

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR



Master en Investigación e Innovación

TRABAJO FIN DE MASTER

**SISTEMA DE EYE TRACKER EN
REALIDAD VIRTUAL PARA
PERSONAS QUE TIENEN
DISCAPACIDAD MOTORA**

Autor: Cristian Fernández Del Pozo
Tutor: Francisco de Borja Rodríguez Ortiz

ENERO 2019

SISTEMA DE EYE TRACKER EN REALIDAD VIRTUAL PARA PERSONAS QUE TIENEN DISCAPACIDAD MOTORA

Autor: Cristian Fernández Del Pozo
Tutor: Francisco de Borja Rodríguez Ortiz

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
ENERO 2019

Resumen

Resumen

RESUMEN

Palabras Clave

Realidad Virtual, VR, Tecnologías de la Información y la comunicación, Oculus Rift, Google Cardboard, Navegación, Personas en situación de discapacidad

Abstract

Abstract

Key words

Virtual Reality, VR, Information and Communication Technologies, Oculus Rift, Google Cardboard, Navigation, People with disabilities.

Agradecimientos

Agradecimientos.....

Índice general

Agradecimientos	v
Índice de Figuras	ix
Índice de Tablas	xI
1. Introducción	1
1.1. Motivación del proyecto	1
1.2. Objetivos y enfoque	2
1.3. Desarrollo y plan de trabajo	3
2. Estado del arte	5
2.1. La realidad virtual a día de hoy	5
2.2. Oculus Rift y otros sistemas de gafas de realidad virtual	6
2.3. Unity 3D	7
2.4. Discapacidades motoras habituales	8
2.5. La realidad virtual y las discapacidades	9
3. Sistema, diseño y desarrollo	11
3.1. Oculus Rift	11
3.1.1. Aspectos Físicos	11
3.1.2. Hardware	13
3.1.3. Software	15
3.1.4. Configuración del DK2	16
3.2. Unity 3D	16
3.2.1. C#, JavaScript y Mono Runtime	16
3.2.2. Descripción de Unity 3D	17
3.3. Requisitos	20
3.3.1. Requisitos funcionales	20
3.3.2. Requisitos no funcionales	21
3.4. Diseño del sistema	22

3.4.1. Estructura del código	22
3.5. Desarrollo del sistema	23
3.6. Diagrama de clases	23
3.7. Casos de uso	26
3.8. Problemas encontrados y soluciones	29
4. Experimentos Realizados y Resultados	31
4.1. Estudio de efectos de las gafas de realidad virtual	31
4.1.1. Metodología	31
4.1.2. Resultados	32
4.1.3. Conclusiones	32
4.2. Estudio de comodidad y usabilidad del sistema	33
4.2.1. Metodología	33
4.2.2. Resultados	33
4.2.3. Conclusiones	34
5. Conclusiones, discusión y trabajo futuro	35
Glosario de acrónimos	37
Bibliografía	38
A. Casos de Uso	41
A.1. Caso 1	41
A.2. Caso 2	43
A.3. Caso 3	46
B. Código Fuente	53
B.1. Menu	53
B.2. Panel de información	57
B.3. Navegador	58
B.4. Teclado Virtual	65
B.5. Sistema de interacción Gaze	78
B.6. Sistema de emulación de clicks y teclado	80
B.7. Utilidades	83

Índice de Figuras

1.1.	Modelo de ciclo de vida incremental iterativo. Fuente: https://upload.wikimedia.org/wikipedia/commons/c/c4/Modelo_Iterativo_Incremental.jpg	3
2.1.	Videoconsola <i>Virtual Boy</i> de Nintendo Company. (1995) Fuente: https://upload.wikimedia.org/wikipedia/commons/c/ce/Virtual-Boy-wController.jpg	6
2.2.	<i>Archos VR Glasses, Samsung Gear VR y Google Cardboard</i> Fuente: https://c2.staticflickr.com/6/5607/14961040484_ff7643245d_b.jpg	7
2.3.	Interfaz de usuario de <i>Unity 3D</i> Fuente: https://c1.staticflickr.com/5/4130/5035620615_5891474154_b.jpg	8
2.4.	Vía piramidal motora. Fuente: https://upload.wikimedia.org/wikipedia/commons/6/68/Gray764.png	9
3.1.	Diagrama que representa como las imágenes percibidas por nuestros ojos fusionan una imagen. Fuente: https://es.wikipedia.org/wiki/Estereoscopia#/media/File:Estereoglifo.jpg	12
3.2.	Efecto <i>Motion Blur</i> en un autobús y pantalla de la versión DK2. Fuente: https://upload.wikimedia.org/wikipedia/commons/2/26/London_bus_and_telephone_box_on_Haymarket.jpg	13
3.3.	Ejemplo de rotaciones posibles sobre un objeto, en este caso un avión. Fuente: https://upload.wikimedia.org/wikipedia/commons/5/54/Flight_dynamics_with_text.png	14
3.4.	Sensor de infrarrojos de la versión DK2. Fuente: https://upload.wikimedia.org/wikipedia/commons/9/9c/Oculus_Rift_Development_kit_2_positional_tracker.jpg	14
3.5.	Pantalla de animación de <i>Unity 3D</i> . Fuente: https://i.vimeocdn.com/video/459915521_1280x960.jpg	19
3.6.	Diagrama de clases de la aplicación. Muestra las Vistas del patrón MVC	24
3.7.	Diagrama de clases de la aplicación. Muestra los Controladores del patrón MVC .	25
3.8.	Diagrama de caso de uso 1	26
3.9.	Diagrama de caso de uso 2	27
3.10.	Diagrama de caso de uso 3	28
A.1.	Pantalla de inicio del sistema.	41
A.2.	Usuario seleccionado la opción para abrir el menú.	42
A.3.	Menú desplegado.	42

A.4. Usuario seleccionado la opción para apagar el sistema.	43
A.5. Pantalla de inicio del sistema.	43
A.6. Usuario seleccionado la opción para abrir el menú.	44
A.7. Menú desplegado.	44
A.8. Usuario seleccionando la opción para abrir el navegador.	45
A.9. Navegador del sistema.	45
A.10. Usuario seleccionado la opción para cerrar el navegador.	46
A.11. Usuario seleccionado la opción para apagar el sistema	46
A.12. Pantalla de inicio del sistema.	47
A.13. Usuario seleccionado la opción para abrir el menú.	47
A.14. Menú desplegado.	48
A.15. Usuario seleccionando la opción para abrir el navegador.	48
A.16. Navegador del sistema.	49
A.17. Limpiando campo para escribir la URL.	49
A.18. Escribiendo con el teclado virtual.	50
A.19. Cargando la URL.	50
A.20. Visualizando un vídeo.	51
A.21. Usuario seleccionado la opción para cerrar el navegador.	51
A.22. Usuario seleccionado la opción para apagar el sistema	52

Todas las imágenes presentes han sido obtenidas a través de *Google Images* bajo licencia de uso no comercial y modificación o bien son de mi propia autoría.

