizar los movimientos; se ha conseguido paliar la sensación de mareo. Sin embargo, no es la solución definitiva para este tipo de juegos, ya que el malestar depende sobretodo de la calidad de los sensores y cómo responden al movimiento de la cabeza.

Además, se han desarrollado los mecanismos de interacción: el control de la acción se realiza con el ratón, mientras que con el HMD se maneja de forma natural la cámara. Con ello se ha conseguido enriquecer la cantidad de acciones que se pueden realizar en el juego. Ya no es preciso mirar un elemento para poder realizar una acción sobre él. Podría ser una fórmula alternativa de interacción con el juego y una solución a corto plazo al problema del

importe del sistema, ya que se utilizan el HMD juntamente con dispositivos clásicos que están en todo ordenador como son el teclado y el ratón, incluso fácilmente ajustable a un *Gamepad*.

En la segunda línea de desarrollo, el modo de interacción con los objetos del juego es un ejemplo sencillo y directo de manejo del dispositivo HMD y de la jugabilidad propuesta. El usuario debe utilizar tanto su mirada como el ratón, para realizar las acciones sobre los objetos con los que puede interactuar.

Por último, se ha conseguido una atmósfera envolvente con el entorno 3D que se ha creado, muy apreciable en juegos de primera persona y particularmente de Realidad Virtual. Se ha tenido especial cuidado en la iluminación y sombras que se proyectan en la escena, así como en las proporciones de los objetos tanto decorativos como interactuables.

La investigación y desarrollo de nuevas tecnologías en el ámbito que me corresponde, ha resultado una experiencia gratificante y adictiva. La involucración en el detalle es lo que da vida a estas creaciones y su resultado es inspirador. Sin duda alguna, se continuará con el desarrollo de este prototipo y otros potenciales proyectos.

## 5.2 Conclusions

During the initial phases of this project two main targets were defined.

The first target was to research new human-machine interaction technologies in the video game field. To this day, Virtual Reality technology is not developed enough. For instance, the HMD by Oculus is still too heavy for a long playtime, the size of pixels stands out too much before our eyes, the sensors response is not fast enough, and a powerful computer is needed in order to execute it smoothly. As a player, it can be really frustrating to feel motion sickness (Virtual Reality sickness) after playing only for 20 minutes and it takes months to get used to it.

However, dozens of companies are investing a considerable amount of resources in developing this technology and have already made great progress in the task. Over the course of 2016, those companies will release their hardware final versions to consumers and it will be time to start measuring its real potential.

The second one consisted in creating a first person video game prototype, and within this, three different lines of development: a character controller, a method of interacting with the game elements, and a 3D environment.

In the first place, it has been implemented a character controller as realistic and comfortable as possible for the player. This line is divided into two parts:

For starters, a body and animations for the virtual character have been created. In that sense, the real difficulty is to synchronize the virtual body with the physical one considering that the sensors the Oculus Rift contains are insufficient in number and type. Although a

virtual body has been integrated for this project, it has turned out to be imperfect on account of the already stated.

In the last part, the functioning mode has been developed, which mainly consist in procuring smooth movements for the character and therefore reducing the motion sickness. Still it is not a definitive solution for virtual sickness, since it especially depends on the sensors quality and the way they respond to the head movement.

In addition, the interaction mechanisms have been developed: the action is controlled with the mouse, while the camera is operated by the user's head movements in a natural way. With this, it has been achieved an improvement in the number of actions that can be carried out in the game; it is not necessary to look at an element in order to trigger the action. It could be an alternative formula of game interaction and a short term solution to the system's price issue, since the HMD is used together with the classic devices found in every PC like the keyboard and the mouse, and it is easily adjustable to a gamepad.

In the second line of development, the interaction method with the game elements is a simple and direct example of the use of the HMD device and proposed playability. The user must utilize his/her sight as much as the mouse, in order to carry out the actions over the interactive objects.

Finally, thanks to the 3D environment created, it has been achieved an enveloping atmosphere, a much appreciated feature in first person games generally and Virtual Reality games particularly. The work performed on illuminations and projected shadows for the game scene has been especially meticulous, so has been the treatment of proportions in decorative objects as well as in interactive objects.

The research and development of new technologies within my area of expertise, has been an addictive and rewarding experience. To care about details is what gives life to these creations and its result is inspiring. Beyond any doubt, the development of this prototype and other potential projects will be continued.

## 5.3 Líneas futuras

El prototipo ya es una demo jugable de lo que se quería conseguir en este proyecto, no obstante, todavía está lejos de convertirse en una aplicación para el gran público. Las líneas que deberían tratarse en primera instancia y en segunda instancia, las enfocadas a la jugabilidad, son:

- Resulta poco natural avanzar con el personaje hacia adelante (mediante el teclado) y tener que corregir su posición lateral avanzados unos metros. Una posible solución es realizar ese movimiento siguiendo el vector de trayectoria del HMD dentro de unos límites laterales izquierdo y derecho. Así el cuerpo y por tanto la dirección, se determinarán por la cabeza del jugador. Si se detectara un movimiento por encima de

los límites, el movimiento del personaje obedecería a la dirección recogida del teclado, como es ahora lo habitual.

- Como se ha descrito anteriormente, en ocasiones hay que reiniciar el juego si en primera instancia no recoge correctamente los datos de los sensores de la posición global. Se ha observado que en algunas aplicaciones comerciales ocurre de forma similar, ya que se trata de un problema de drivers en su versión beta 0.7.0.0. Sin embargo, se puede mejorar la respuesta inicial de posición global para este juego.