Índice de Tablas

2.1. Cuantificación del déficit de la fuerza motora.	8
4.1. Resultados de test de efectos de las <i>Oculus Rift DK2</i> . La primera columna corresponde al tipo de lente. La segunda columna a la pregunta de si el usuario suele jugar a videojuegos. La tercera columna corresponde a la pregunta de si tras la prueba, el usuario ha sufrido vértigo. La cuarta columna a si a sufrido mareos. La quinta columna representa la respuesta a la pregunta de si el usuario ha sufrido distorsión de la realidad.	32
4.2. Resultados de test de comodidad y usabilidad. La primera columna corresponde al tipo de lente usada. La segunda columna corresponde al tiempo usando teclado, ratón y el navegador <i>Mozilla Firefox</i> , la tercera columna corresponde a los tiempos usando el sistema de este trabajo sin entrenamiento, la cuarta columna corresponde a los tiempos del sistema desarrollado en este trabajo con entrenamiento de tres a cinco minutos. Resultados de tiempos en segundos.	34
4.3. Medias y desviación estándar de los datos obtenidos en la tabla 4.2. Resultados de tiempos en segundos.	34

1

Introducción

1.1. Motivación del proyecto

1.2. Objetivos y enfoque

1.3. Desarrollo y plan de trabajo

2

Estado del arte

2.1. La realidad virtual a día de hoy

Realidad virtual. *Representación de escenas o imágenes de objetos producida por un sistema informático, que da la sensación de su existencia real [3].*

Según la anterior definición, la realidad virtual tiene como objetivo "engañar" a nuestra mente de todas las maneras posibles. No son solo unas gafas que confunden a nuestro cerebro al hacer que cada ojo perciba una imagen con una transformación afín, sino a cualquier sistema que sea capaz de jugar con nuestros sentidos, como guantes que simulen sensaciones táctiles, cascos de sonidos envolventes o sistemas más complejos. En resumen, la inmersión en la realidad virtual se consigue con todo aquello que proporcione información a nuestro cerebro generando una sensación que consiga emular el mundo real.

A día de hoy la realidad virtual está en auge. Los primeros sistemas no son tan novedosos como nosotros creemos. En la década de los ochenta - noventa (figura 2.1 ya existían sistemas de realidad virtual destinados a ocio y a fines militares. Aunque, como es obvio, eran tecnologías tan limitadas que la sensación de inmersión era demasiado baja y el coste muy elevado, por lo que no tuvieron mucho éxito.

Vivimos en una época en la que la tecnología para crear espacios tridimensionales es tan potente que a veces cuesta diferenciar el mundo real del virtual. Por ese motivo la sensación de inmersión es muchísimo mayor que hace veinte años. Esto, sumado al aumento de producción de tecnología y a su abaratamiento, ha desencadenado la aparición de dispositivos cada vez más innovadores. Vivimos en una época de ciencia ficción.

Algunos de los dispositivos que cumplen las características antes mencionadas son las gafas *Oculus Rift*, o su competencia las gafas *HTC Vive*[4]. También existe su versión mas económica para dispositivos móviles aprovechando sus sensores de posición y rotación. Dichas gafas empezaron con la aparición de las *Google Cardboard*[5] en el año 2014. Mencionar también que el éxito de estos lanzamientos no es debido solo a su fantástica tecnología, sino también a que han proporcionado de manera libre herramientas a los programadores para el desarrollo de sus propias aplicaciones y programas.

En concreto, tanto las *Oculus Rift* como las *Google Cardboard* tienen a disposición de cualquiera y de manera gratuita un SDK para *Unity 3D* que permite crear un entorno de VR.



Figura 2.1: Videoconsola *Virtual Boy* de Nintendo Company. (1995)

Fuente:<https://upload.wikimedia.org/wikipedia/commons/c/ce/Virtual-Boy-wController.jpg>

Pero, como mencioné anteriormente, no todo gira en torno a gafas de realidad virtual basada en imágenes: existe, como ejemplo, la plataforma llamada *Virtuix Omni* que nos permite desplazarnos por un mundo virtual andando o corriendo. Consiste en una base fija en el suelo que detecta los pasos del usuario gracias a unas zapatillas especiales [6].

Otro ejemplo es el parque temático llamado *The Void* que recrea un escenario físico para adecuarlo al que se percibe en el mundo virtual a través de unas gafas, introduciendo elementos como ventiladores para emular ráfagas de viento, creando así una mayor sensación de realismo al mezclar los estímulos táctiles que percibimos a través de nuestros sentidos [7].

En conclusión, la realidad virtual es una tecnología que no es moderna pero que está viviendo actualmente su época dorada gracias al sector del ocio. Aún así, surgen aplicaciones para un uso con fin social, humanitario o médico gracias a las facilidades que hay para crear aplicaciones propias.

Dispositivos como las gafas *Oculus Rift* o *HTC Vive* permiten disfrutar de sensaciones nuevas llevando el ocio a un nuevo nivel y por ello su fama crece exponencialmente gracias a su coste asequible.

2.2. Oculus Rift y otros sistemas de gafas de realidad virtual

Las gafas de realidad virtual son los dispositivos que más impacto han tenido gracias a que proporcionan la mayor sensación de inmersión: nos permiten ver un mundo distinto al nuestro. *Oculus Rift* tiene el honor de ser el primer sistema HMD (*Head-Mounted Display*) al despuntar en el sector gracias al uso de tecnologías como los giroscopios y los acelerómetros. Está soportado por un ordenador dejando los costes computacionales de la generación de gráficos a la tarjeta gráfica del ordenador limitándose exclusivamente a mostrar imágenes y a detectar movimientos.

Aunque haya sido uno de los primeros no es el único. Su principal competencia son las gafas desarrolladas por *Valve Corporation* y *HTC Corporation* llamadas *HTC Vive*. Se basan en los mismos principios y su diferencia es cuestión de calidad de hardware y no de funcionalidad.

Otro sector son los cascos de realidad virtual destinados a los *SmartPhones*. Las primeras gafas en surgir en este sector fueron las *Google Cardboard*, actualmente en su segunda versión.

Como característica principal están construidas con cartón, siendo las primeras versiones no comerciales esquemas publicados en Internet para su construcción casera.

Actualmente han surgido modelos más trabajados que sustituyen el cartón por chasis plásticos. Obviamente el coste también se ve incrementado, siendo el precio de las *Google Cardboard* de 5€ a 10€ y de otros dispositivos, como las *Archos Glasses*, rondando los 25€ (figura 2.2).



Figura 2.2: *Archos VR Glasses*, *Samsung Gear VR* y *Google Cardboard*

Fuente:https://c2.staticflickr.com/6/5607/14961040484_ff7643245d_b.jpg

Además de lo mencionado, *Oculus* ha desarrollado para *Samsung* las *Samsung Gear VR*, un casco de realidad virtual destinado para teléfonos *Samsung* que incorpora un touchpad con el que interaccionar. Su precio se aproxima a los 100€.

2.3. Unity 3D

Unity 3D [8] es una herramienta de diseño e implementación de videojuegos que ha revolucionado el mercado. Una de sus principales características es la capacidad de generar versiones para distintas plataformas como pc, videoconsolas o contenido online. Esto es gracias a que utiliza *C# / JavaScript* bajo el intérprete *Mono Runtime*. Proporciona funcionalidades para diferenciar plataformas, de modo que no hace falta generar distintos proyectos pudiendo estar la codificación para distintas plataformas en un solo proyecto.

Otra de las características revolucionarias es que incluye herramientas para diseñar en 3D, generar animaciones, manejar partículas y un motor físico. Antes de *Unity 3D*, estas características se tenían que desarrollar por separado con programas como *Blender 3D* [9], el software de modelado 3D y animación de código abierto.

Además proporciona una tienda online donde se puede comprar u obtener gratis determinados componentes para incluirlos en nuestros juegos y aplicaciones.