- Según la destreza del jugador, se puede perder el foco del *Crosshair de acción*. Para solucionar este defecto, se debe implementar un mecanismo para devolverlo al centro de la vista.

- Aprovechar la aportación de los nuevos drivers para mejorar la posición del cuerpo virtual respecto al físico del jugador, si fuera posible.

En segunda instancia, en vistas a mejorar la experiencia jugable:

- Utilizar el algoritmo de cinemática inversa FABRIK para realizar la animación del brazo derecho del personaje. Este movimiento consistiría en reproducir el movimiento del brazo humano al tocar o alcanzar un objeto, y sería vinculado a la acción de clic izquierdo con el ratón (Acción del personaje).

- Agregar nuevas escenas y puzles que aprovechen en mayor profundidad el HMD. Actualmente solo existe uno acabado; se ha dejado otro incompleto y su diseño ya se ha realizado sobre papel.

- Implementación de un menú de inicio y configuración. Donde poder ajustar los controles, incluso las velocidades de rotación y desplazamiento para una mayor comodidad para el jugador. También podría contener inicio y recuperación de partida.

- Mejorar la interfaz de usuario. Implementación de menús con indicadores de la situación del jugador, inventario y puzles.

- Mejorar el aspecto visual. La mayoría de texturas son simples y planas, y los objetos son de estructura sencilla.

- Agregar sonidos, por ejemplo, al caminar, saltar, caer, accionar los interruptores, movimiento de las plataformas, etc. Actualmente el juego no consta de ningún sonido a excepción de la música ambiental.

- Implementación de objetos o armas virtuales que provocasen que el jugador tuviera que colocarse de determinadas maneras para poder utilizarlos o mejorar su precisión. Esto daría un uso exclusivo al posicionamiento global.

- Mejorar las animaciones del personaje para que parezcan más naturales.

A largo plazo, los objetivos que podrían encajar en este proyecto son:

- Implementación de otra tecnología como Microsoft Kinect para mejorar la relación entre el cuerpo físico y el cuerpo virtual.

- Implementación de gestor de datos estadísticos sobre la forma en la que afronta la escena el jugador para crear perfiles de usuarios.

- Desarrollo de una inteligencia artificial que utilice el gestor de datos estadísticos para ofrecer variantes de la escena al jugador según su perfil.

- Desarrollo de una inteligencia artificial con aprendizaje automático, que utilice el gestor de datos estadísticos para resolver los puzles y que haga de apoyo o rival al jugador según sus necesidades.

# Capítulo 6. Presupuesto

El proyecto está dividido en una serie de fases diferenciadas, que en este apartado se han asociado al tiempo en horas utilizado para completarlas, y al coste económico que asciende a 20€/h. La siguiente tabla muestra una estimación del coste en el caso de que se llevara a cabo este proyecto como una actividad remunerada por un ingeniero informático.

| Fases | Duración (h) | Importe (€) |
|---|---:|---:|
| Investigación | 45 | 900,00 |
| Análisis | 60 | 1.200,00 |
| Desarrollo | 240 | 4.800,00 |
| **Total** | **345 h** | **6.900,00 €** |

Tabla 5: Estimación de coste del proyecto.

En la siguiente tabla se muestra los costes de los principales componentes de PC, como hardware mínimo para poder utilizar el software desarrollado.

| Materiales | Importe (€) |
|---|---:|
| Procesador Intel i5-4560 o equivalente | 170,00 |
| RAM 8GB | 60,00 |
| Tarjeta gráfica NVIDIA GTX 970 o AMD R9 290 | 370,00 |
| Oculus Rift DK 2 y gastos de envío | 450,00 |
| **Total** | **1050,00 €** |

Tabla 6: Costes de materiales.

# Apéndice A

# Descripción de los procedimientos del controlador del personaje

A continuación, se describirán los procedimientos que han sido nombrados en este capítulo, pero en forma de más bajo nivel en cuanto a implementación.

> *Para poder hacer una descripción más precisa y detallada, las clases desarrolladas que intervienen directamente en los procedimientos expuestos se representan en color azul y con tipo de letra monospace, ejemplo:* Class.
>
> *Los procedimientos, funciones y variables se representarán en color negro y con el mismo tipo de letra, ejemplo:* procedure.
>
> *Las cadenas de constantes literales se representarán en color rojo, entre comillas dobles y con el mismo tipo de letra, ejemplo:* "const".
>
> *Las asignaciones a variables en color azul y con el mismo tipo de letra, ejemplo:* true.

La clase desarrollada para obtener los objetos del cuerpo del personaje es `PlayerObjects`.

Es preciso describir los eventos de los scripts en *Unity* que se han utilizado:

- **`Awake:`** Este evento es llamado al iniciarse la aplicación y antes de cualquier evento `Start`. Si la instancia del script no está activa no es llamado hasta que lo esté. Debe colocarse el código que solo se ejecutará una vez al inicio de la aplicación. Si el juego se compone de varias escenas, el código solo se ejecutará una vez.

- **`Start:`** Se ejecuta en la primera actualización del frame[14] si la instancia del script está activa. Si el juego se compone de varias escenas, el código se ejecutará una vez en cada escena.

- **`Update:`** Es llamado en cada actualización de frame. En ese evento se escribe el código que debe ser ejecutado en cada frame.