Una de las desventajas con las que cuenta este software es que su curva logarítmica de aprendizaje describe un avance lento; expresado en otras palabras, el proceso de su aprendizaje se prolonga en el tiempo debido a que la cantidad de conceptos y tecnologías a adquirir puede llegar a ser abrumadora. Aun así, los desarrolladores cuentan con una colección excelente de tutoriales [10] en los que se esfuerzan en ayudar de manera gratuita a la comunidad. Existe también una gran cantidad de foros y sitios web donde la comunidad resuelve dudas. En España destaca la web <http://www.unityspain.com/>.

En conclusión, *Unity 3D* ha llegado a ser una de las mejores tecnologías para el desarrollo de videojuegos adoptando una filosofía que premia a la comunidad y da facilidades para aprender. Aunque no es un software cerrado, como toda empresa busca un beneficio económico. Dicho beneficio lo consigue gracias al soporte técnico y al desbloqueo de herramientas, útiles en su mayoría cuando uno se embarca en un proyecto verdaderamente grande en *Unity 3D*, con lo que

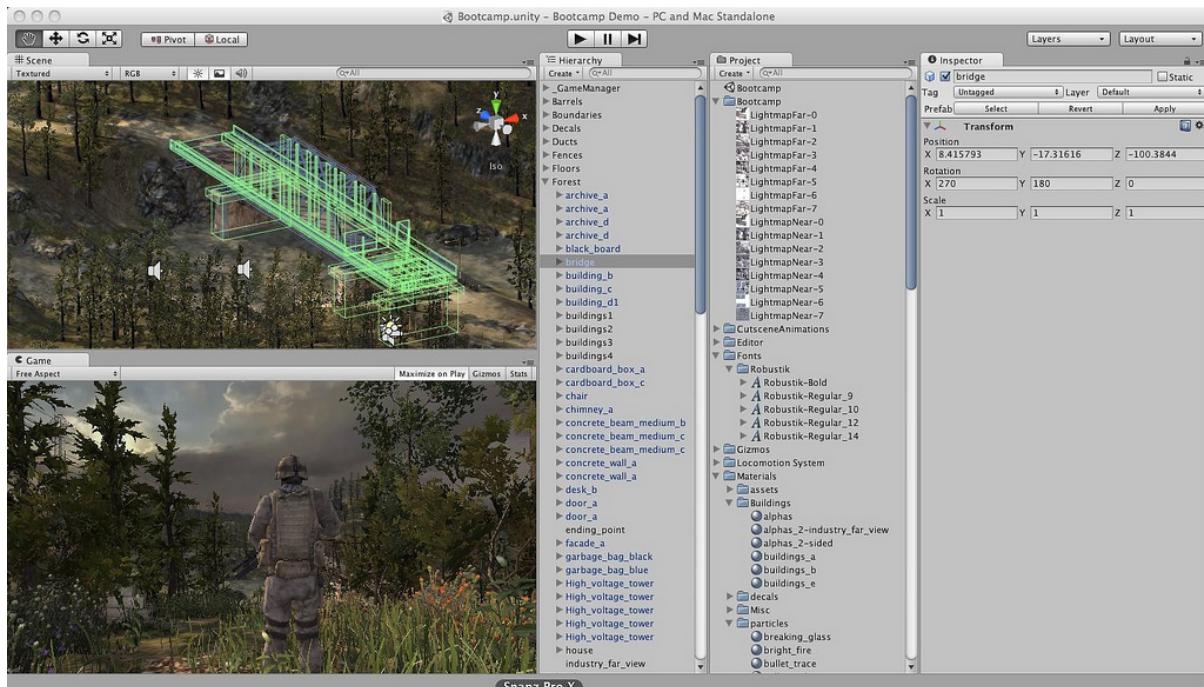


Figura 2.3: Interfaz de usuario de *Unity 3D*

Fuente:https://c1.staticflickr.com/5/4130/5035620615_5891474154_b.jpg

no afecta al desarrollador que solo quiere aprender o hacer algo simple. Por este motivo se ha elegido como plataforma base para el desarrollo de este TFG.

2.4. Discapacidades motoras habituales

Las discapacidades motoras son aquellas discapacidades que agregan una desventaja en el movimiento o en la fuerza de los movimientos de una persona. Según el cuadro 2.1, el déficit de fuerza se puede catalogar en seis niveles siendo el 0 la parálisis total y el 5 el estado normal.

Nivel déficit motor	
Nivel	Descripción
0	Parálisis completa
1	Contracción muscular sin movimiento
2	Movimiento pero sin fuerza para superar la gravedad
3	Movimiento pero sin fuerza para superar una resistencia física
4	Movimiento más débil del esperado
5	Fuerza normal

Cuadro 2.1: Cuantificación del déficit de la fuerza motora.

Como se ha visto, no todas las discapacidades motoras indican una imposibilidad de movimiento, sino que únicamente pueden limitarlo. Además, muchas de estas limitaciones desembocan en una atrofia muscular derivada del déficit de uso.

El sistema piramidal es la vía neuronal localizada en el sistema nervioso central que ejerce la acción del movimiento voluntario en el cuerpo humano. Esta orden de movimiento se genera en la corteza cerebral, donde se emplaza el núcleo de la primera motoneurona. El axón (terminación) de esta primera motoneurona desciende hasta la médula espinal donde hace contacto con la segunda motoneurona. A esta conexión se le llama sinapsis y es un intercambio de información por diferencias de potencial entre axones. La segunda motoneurona es la encargada de conectarse

al músculo y generar el impulso nervioso que provoca la contracción de dicho músculo [11].

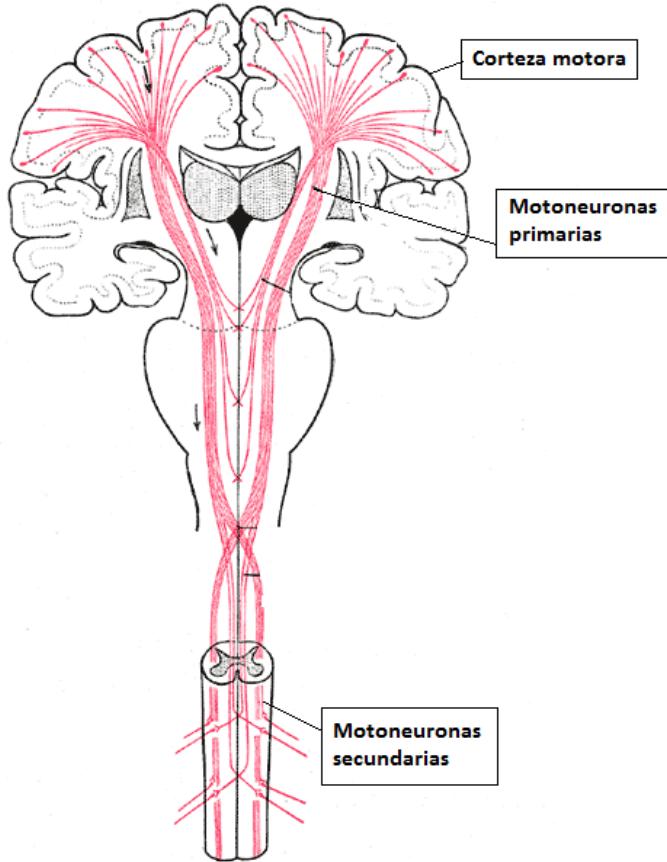


Figura 2.4: Vía piramidal motora.