Tales eventos se han utilizado en el desarrollo de todo el prototipo, no solo para lo involucrado en este capítulo.

[14] **Frame** es equivalente a fotograma.

### Ejes de coordenadas

Es importante conocer la colocación de ejes de coordenadas para poder entender el funcionamiento de algunos procedimientos descritos:

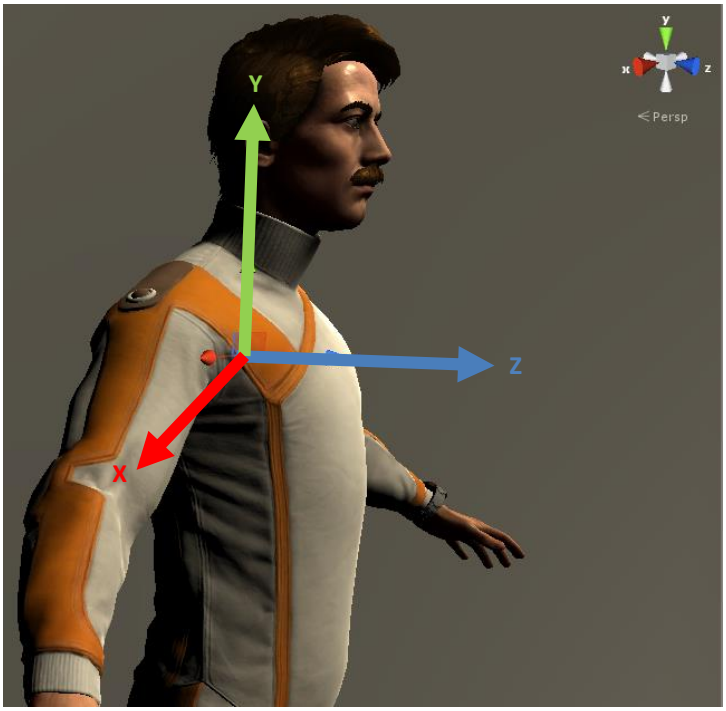



Figura 59: Captura de los ejes en Unity.

# Cuerpo del personaje

Para facilitar el desarrollo de las funciones del personaje, se han asignado variables con las direcciones de los objetos utilizados.

El *Controlador de personaje* (Character Controller) de *Unity* consiste en una cápsula de colisión[15] (*collider*) principalmente. Para que los parámetros de esa cápsula se ajustasen a la Realidad Virtual se ha llevado a cabo el procedimiento de ensayo y error, quedando configurado de la siguiente manera:

[15] **Cápsula de colisión** o ***collider*** es un componente lógico de *Unity* que da características y magnitudes físicas a objetos del editor. Representa los límites físicos, por ejemplo, límites de colisión. También representa el peso del objeto y el material del que está hecho.

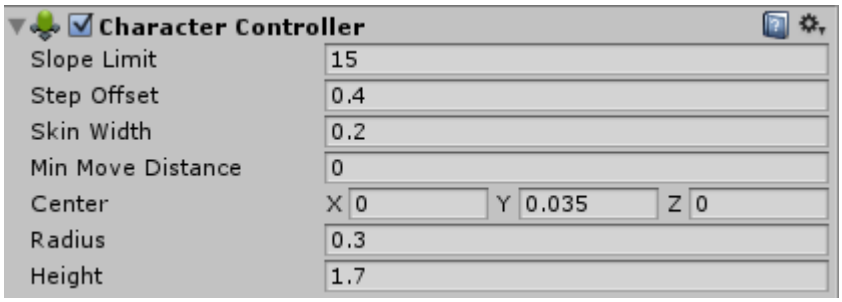



Figura 60: Parámetros del controlador del personaje.

Donde:

| Propiedad | Función |
|---|---|
| **Slope Limit** | Límite de la pendiente que el personaje podrá afrontar. |
| **Step Offset** | El máximo tamaño de escalón que podrá afrontar. |
| **Skin width** | Profundidad en la que puede ser penetrado por otro colisionador. |
| **Min Move Distance** | El mínimo espacio de movimiento. |
| **Center** | El centro de la cápsula de colisión respecto al personaje. |
| **Radius** | El radio de la cápsula. |
| **Height** | La altura de la cápsula. |

Tabla 7: Descripción de propiedades del controlador del personaje.

El elemento del editor de *Unity* se llama `PlayerAnimator`.

A continuación, se enumeran las acciones de las animaciones, las transacciones en entre ellas y su modo de funcionamiento:

| Acción | Funcionamiento | Transacciones |
|---|---|---|
| Idle<br>**Movimiento estacionario.** | Si no se acciona ninguna tecla, incluso si se está moviendo la cabeza o el ratón. | **Salientes**: Jump, Forward, Jump forward, Strafe left, Strafe right, Rotate left, Rotate right, Backward, Crouch Rotate left, Crouch Rotate right, y Crouch.<br>**Entrantes:** Jump, Forward, Jump forward, Strafe left, Strafe right, Rotate left, Rotate right, Backward, Crouch y Crouch Walk. |
| Forward<br>**Movimiento hacia adelante.** | Pulsando tecla de movimiento hacia adelante. Tiene dos velocidades: caminando y corriendo manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Idle, Jump forward y Crouch.<br>**Entrantes:** Idle, Jump forward, Strafe left, Strafe right, Rotate left, Rotate right, Backward y Crouch. |
| Backward<br>**Movimiento hacía atrás.** | Pulsando la tecla de movimiento hacia atrás. Puede retroceder con dos velocidades: caminando y corriendo manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Idle, Jump forward y Crouch.<br>**Entrantes:** Idle, Jump forward, Strafe left, Strafe right, Rotate left, Rotate right, Backward y Crouch. |
| Jump<br>**Movimiento de salto estacionario.** | Pulsando la tecla de salto mientras se está en modo **idle**. También se activa cuando el personaje cae. | **Salientes**: Idle, Strafe left, Strafe right, Rotate left y Rotate right.<br>**Entrantes:** Idle, Strafe left, Strafe right, Rotate left y Rotate right. |
| Jump forward<br>**Movimiento de salto con movimiento de avance.** | Pulsando la tecla de salto mientras se está **avanzando** o **retrocediendo**. También se activa cuando el personaje cae. Se puede aumentar la velocidad de desplazamiento en el aire manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Forward, Backward y Crouch.<br>**Entrantes:** Forward y Backward. |
| Strafe left<br>**Movimiento lateral hacia la izquierda.** | Pulsando la tecla de desplazamiento lateral izquierda. Se puede aumentar la velocidad de desplazamiento manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Idle, Jump, Forward, Backward y Crouch.<br>**Entrantes:** Idle y Jump. |
| Strafe right<br>**Movimiento lateral hacia la derecha.** | Pulsando la tecla de desplazamiento lateral izquierda. Se puede aumentar la velocidad de desplazamiento manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Idle, Jump, Forward, Backward y Crouch.<br>**Entrantes:** Idle y Jump. |
| Rotate left<br>**Movimiento de rotación hacia la izquierda.** | Pulsando la tecla de rotación izquierda. Se puede aumentar la velocidad de rotación manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Idle, Jump, Forward, Backward y Crouch.<br>**Entrantres:** Idle y Jump. |
| Rotate right<br>**Movimiento de rotación hacia la** | Pulsando la tecla de rotación derecha. Se puede aumentar la velocidad de rotación manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Idle, Jump, Forward, Backward y Crouch.<br>**Entrantes:** Idle y Jump. |

| | | |
|---|---|---|
| **derecha.** | | |
| Crouch<br>**Movimiento de agacharse y levantarse** | Pulsando la tecla de agacharse y si se encuentra agachado se levantará. | **Salientes**: Idle, Crouch Strafe left, Crouch Strafe right, Crouch Rotate left, Crouch Rotate right, Crouch Walk y Crouch Walk Backward.<br>**Entrantes:** Idle, Crouch Strafe left, Crouch Strafe right, Crouch Rotate left, Crouch Rotate right, Crouch Walk, Crouch Walk Backward Jump, Forward, Strafe left, Strafe right, Rotate left, Rotate right y Backward. |
| Crouch Walk<br>**Movimiento de avanzar agachado.** | Pulsando la tecla de avanzar mientras se está agachado. Se puede aumentar la velocidad de desplazamiento manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Idle y Crouch.<br>**Entrantes:** Crouch. |
| Crouch Walk back<br>**Movimiento de retroceder agachado.** | Pulsando la tecla de retroceder mientras se está agachado. Se puede aumentar la velocidad de desplazamiento manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Idle y Crouch.<br>**Entrantes:** Crouch. |
| Crouch Strafe left<br>**Movimiento de desplazamiento lateral izquierdo agachado.** | Pulsando la tecla de desplazamiento lateral izquierda mientras se está agachado. Se puede aumentar la velocidad de desplazamiento manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Crouch.<br>**Entrantes:** Crouch. |
| Crouch Strafe right<br>**Movimiento de desplazamiento lateral derecho agachado.** | Pulsando la tecla de desplazamiento lateral derecha mientras se está agachado. Se puede aumentar la velocidad de desplazamiento manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Crouch.<br>**Entrantes:** Crouch. |
| Crouch Rotate left<br>**Movimiento de rotación hacia la izquierda agachado.** | Pulsando la tecla de rotación izquierda mientras se está agachado. Se puede aumentar la velocidad de rotación manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Idle y Crouch.<br>**Entrantes:** Crouch. |
| Crouch Rotate right<br>**Movimiento de rotación hacia la derecha agachado.** | Pulsando la tecla de rotación derecha mientras se está agachado. Se puede aumentar la velocidad de rotación manteniendo la tecla de aumentar velocidad presionada. | **Salientes**: Idle y Crouch.<br>**Entrantes:** Crouch. |

Tabla 8: Enumeración de animaciones.

Las transacciones nombradas en la tabla anterior, que se encuentran en el módulo del animador de *Unity*, se pueden ver en la siguiente ilustración:

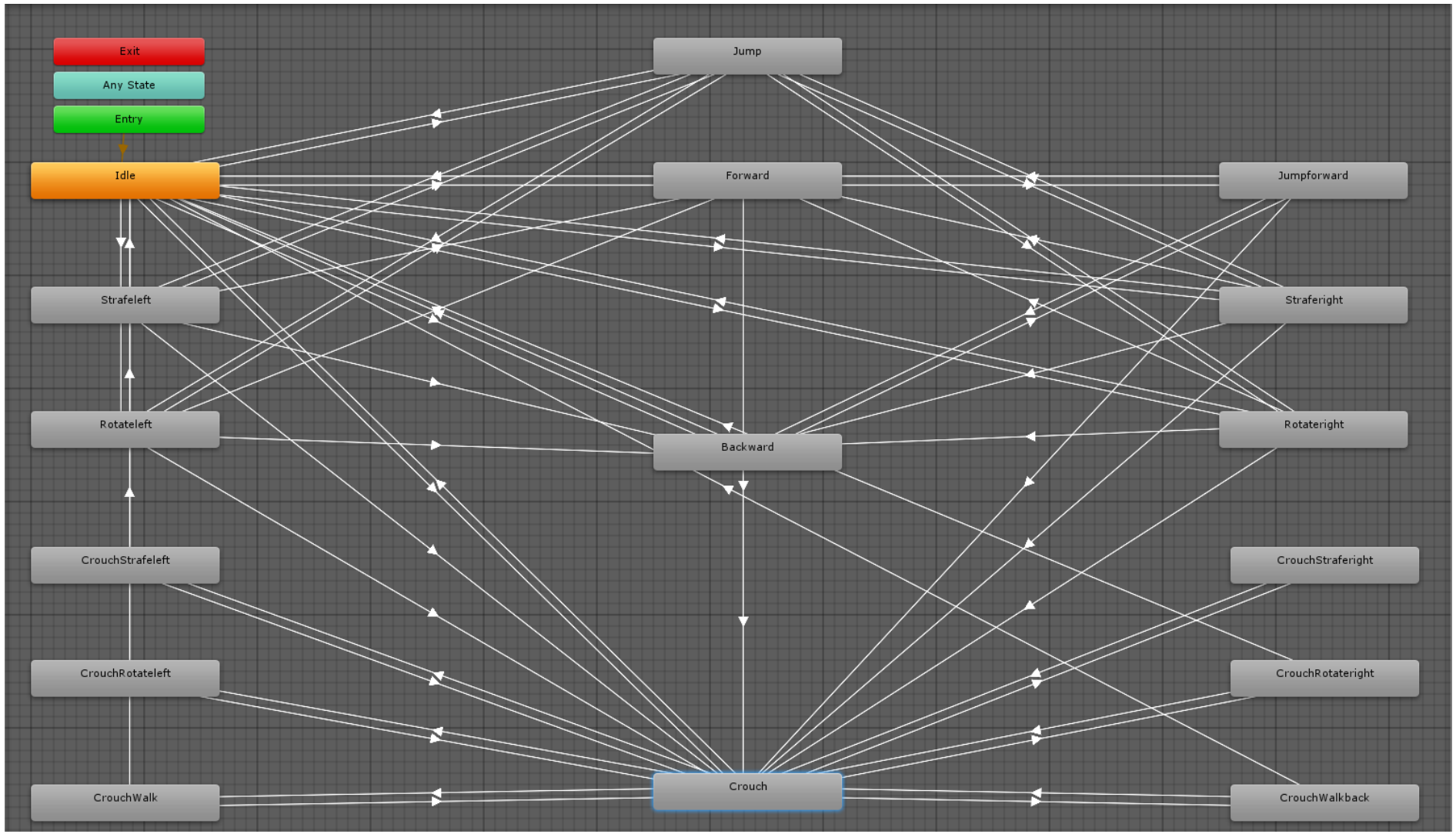



Figura 61: Esquema de transacción de animaciones de Unity.

# Apéndice B

# Descripción de los procedimientos de los elementos de interacción especial

En este apartado se describirán las propiedades y procedimientos de los objetos interactuables del juego, nombrados anteriormente, y otros elementos lógicos.

## Switches (Interruptores)

Existen seis tipos de interruptores enumerados en la clase estática Switches: SwitchA, SwitchB, SwitchBButton, SwitchWatch, PlatformSwitchA y SwitchTeleport.

Esta clase se encarga principalmente de reconocer si un objeto es un interruptor, realizar las subrutinas pertinentes cuando se han dejado de recibir por el *Raycasting de las cámaras* (visto en el capítulo *Controlador del Personaje*) y accionamiento.

- **Reconocer** (`recognize`): Primeramente, comprueba el *Tag*[16] que debe ser igual a `"Switch"`, a continuación, distingue que tipo de interruptor es por su nombre de objeto y realiza los procedimientos pertinentes: obtener la instancia del *GameObject*, la instancia de la clase y el primer proceso de activación, por lo general, aunque en cada *switch* varía. Si el *Raycasting de las cámaras* envía un objeto que no tiene la etiqueta descrita, se llama al procedimiento de *desactivar la observación* (`nothinglooking`).

- **Desactivar la observación** (`nothinglooking`): llamará al procedimiento de *no visto* (`notwatched`) de la última instancia de objeto que se ha activado con el *Raycasting de las cámaras*, para aquellos interruptores que sea oportuno.

- **Acción** (`action`): realizará la llamada al procedimiento de acción del interruptor enviado.

### SwitchA

El interruptor se encuentra en estado ***desactivado*** por defecto. Cuando el jugador lo mira directamente, es decir, el *Raycasting de las cámaras* lo reconoce (`recognize`) como *SwitchA* cambia a su estado ***observado***. Estando en este último estado y no en otro, haciendo clic sobre su superficie, de la manera que es recibido desde el *Raycasting del Crosshair* (`action`), se **activará** el interruptor quedando en ese estado. Si se vuelve a hacer clic sobre él, siendo *observado* o no, se **desactivará**.

[16] Los ***GameObjects*** en *Unity* contienen un campo ***Tag*** que se suele utilizar para diferenciar tipos de objetos dentro del código de la aplicación.

## SwitchBButton

SwitchBButton es la clase del Activador por acción ya descrito.

## SwitchB

El interruptor se encuentra en estado ***desactivado*** por defecto. Cuando el jugador mira directamente al *Activador de cámara*, es decir, el *Raycasting de las cámaras* lo reconoce (`recognize`) como *switchB* cambia a su estado ***observado***. Para ello es necesario que el personaje se sitúe lo más cerca posible del interruptor y se incline hacia adelante hasta estar a una distancia del *Activador por cámara* de 0.15 metros o menos. Moviendo la cabeza, el jugador podrá hacer incidir el *collider* del *Haz de Luz* con el del *Receptor*, en ese momento se activará su *trigger*[17] (`LightbeamReceiverTrigger`) y el interruptor pasará al estado ***apuntado***. Partiendo de este estado y no de otro, para concluir la **activación**, se debe hacer clic sobre la superficie del *Activador por acción* (`SwitchBButton`), de la manera que es recibido desde el *Raycasting del Crosshair* (`action`). Si se vuelve a hacer clic sobre él, quedará en ***apuntado***.

## SwitchWatch

Este interruptor se utiliza como complementario de: `PlatformSwitchA`. Tiene dos estados: ***activado*** y ***desactivado***.

Su procedimiento consiste en pasar al estado ***activado*** (`Switches.actionWithoutCollider()`) cuando se pulsa el botón derecho del ratón y a ***desactivado*** en cuanto se suelta (`Switches.actionWithoutCollider()`).

## PlatformSwitchA

La función que realiza este interruptor es ordenar que se desplace a un *Prefab*[18] concretamente una plataforma. Consta de tres estados: ***desactivado***, ***observado*** y ***activado***.

El jugador debe seguir el movimiento de la plataforma con la cabeza y hacer clic con el botón secundario del ratón sin importar donde este colocado el *Crosshair*.

Su funcionamiento consiste en pasar al estado ***observado*** si el *Raycasting de las cámaras* lo reconoce (`recognize`) como Platform*SwitchA*. En esa función (`watched`) se activa el brazo izquierdo (`PlayerObjects.playerAnimator.activeArmLeft()`) para indicar al jugador que se puede realizar una acción y se verifica si el *interruptor del reloj* (`SwitchWatch`) está activo (`PlayerObjects.Watch.active`), en ese caso el interruptor pasa a estado ***activado*** y con ello activará el movimiento de la plataforma (cuyo procedimiento se describirá con posterioridad). La plataforma se moverá mientras esté activo, es decir, mientras el jugador pulse el botón derecho del ratón y la plataforma no haya llegado a su tope previamente definido. Si el

[17] **Trigger** o **disparador,** en *Unity,* tipo de script que se encarga de accionar unos procedimientos cuando se realizan una serie de acciones predefinidas sobre los límites de un *collider.*

[18] Los ***Prefabs*** en *Unity* son contenedores de *GameObjects* que pueden ser instanciados y al salvar una modificación en uno de ellos se representaría en todos. Pueden ser utilizados de esa manera o solo como conjunto de *GameObjects.*

jugador vuelve a activar la plataforma, ésta se moverá en sentido contrario hasta llegar a su otro tope. Si *interruptor del reloj* no está activo, el interruptor se **desactivará**, parando la acción de la plataforma.

### SwitchTeleport

Tiene tres estados: ***no visible***, ***visible*** y ***activado***.

Para hacer ***visible*** este interruptor es necesario tener el objeto manzana (`_haveApple`) en el inventario (`Inventory`), además de recibir del *Raycasting del Crosshair* (`action`) el *GameObject* con nombre `"branch"`. Si se hace clic sobre la superficie del interruptor, de manera que es recibido desde el *Raycasting del Crosshair* (`action`), **accionará** la animación del *Teleportador* y activará su *trigger* (`SceneChange`).

## Plataformas

Existe una clase general llamada `PlatformA` que contiene los algoritmos para el movimiento de las plataformas.

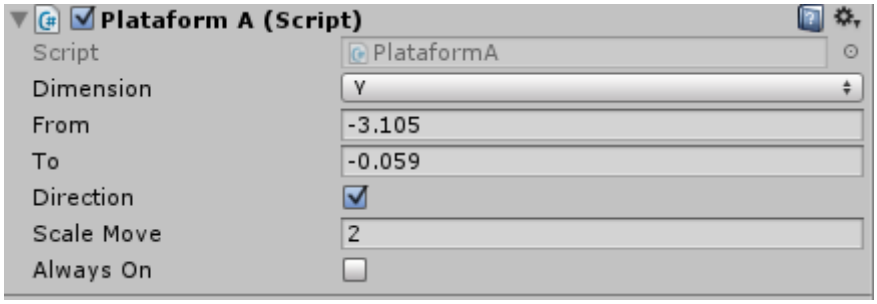



Figura 62: Parámetros de PlatformA.

Donde:

| Propiedad | Función |
|---|---|
| **Dimension** | Dimensión en la que se mueve la plataforma: x, y, z. |
| **From** | El tope mínimo del movimiento de la plataforma. |
| **To** | El tope máximo del movimiento de la plataforma. |
| **Direction** | La dirección que tomará de partida: True = [From, To] \| False = [To, From]. |
| **Scale Move** | La velocidad con la que se mueve la plataforma. |
| **Always on** | Indica si siempre está moviéndose o si se accionará mediante un interruptor. |

Tabla 9: Descripción de propiedades de PlatformA.