Fuente:<https://upload.wikimedia.org/wikipedia/commons/6/68/Gray764.png>

Las lesiones en la primera motoneurona responden a alteraciones de naturaleza diversa como problemas vasculares, isquémicos, inflamatorios o tumorales; siendo infrecuentes las lesiones ocasionadas por traumatismos. Las manifestaciones de esta lesión se dan en forma de pérdida en la precisión del movimiento, rigidez en extremidades, pérdida de reflejos o combinación de varias. Las lesiones en la segunda motoneurona son provocadas por los mismos motivos que la primera motoneurona sumando los traumatismos y lesiones degenerativas. Todas las lesiones en este punto producen parálisis totales o parciales [12].

En conclusión, las discapacidades motoras pueden tener distinta sintomatología y no se limitan solo a una imposibilidad total de movimiento. Pueden generar pérdida de fuerza, temblores o reducción de la capacidad de movimiento de una extremidad. Todo esto se debe tener en cuenta a la hora de desarrollar un sistema que pretenda ayudar a este sector.

2.5. La realidad virtual y las discapacidades

Las Tecnologías de la Información y del Conocimiento surgen para dar soluciones a los problemas que acarrea la humanidad y por tanto es lógico pensar que proporcionen o que al menos intenten proporcionar soluciones a las discapacidades.

Las interfaces cerebro-máquina pueden ser una respuesta a las discapacidades motoras. Cuando nuestro cuerpo se convierte en nuestra cárcel, nuestra mente es nuestro único refugio. An-

lizando los patrones de las ondas producidas por nuestro cerebro se puede obtener cierta información de los movimientos de una persona. Un caso de uso de esto es un estudio realizado para mover una silla de ruedas a través de esta tecnología en un entorno virtual. Se analizaban los patrones gracias a un casco de electroencefalogramas (EEG)[13]. Tal vez este sea el principio de la utópica idea de crear un sistema para superar totalmente las discapacidades o de desarrollar una interfaz total de movimiento en entornos virtuales, como se ha visto solo en el cine.

En el año 2014 se hizo un estudio en el que se evaluaba las ondas captadas por un EEG y la sensación de estabilidad y equilibrio demostrando que la realidad virtual es capaz de influir en dichas sensaciones. La conclusión de este estudio fue que la sensación de inmersión es lo suficientemente grande como para producir que nuestros reflejos naturales se activen por sí solos [14].

Ignorando ya las discapacidades motoras, la realidad virtual se puede usar como una herramienta más dentro de las TIC con el fin de ayudar a las personas. En el año 2015 se realizó un estudio en el que se intentaba que pacientes con autismo adquirieran la habilidad para cruzar la calle y seguir caminos. El experimento consistió en generar una ciudad tridimensional con tecnología de realidad virtual para que personas con autismo en un rango de edad entre los 19 y los 45 años tuvieran que aprender y seguir un camino en el que se tienen que enfrentar a semáforos, pasos de cebra y demás elementos de la vida urbana [15].

En conclusión, la realidad virtual es un campo muy prometedor que, en conjunto con otras tecnologías, puede dar como resultado herramientas capaces de ayudar a las personas y que merece de un esfuerzo por nuestra parte para que evolucionen.

3

Sistema, diseño y desarrollo

En este capítulo se exponen las labores desarrolladas en este TFG. Para ello, primero hay que entender en detalle las tecnologías implicadas en su desarrollo: *Oculus Rift* y *Unity 3D*. Después se pasa a la parte de diseño y desarrollo del software, para ello hay una descripción de los requisitos funcionales y no funcionales subcatalogados en *Fundamentales* (cuyo objetivo es este TFG) o *No Fundamentales* cuyo desarrollo se pospondrá a un trabajo futuro. Se mostrará a continuación el diseño de clases a través de un diagrama de clases de la aplicación y los distintos escenarios de casos de uso del sistema.

3.1. Oculus Rift

En esta sección nos centraremos en las *Oculus Rift* más en profundidad. Hay que tener en cuenta que muchos de sus principios básicos de funcionamiento los comparte con otros dispositivos como las *HTC Vive* o las *Google Cardboard*.

3.1.1. Aspectos Físicos

El principal principio en el que se basan las *Oculus Rift*, y casi todos los sistemas de realidad virtual enfocados al sentido de la vista, es en la capacidad de nuestro cerebro para fusionar imágenes cuando se presentan por individual en cada ojo. El impedimento que esto conlleva es que usuarios con enfermedades como el estrabismo ocular tienen dificultades para realizar dicha fusión. En consecuencia, tienen problemas en la percepción de la profundidad y en el caso de uso de tecnologías 3D (no solo las enfocadas a la realidad virtual) pueden sufrir mareos e incapacidad para percibir la formación de la imagen 3D, ya que solo serán capaces de ver figuras borrosas, de manera similar a cuando nos quitamos las gafas en un cine 3D.

Este principio se basa en la convergencia ocular y en el ángulo de convergencia que forman los dos ojos con el objeto que se percibe. Imaginemos un objeto situado frente a nosotros: debido a que disponemos de una visión binocular (cada ojo ve el objeto por separado) la focalización de este objeto percibido requiere del posicionamiento de cada ojo en un ángulo adecuado, de modo que ambas imágenes convergirán y se percibirán como una sola y con una profundidad adecuada respecto al espacio que le rodea. El ángulo de convergencia que forman los dos ojos

con el objeto que se percibe aumenta a medida que el objeto se halla más próximo y disminuye cuando el objeto está más lejano. Esta distancia de convergencia suele estar en un valor entre 7 y 11 cm. Cuando el ángulo o distancia de convergencia no es adecuado se produce la llamada diplopia o visión doble, que explica porqué cuando acercamos mucho a la nariz un objeto, la imagen no se forma correctamente en nuestro cerebro y la vemos borrosa.

En la figura 3.1 se puede ver un pequeño esquema en el que se ha representado el ángulo de convergencia de cada ojo y cómo se produce la fusión (3D) cuando se posiciona a la distancia adecuada.

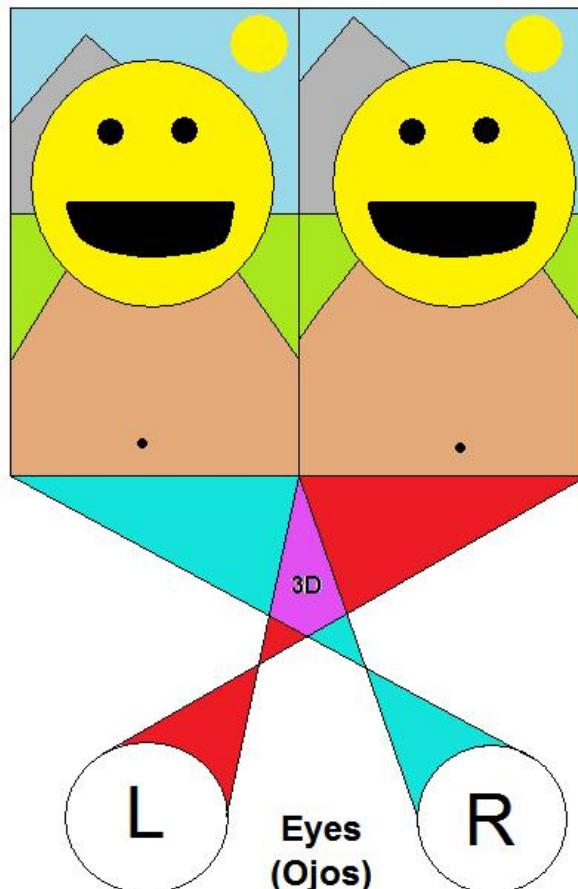


Figura 3.1: Diagrama que representa como las imágenes percibidas por nuestros ojos fusionan una imagen.