Los *Prefabs* donde ha sido asignada la clase `PlatformA` son: PlatformSwitchA, PlatformSwitchB, PlatformSwitchC, ElevatorA, ElevatorB, ObstacleA y PlatformA. Serán descritos a continuación.

### PlatformSwitchA, PlatformSwitchB y PlatformSwitchC

Sus funcionamientos se basan en las clases `PlatformSwitchA` y `SwitchWatch` por tanto está determinado dentro de su eje de acción (`Dimension`) y sus límites (`From` y `To`). Si el jugador mira directamente las zonas de color verde del objeto (Activadores, que pondrán activos sus *triggers* (`PlatformTriggerA`)) y a su vez mantiene el botón derecho del ratón pulsado, activará la plataforma. Proceso descrito en detalle en **PlatformSwitchA**.

### ElevatorA y ElevatorB

Su funcionamiento se basa en las clases `PlatformSwitchA` y `SwitchWatch` por tanto viene determinado dentro de su eje de acción (`Dimension`) y sus límites (`From` y `To`). Además, contienen el *Prefab* **PlatformSwitchA** y sus características. Si el jugador mira directamente las zonas de color verde del objeto (Activadores, que pondrán activos sus *triggers* (`PlatformTriggerA`)) y a su vez mantiene el botón derecho del ratón, activará la plataforma. Proceso descrito en detalle en **PlatformSwitchA**.

### ObstacleA y PlatformA

Su funcionalidad viene determinada por `PlatformA` y `AlwaysOn = true` por lo que está en continuo movimiento dentro de su eje de acción (`Dimension`) y sus límites (`From` y `To`). Hace uso de `PlatformTriggerA` para realizar el movimiento del personaje junto con la plataforma.

## Otros elementos

### GateA

Consiste en un *Prefab* que depende de `SwitchA`. Tiene dos estados: ***cerrada*** y ***abierta***.

Al activar `SwitchA` la puerta se abre ejecutando el procedimiento `changeEnabledOfChildrenObject(true)` que consiste en hacer invisibles los objetos hijos del *Prefab,* exceptuando el interruptor.

### Inventory

La funcionalidad del inventario es muy sencilla, pero existe como clase con la intención de aumentar sus posibilidades en el futuro. Contiene una propiedad que indica si se ha hecho

clic sobre la superficie del *GameObject* `Apple`, de la manera que es recibido desde el *Raycasting del Crosshair* (`action`). En ese instante hará invisible a `Apple` y asignará `true` a `_haveApple`.

Otra función es hacer visible `SwitchTeleport`, si el objeto recibido desde el *Raycasting del Crosshair* (`action`) es `"branch"` y además `_haveApple` es `true`.

### PlatformReturnStart

Su funcionamiento viene determinado por su *trigger* `ReturnStartTrigger`. Provoca que el personaje sea enviado de vuelta a las coordenadas iniciales de la escena y suele situarse en zonas donde el jugador no podrá volver a la zona jugable por sí solo.

## Triggers

Para este prototipo se han utilizado los eventos `OnTriggerEnter`, `OnTriggerStay` y `OnTriggerExit` de *trigger* de *Unity*:

- **`OnTriggerEnter:`** se produce cuando otro *collider* entra en colisión con el del contenedor del *trigger*.
- **`OnTriggerStay:`** se produce mientras otro *collider* continua en colisión con el del contenedor del *trigger*.
- **`OnTriggerExit:`** se produce cuando un *collider* deja de hacer colisión con el del contenedor del *trigger*.

### ReturnStartTrigger

Cuando el *collider* del personaje entra en contacto con el contenedor del *trigger* (`OnTriggerEnter`) modifica su posición para devolverlo al punto determinado por `ReturnPosition`.

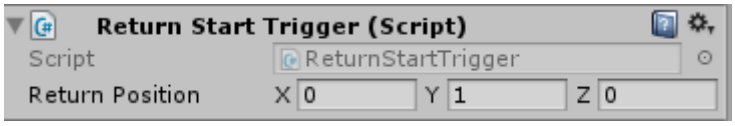



Figura 63: Parámetros de ReturnStartTrigger.

### LightbeamReceiverTrigger

En **`OnTriggerEnter`** si el G*ameObject* que ha colisionado tiene como etiqueta `"Switch"` ejecuta `pointed` que cambiará el estado a ***Apuntado*** de la instancia recibida de la clase `SwitchB`.

En **`OnTriggerExit`** si el G*ameObject* que ha colisionado tiene como etiqueta `"Switch"` ejecuta `notpointed` que cambiará el estado a ***Observado*** de la instancia recibida de la clase `SwitchB`.

## PlatformTriggerA

Mientras el *collider* del objeto recibido se encuentre dentro de los límites del contenedor, modifica la posición de dicho G*ameObject* igualándola a la del contenedor de la instancia. Esto provoca que el personaje se mueva junto con la plataforma, por ejemplo.

## SceneChange

Cuando el *collider* del personaje entra en contacto con el contenedor del *trigger* (`OnTriggerEnter`) cambia a la escena especificada en `changescena`.

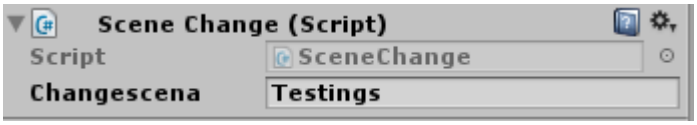



Figura 64: Parámetros de SceneChange.