Fuente:<https://es.wikipedia.org/wiki/Estereoscopia#/media/File:Estereoglifo.jpg>

El problema de un sistema de gafas de realidad virtual es que no es capaz de proporcionar una imagen a una distancia de convergencia adecuada, debido a que esta distancia de la que hablábamos anteriormente es de aproximadamente 10 cm, por lo que haría necesario diseñar unas gafas de profundidad excesiva. Para dar solución a este problema se ha ideado un sistema en el que se generan las dos imágenes con una serie de transformaciones afines pertinentes, de modo que sea posible corregir esta distorsión provocada por la poca profundidad de las gafas. Además se utilizan unas lentes capaces de aumentar la imagen para que el ojo las perciba adecuadamente y se produzca la fusión y la percepción del mundo 3D. A este tipo de imágenes se las conoce como imágenes estereoscópicas [16]. El concepto de imagen estereoscópica no es actual sino que surgió hace tiempo, existiendo juguetes con estas imágenes que datan del siglo anterior.

3.1.2. Hardware

El hardware que se va a describir a continuación es el correspondiente a las *Oculus Rift Development Kit 2 (DK2)*, que concierne a la versión con la que se ha trabajado. Actualmente existe una nueva versión comercial, de manera que esta versión ha quedado obsoleta. Las diferencias principales con sus rivales como las *HTC Vive* son las características HW que las definen, es decir, tamaño de pantalla, calidad de ésta, etc.

Pantalla

Para representar las imágenes con las que trabaja cuentan con una pantalla con una resolución de 1980x1080 con una latencia de imagen en el rango de los [2,3] ms, lo que garantiza una velocidad de refresco de imagen casi en tiempo real. En la versión DK1, el alto tiempo de latencia provocaba el llamado efecto *Judder* que consiste en un desfase de la señal de audio y vídeo. Basada en la tecnología OLED [17], se consigue eliminar el efecto *Judder* [18] y el llamado *Motion Blur* [19] que tiene lugar en imágenes o vídeos en los que aparece un movimiento rápido.



Figura 3.2: Efecto *Motion Blur* en un autobús y pantalla de la versión DK2.

Fuente:https://upload.wikimedia.org/wikipedia/commons/2/26/London_bus_and_telephone_box_on_Haymarket.jpg

Para la conexión con el ordenador se utiliza un conector tipo HDMI, lo que permite que en el frontal de las gafas haya un conector Jack de audio. El audio es una señal que se manda junto con el video por un canal HDMI.

Giroscopio y acelerómetro

La base de todas las gafas de realidad virtual es la capacidad de detección de los movimientos rotacionales de la cabeza junto con el movimiento acorde de la cámara dentro del mundo tridimensional. Hablando exclusivamente de rotaciones, un objeto puede rotar únicamente en torno a tres vectores, ejemplificado en la figura 3.3. Para poder implementar esta funcionalidad, *Oculus Rift* cuenta con un giroscopio de tres ejes y un acelerómetro. Así calcula la dirección de rotación y aceleración del movimiento. Gracias a que los *Smartphones* actuales también tienen estos dispositivos, existen gafas como las *Google Cardboard*.

Sensor Infrarrojos

Una de las mejoras sobre la versión DK1 es la incorporación de emisores de infrarrojos en la carcasa de las gafas, lo que permite que un sensor que se asemeja mucho a una cámara,

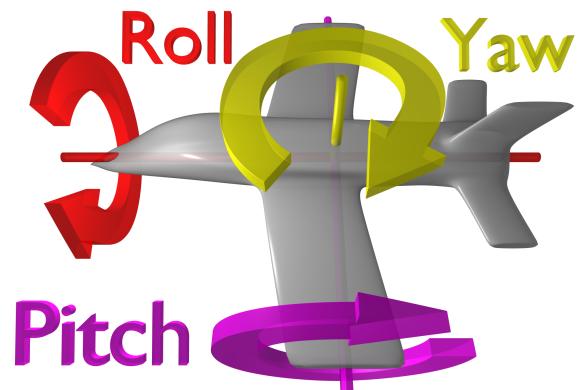


Figura 3.3: Ejemplo de rotaciones posibles sobre un objeto, en este caso un avión.

Fuente:https://upload.wikimedia.org/wikipedia/commons/5/54/Flight_dynamics_with_text.png

figura 3.4 , detecte los movimientos de una persona relacionados con la profundidad. Es decir, el usuario puede acercarse, alejarse, inclinarse a la izquierda o a la derecha y el sistema lo detectará. Esto es una mejora asombrosa, pues genera mucha más libertad de movimiento y nos permite introducirnos más en los espacios tridimensionales.



Figura 3.4: Sensor de infrarrojos de la versión DK2.

Fuente:https://upload.wikimedia.org/wikipedia/commons/9/9c/Oculus_Rift_Development_kit_2_positional_tracker.jpg

Dependencias de hardware

Como ya se ha mencionado antes, las *Oculus Rift/HTC Vive* son dependientes de un ordenador. Tienen conexión HDMI para acceder a la tarjeta gráfica de un ordenador. En realidad son casi como un monitor extra. Casi por que hasta hace no mucho, para determinados juegos, se daba soporte a estos dispositivos configurando las gafas como un monitor externo, aunque de eso se hablará en más detalle en secciones siguientes.

Por este motivo, se requiere de un ordenador medianamente potente para su uso, hecho que genera un coste económico oculto. Este TFG se ha desarrollado con un ordenador con las características recomendadas por *Oculus Rift*.

Estas características son:

- Procesador Intel I7 a 3.2Ghz.
- 16 Gb de Memoria RAM DDR5.
- T.Gráfica *Nvidia GeForce 970X* con 5 Gb DDR5.
- HDD 1Tb rotatorio.
- Windows 8.1

Los requisitos mínimos de este dispositivo son:

- Procesador Intel I5-4590 o similar
- 8 Gb de Memoria RAM.
- T.Gráfica *Nvidia GTX 600 serie AMD Radeon HD 7000 series.*
- Windows 7

3.1.3. Software

Para desarrollar con las *Oculus Rift DK2* el fabricante incluye un SDK bastante completo que proporciona acceso a sus sensores, así como herramientas ya creadas para que el desarrollo sea lo más modular posible. A parte de las descritas a continuación, también existen herramientas para el desarrollo de las *Samsung Gear VR*, unas gafas de realidad virtual para móviles creadas por la compañía para *Samsung*.

SDK y Runtime

Este módulo es el módulo básico que se debe instalar para usar el dispositivo. Contiene los drivers y librerías necesarias para dar soporte [20] :

- **Oculus Audio SDK** para dar soporte a la tecnología VST para el tratamiento y procesamiento de audio [21] y añadir efectos de audio para obtener más sensación de inmersión. Un ejemplo de esto sería que, si giramos la cabeza y ponemos un oído más cerca de la fuente que el otro, el sistema detecta este hecho y aumenta el volumen del canal estéreo correspondiente.
- **Oculus Platform SDK** para el soporte del motor gráfico y de videojuegos *Unreal Engine*.
- **Oculus PC SDK** proporciona drivers básicos así como ejemplos y herramientas para *Windows*. Existe también un hilo paralelo para *OS X* y antiguamente también para distribuciones *Linux* pero fue retirado cuando *Unity 3D* quitó su soporte para dicho SO. Con el runtime se puede desarrollar en *C++* sin necesidad de un editor de videojuegos.

Herramientas para desarrollo

Esta sección se compone de las herramientas y SDK's para desarrollar videojuegos en *Unity 3D* y *Unreal Engine*.