# Bibliografía


[1] 1. **Canet, Mar.** Innovación en interfaces para videojuegos desde el Game Art. *Artículo sobre interfaces hombre-máquina.* [En línea] 11 de noviembre de 2015. http://www.injuve.es/sites/default/files/2012/46/publicaciones/Revista98_10.pdf.

[2] 2. **Microsoft.** Sitio web del desarrollador Microsoft Kinect. [En línea] https://dev.windows.com/en-us/kinect.

[3] 3. **Leap Motion.** Sitio web de Leap Motion. [En línea] https://www.leapmotion.com/.

[4] 4. **Wikipedia.** Realidad Virtual en Wikipedia. [En línea] https://es.wikipedia.org/wiki/Realidad_virtual.

[5] 5. —. Head-Mounted Display en la Wikipedia. [En línea] https://es.wikipedia.org/wiki/Casco_de_realidad_virtual.

[6] 6. —. Primer videojuego en la Wikipedia. [En línea] https://es.wikipedia.org/wiki/Primer_videojuego.

[7] 7. **Otakufreaks.** Primer videojuego en Otakufreaks. [En línea] http://www.otakufreaks.com/historia-de-los-videojuegos-el-origen-y-los-inicios/.

[8] 8. **Wikipedia.** Magnavox Odyssey en la Wikipedia. [En línea] https://es.wikipedia.org/wiki/Magnavox_Odyssey.

[9] 9. —. Virtual Boy en la Wikipedia. [En línea] https://es.wikipedia.org/wiki/Virtual_Boy.

[10] 10. **Retromaquinitas.** Virtual Boy en Retromaquinitas. [En línea] http://www.retromaquinitas.com/index.php/portatiles/quinta-generacion-portatiles/virtual-boy.

[11] 11. **Wikipedia.** Juegos en primera persona en Wikipedia . [En línea] https://es.wikipedia.org/wiki/Primera_persona_%28videojuegos%29.

[12] 12. —. John Carmack en la Wikipedia. [En línea] https://es.wikipedia.org/wiki/John_Carmack.

[13] 13. **Oculus VR.** Sitio web de la compañía Oculus. [En línea] www.oculus.com.

[14] 14. **Wikipedia.** Oculus VR en la Wikipedia. [En línea] https://es.wikipedia.org/wiki/Oculus_VR.

[15] 15. **Valve y HTC.** Sitio web de HTC Vive. [En línea] https://www.htcvive.com/us/.

[16] 16. **Itpro.** HTC Vive en Itpro. [En línea] http://www.itpro.co.uk/desktop-hardware/24985/htc-vive-release-date-price-features-and-specs-htc-to-put-on-immersive.

[17] 17. **Xataka.** HTC Vive en Xataka. [En línea] http://www.xataka.com/realidad-virtual-aumentada/htc-vive-pre-primeras-impresiones-asi-es-la-verdadera-realidad-virtual.

[18] 18. **Microsoft.** Sitio web de Microsoft Hololens. [En línea] https://www.microsoft.com/microsoft-hololens/en-us.

[19] 19. **Itpro.** Microsoft Hololens en Itpro. [En línea] http://www.itpro.co.uk/mobile/24780/microsoft-hololens-release-date-rumours-specs-pricing-3.

[20] 20. **Wareable.** Artículos sobre los mejores juegos de realidad virtual. [En línea] http://www.wareable.com/oculus-rift/the-best-oculus-rift-games.

[21] 21. **Crytek.** Sitio Web de The Climb. [En línea] 2016. The Climb.

[22] 22. **Vandal.** Artículo sobre el videojuego The Climb en la revista Vandal. [En línea] http://www.vandal.net/avances/pc/the-climb/35107/1.

[23] 23. —. Sitio Web de ADR1FT en Vandal. [En línea] 2016. http://www.vandal.net/juegos/ps4/adr1ft/27337.

[24] 24. —. Análisis de The Witness en Vandal. [En línea] 2016. The Witness.

[25] 25. **Steam.** Sitio web de Detached en Steam. [En línea] http://store.steampowered.com/app/436230/.

[26] 26. **Mindfield Games Ltd.** Sitio Web de P.O.L.L.E.N. [En línea] 2016. http://www.pollengame.com.

[27] 27. **Wikipedia.** Oculus Rift en la Wikipedia. [En línea] https://es.wikipedia.org/wiki/Oculus_Rift.

[28] 28. **Oculus VR.** Sitio Web de Oculus Rift Development Kit 2. [En línea] https://www.oculus.com/en-us/dk2/.

[29] 29. **ifixit.** Sitio web de ifixit expertos en desmontar productos tecnológicos. *Artículo sobre las características técnicas y reparabilidad de Oculus Rift DK2.* [En línea] 11 de diciembre de 2015. https://www.ifixit.com/Teardown/Oculus+Rift+Development+Kit+2+Teardown/27613.

[30] 30. **Unity Technologies.** Sitio web de Unity 3D. [En línea] https://unity3d.com/es.

[31] 31. —. Manual de Unity 3D. [En línea] http://docs.unity3d.com/es/current/Manual/.

[32] 32. **Wikipedia.** Ejes del avión en la Wikipedia. [En línea] https://es.wikipedia.org/wiki/Ejes_del_avi%C3%B3n.