- **Unreal Engine 4 Integration** para dar soporte a *Unreal Engine 4*.
- **Oculus Utilities for Unity 5** para dar soporte a *Unity 3D*.

3.1.4. Configuración del DK2

Para desarrollar con el modelo *DK2* primero el usuario debe ajustarse las gafas de manera personal. Dentro del software básico de drivers, hay un editor de perfil que permite crear y guardar ajustes para cada usuario del dispositivo. Los datos que se guardan son los siguientes:

- **Distancia entre ojos:** Se calcula la distancia entre el centro del iris de cada ojo. Este es un valor necesario para el cálculo de las distorsiones a generar para obtener la fusión de imágenes. Un mal ajuste hace que veamos borroso el mundo virtual. El propio gestor de perfiles tiene una funcionalidad de test en la que se nos hace una prueba para que se autoajuste este valor.
- **Altura del cuello respecto al hombro:** Como se ha comentado, *Oculus Rift DK2* incluye un sensor de posición. Este parámetro es un ajuste para dicho sensor.
- **Altura del usuario:** Este parámetro tiene la misma finalidad y complementa al de la altura del cuello.

Además de los mencionados anteriormente también se almacena el nombre del perfil, el genero y la edad de dicho usuario.

Por otro lado se cuenta también con dos tipos de lentes distintas: las lentes de tipo A están pensadas para personas que no tienen ningún tipo de problema visual mientras que las de tipo B están pensadas para lo contrario. Una de las limitaciones de estas lentes es que las lentes de tipo B generalizan, pudiendo haber muchas combinaciones distintas de defectos oculares que se solucionen con las lentes adecuadas. Una idea de futuro podría ser que en las ópticas se pudieran personalizar dichas lentes.

3.2. Unity 3D

Unity 3D es un software de creación de videojuegos que sigue una filosofía multiplataforma. Dan soporte a casi todos los sistemas actuales: *Linux*, *OS X*, *Windows*, *Android*, *iOS*, *Tizen*, *PlayStation 4* y muchos otros. Por este motivo y por la existencia de un SDK de *Oculus Rift* se ha elegido esta herramienta para desarrollo del TFG.

Unity 3D incluye herramientas para añadir publicidad, hacer análisis estadísticos sobre los jugadores, crear plataformas multijugador y utilizar sistemas de control de versiones. Muchas de estas características se desbloquean al comprar la licencia de uso comercial. En este caso, se ha usado la versión libre, por lo que muchas de estas herramientas no están incluidas.

A continuación se explican algunas de las características más importantes de *Unity 3D*.

3.2.1. C#, JavaScript y Mono Runtime

Para el desarrollo de código se usa *C#* y *JavaScript* que se interpreta en *Mono Runtime*. Este es el secreto de su capacidad multiplataforma. No existe diferencia en las funcionalidades que aporta cada lenguaje a nivel de *Unity 3D*, siendo a gusto del programador el uso de uno u otro. Se aconseja, por otro lado, el uso de *JavaScript* para la creación de funcionalidades simples y de fácil diseño, dejando el resto a *C#*.

C# tiene varias ventajas, entre la que se encuentra que es un lenguaje tipado y no tipado. Es decir, tiene clases y tipos de datos básicos como int, string, double, etc. También posee un tipo

genérico que permite una programación no tipada. Dicho tipo de dato se usa haciendo declaraciones con la palabra reservada *var*. *C#* es un lenguaje parecido a *Java* desde el punto de vista de que se ejecuta sobre una máquina virtual que traduce a código máquina. La máquina virtual se llama *Common Language Runtime*(CLR) y admite más lenguajes como *F#* permitiendo crear proyectos mixtos. *Unity 3D* utiliza *Mono Runtime* que es la versión open source de CLR. *Mono Runtime*[22] es un interprete que se encarga del manejo de las llamadas a sistema operativo y de la ejecución del código. Maneja la memoria de los programas y su liberación a través del *Garbage Collector*. *C#* es un lenguaje con orientación a objetos que incluye la característica de la *Herencia Múltiple*, que permite que una clase obtenga propiedades de más de un tipo distinto de clase padre.

Dado que utiliza *Mono Runtime*, se puede usar el IDE *Mono* para programar, siendo este el predilecto en *OS X*, aunque se potencia más el uso de *Visual Studio*, el IDE predilecto de *Windows*. Uno de los defectos de *Visual Studio* es su tamaño, ya que llega a los 20 Gb debido a que incorpora herramientas para desarrollar con el framework *Windows Forms* y *Xamarin Forms*. Por ese motivo se ha elegido como IDE *Mono*.

3.2.2. Descripción de Unity 3D

A continuación se describirá el uso y características del desarrollo y programación con *Unity 3D* que es el software con el que se ha gestionado y creado este trabajo.

Elementos Gráficos: GameObjects y escenas

La clase *GameObject* [23] es la clase base de todo elemento gráfico de *Unity 3D*. Incluye funciones para instanciarlo, destruirlo, generar transformaciones sobre él u obtener referencia sobre *GameObjects* similares. También puede contener en su interior otros *GameObjects*. Un elemento 2D que represente un botón o un panel sobre el que dibujar también es un *GameObject*.

Los *GameObjects* se instancian en lo denominado escena. Una escena es una representación tridimensional en primer lugar del espacio donde se va a representar el juego. Para juegos en 2D también se usan escenas solo que se juega con las cámaras para que no tenga efecto 3D.

Todo conjunto de *GameObjects* puede guardarse en una plantilla para poder reutilizar más adelante. Dichas plantillas se llaman Prefabs y están creadas para realizar un uso y diseño modular de todos los objetos que creemos. Para incorporarlo al juego solo tenemos que pinchar en él en la ventana de assets y arrastrarlo a la posición que deseemos dentro de la escena.

Parte del SDK de *Oculus Rift* para *Unity 3D* está compuesto de Prefabs que permiten usar el dispositivo con algo tan simple como arrastrar un Prefab a la escena del juego con la que estamos trabajando. En concreto, uno de los Prefabs más útiles que tiene es el de la cámara que ya genera las distorsiones adecuadas para crear imágenes estereoscópicas.

Clase base de control: MonoBehaviour

La clase *MonoBehaviour* [24] proporciona acceso al ciclo de vida que tiene un script (código a ejecutar en una escena) dentro de un *GameObject*. Similar al ciclo de vida de una aplicación de *Android*. Proporciona un método para inicializar variables y atributos, para liberar recursos, y para ejecutar acciones en cada frame (unidad de refresco de imagen) del juego. También proporciona acceso a funcionalidades de *Unity 3D* como la instanciación de *GameObjects*.

El orden de ejecución, de primero a último, es el que se muestra a continuación. De cada apartado se muestran los más relevantes, pues puede llegar a haber varias rutinas por apartado:

■ Editor

- *Reset* Es el método que se invoca cuando el código es añadido al GameObject.

■ Cuando carga la primera escena

- *OnLevelWasLoaded* Es el método que se invoca cuando la nueva escena o nivel ha sido cargado.

■ Antes de la actualización del primer frame

- *Start* Start es llamado antes de la primera actualización de frame solo si la instancia del script está activada.

■ Entre frames

- *OnApplicationPause* Es el método que se invoca cuando se pausa la escena

■ Orden de actualización

- *FixedUpdate* Es el método que se invoca cuando los frames por segundo (FPS) son demasiado bajos.
- *Update* Es el método que se invoca una vez por frame. En esta función se hacen cálculos de movimientos o lógica básica de movimiento.
- *LateUpdate* Es el método que se invoca justo después de Update. Se suele usar en cámaras de tercera persona o similar.

■ Renderizado

- *OnGUI* Es el método que se invoca cuando ocurre un evento en interfaz gráfica. Dicho evento no tiene porqué ser siempre de tipo input.

■ Corrutinas

- *yield* Se ejecuta cuando no hay más objetos a los que invocar la función Update.

■ Cuando el objeto es destruido

- *OnDestroy* Es el método que se invoca cuando el objeto va a ser destruido o eliminado de la escena.

■ Cuando se abandona la escena

- *OnApplicationQuit* Es el método que se invoca cuando la aplicación se va a cerrar.

Estos son los más usados o representativos, pero existen muchos más. Para mas información dirijase a la web de *Unity 3D* donde se explican todos en detalle [25].

Animaciones

Un apartado importante de *Unity 3D* es su capacidad para gestionar y crear animaciones desde el propio editor sin necesidad de terceros, aunque sí permite su importación. Con *Unity 3D* se pueden crear animaciones grabando macros sobre una transformación de un GameObject y controlarlos a través de una máquina de estados. Es decir, se pueden generar diversas animaciones sobre un GameObject, asociar cada una a un estado y definir las transacciones entre estados.

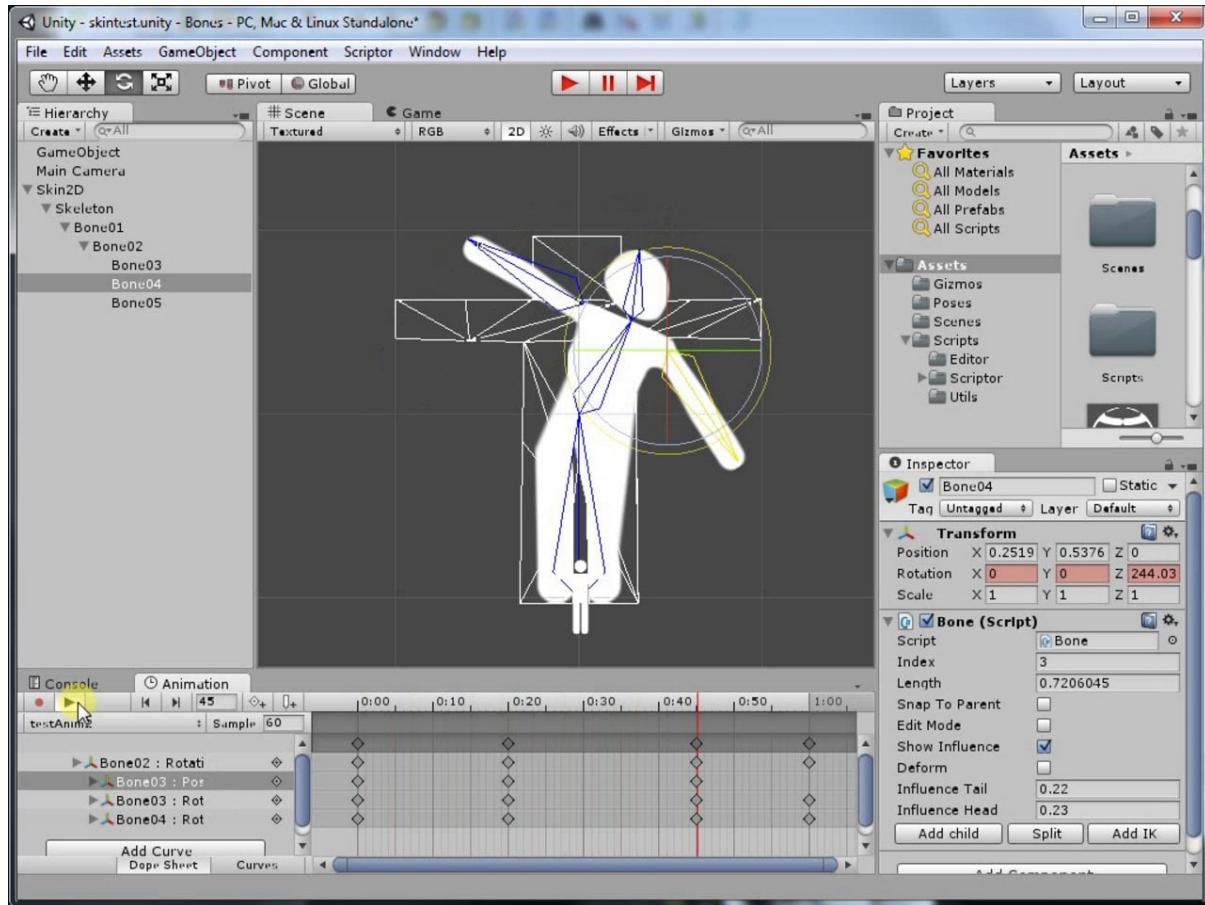


Figura 3.5: Pantalla de animación de *Unity 3D*.

Fuente:https://i.vimeocdn.com/video/459915521_1280x960.jpg

Dichas transacciones se pueden ejecutar al tener como soporte un sistema de eventos que se puede configurar (triggers). Un ejemplo de esto es la figura 3.5.

Grabar una macro de animación es muy simple: solo hay que situarse sobre el GameObject que queremos animar y darle a "añadir nueva animación". Nos saldrá una ventana como la de la figura 3.5. En el cronograma marcamos el 0 como el estado actual del objeto. Pulsamos grabar. Deformamos el GameObject (rotaciones, escalados,etc) y vamos añadiendo marcas de tiempo por cada deformación hasta un estado final. Automáticamente el sistema se encarga de crear la animación haciendo que las deformaciones sean progresivas.

Ya solo nos queda definir cuándo lanzar la animación y eso se hace gracias al sistema de máquina de estados que nos permite definir *Unity 3D*. En él simplemente vamos definiendo estados, partiendo de uno de inicio, y asignando condiciones para que ocurra el cambio de estado. Desde el código vamos lanzado dichas condiciones que pueden ser de dos tipos: triggers (disparadores) o condiciones lógicas.

Recursos y componentes: Asset Store

Una de las principales ventajas de *Unity 3D* es la capacidad para aprovechar el trabajo de otros para poder usarlo en nuestro beneficio personal. Como ya he comentado, el uso de Prefabs permite reutilizar componentes ya creados anteriormente. También se puede llevar a otro punto: existe una tienda virtual, llamada *Asset Store* en la que la comunidad y empresas pueden regalar o vender sus Prefabs y componentes para que otros puedan usarlos. A estos

Prefabs y componentes se les llama Assets y van desde terrenos y animaciones a modelos 3D. Además, es la plataforma desde la que los desarrolladores de *Unity 3D* comparten y distribuyen los Assets básicos a los que puede acceder cualquiera que use *Unity 3D*.

3.3. Requisitos

Cualquier sistema de software responde a una necesidad. Dicha necesidad debe analizarse por partes de manera que sea más sencilla su implementación. Divide y vencerás. En este apartado se hace justamente eso, se divide el sistema en las funcionalidades y requisitos que debe tener. Hay que constatar que no todos los requisitos tienen el mismo nivel de prioridad y que en este TFG no todos han sido resueltos, ya que como se comentó anteriormente este proyecto está desarrollado con la intención de continuarse y de seguir evolucionando. Por ello se dividirá en requisitos funcionales fundamentales y no fundamentales, que serán pospuestos para una versión dos. Muchos de estos requisitos no funcionales derivan de una falta de tiempo para el desarrollo del sistema.

3.3.1. Requisitos funcionales

Los requisitos funcionales son todas aquellas funcionalidades o casos de ejecución que el sistema debe ser capaz de realizar y cumplir.

Fundamentales

El sistema debe tener un diseño modular para que se pueda modificar con sencillez. La modularidad debe ser fundamental en este sistema para que su evolución sea propicia. Este es uno de los principales motivos por los que se ha elegido *Unity 3D* como entorno de desarrollo. Este entorno nos permite, con sus Prefabs, definir objetos que implementan funcionalidades. En este caso, con Prefabs se implementa un navegador web, un sistema de información de fecha y hora constante, un sistema de menú y un sistema de eventos que generan una interfaz de uso con el usuario.

El sistema debe dar capacidad al usuario de interaccionar con los elementos del escenario que le rodea. El usuario ha de ser capaz de interaccionar con lo que le rodea dentro de la escena del mundo virtual. Para ello se han desarrollado los siguientes dos módulos.

- El primer módulo permite una interacción del usuario y los componentes propios gráficos del entorno virtual.
- El segundo permite una emulación de eventos de click de cara al sistema y de pulsación de teclas de teclado, es decir, emula un click de ratón físico y una pulsación de tecla física. En la sección 3.5 se explicarán los motivos.

El sistema debe dar al usuario en todo momento información de fecha y hora actuales. El sistema proporciona información de fecha y hora constante al presentar dichos valores como dos paneles flotantes a la vista del usuario.

El sistema debe disponer de una funcionalidad simple para apagarse. El sistema dispone de un menú en el que existe una opción para apagar el sistema. Hay que ser cautelosos en este punto, pues una persona con tetraplejia puede apagar el sistema pero seguirá con las gafas puestas y con la vista tapada por una pantalla en negro. Debe haber presente un cuidador en este punto.

El sistema debe dar una funcionalidad para navegar por Internet. El sistema dispone de un navegador con el que interaccionar basado en la tecnología *Chromium* [26] de la empresa *Thunderbeast Games LLC* [27] que ha sido modificado para adaptarlo al proyecto en cuestión.

No fundamentales

El sistema debe ser capaz de realizar un autoajuste en función de la discapacidad del usuario. Este requisito se ha clasificado como no fundamental debido a que la complejidad que supone se escapa del ámbito del TFG. El coste de desarrollo de esta funcionalidad es demasiado elevado. Como se ha visto en la sección 2.4 existe un gran número de casos a tener en cuenta. Como punto de partida de desarrollo de esta funcionalidad sería la implementación de un sistema de adaptación al movimiento del cuello del usuario en base a un algoritmo de aprendizaje. Muchos de estos puntos por sí solos podrían ser objeto de estudio de un TFG.

El sistema debe permitir configurar el aspecto visual del escenario principal que rodea al usuario. Uno de los principales motivos por los que Steve Jobs tuvo éxito con su *Macintosh* fue por la idea de añadir una interfaz de usuario amigable y personalizable, alejando a los computadores de la típica terminal de fondo negro y letra verde. Esto demuestra que el éxito de la aceptación del sistema por parte del usuario depende en parte del aspecto y las posibilidades de éste de ser adaptado a los gustos del individuo.

3.3.2. Requisitos no funcionales

Los requisitos no funcionales responden a las necesidades del sistema que no influyen en una funcionalidad o en un caso de uso que debe cumplirse. Engloba características de tipo hardware, económicas o similares.

Fundamentales

El sistema debe ser capaz de ejecutar el mayor número de dispositivos posible con fluidez. Para que se cumpla este requisito es necesario entender que es lo computacionalmente más costoso para el sistema. El mayor coste está en el coste computacional de generar los gráficos. Por ello, el diseño gráfico se ha hecho con texturas sencillas y pocas animaciones. En fases de desarrollo se observó, por ejemplo, que al poner muchos árboles en el entorno 3D con animaciones de viento, la imagen en el *Oculus Rift* iba a saltos.

No fundamentales

El sistema debe poder ejecutarse en móviles Este requisito es deseable y gracias a *Unity 3D* es relativamente sencillo de cumplir. El problema por el cual se ha delegado a un requisito no fundamental es debido a la complejidad de cumplir algunos de los requisitos funcionales que llevarían el desarrollo de este trabajo al doble de tiempo. Se especificarán estos motivos en la sección 3.5.

3.4. Diseño del sistema

El diseño de este sistema es algo complejo dado que está muy influenciado por los propios patrones que fuerza *Unity 3D* a seguir. Primero se explicará la estructura en la que se ha dividido el código en función de las funcionalidades de cada nivel de la jerarquía. A continuación se explicarán los patrones *Façade*, *Modelo-Vista-Controlador* y *Composite* que son el patrones base que se usan.

3.4.1. Estructura del código

En *Unity 3D*, todo el código, Prefabs o similar están en la carpeta denominada *Assets*. Dentro de esta carpeta se encuentran los siguientes directorios:

Cardboard

En este directorio está el SDK básico para la generación de una versión destinada a dispositivos móviles. Incluye scripts de corrección de distorsión para las imágenes estereoscópicas así como un Prefab. Además da una herramienta de debug que permite hacer pruebas en etapa de desarrollo sin tener a mano unas *Google Cardboard*.

Scenes

Aquí se encuentra la escena de *Unity 3D* donde se desarrolla toda la acción. Se recuerda que es en las escenas donde se posicionan los *GameObjects* en los que se basa el sistema.

System

En esta carpeta están las implementaciones que se han realizado en este TFG. Están las animaciones desarrolladas, los fuentes sobre los que se ha basado este TFG y las implementaciones creadas a partir de estos.

- **Animations.** En este directorio se almacenan las animaciones creadas para este TFG. Existen en este momento dos: la primera corresponde a la animación de la mirilla para dar un feedback al usuario cuando se vaya a ejecutar un evento de click y la segunda es la animación destinada al despliegue del menú del sistema.
- **PeterKoch.** En este directorio se almacena el software obtenido de [28] que se usa para implementar un sistema Gaze Input, el necesario para la interacción con los objetos tridimensionales del sistema. Se basa en un *RayCaster*, que no es más que un vector que sale perpendicular a la cámara, cuya función es lanzar y emitir un evento que será registrado por el sistema. Es lo que se usa, por ejemplo, en un videojuego de disparos para que el sistema sepa a qué punto se está apuntando.
- **Prefab.** En este directorio se almacenan aquellos Prefabs que están creados para añadir de manera sencilla funcionalidades al sistema. Existe un Prefab que incluye la cámara del sistema con la fecha, la hora del sistema y la mirilla con la que apuntar. De este Prefab existen dos versiones, una para Pc's y otra para móviles.
- **Scripts.** En este directorio se almacenan aquellos scripts para el desarrollo de las funcionalidades del juego, por ejemplo, el script que toma la fecha y hora del sistema y la plasma en vistas.

Existen más directorios que se omiten por ser irrelevantes. En dichos directorios hay componentes como el cielo que se presenta en el juego o ejemplos de uso de los componentes.

3.5. Desarrollo del sistema

Para el desarrollo del sistema se ha usado el patrón de diseño *Façade*, dado que permite enmascarar el uso de uno o varios sistemas complejos tras una interfaz simple y constante [29] permitiendo así que se puedan cambiar de manera simple dichos sistemas complejos sin que se vea afectado el sistema global. Este patrón se usa en el desarrollo del sistema de emulación de eventos de click y de pulsación de teclas del teclado. En el anexo B se puede observar el código en el que se implementa este desarrollo.

Existe otro tipo de eventos, que son eventos de colisión del *RayCaster* (explicado en la sección 3.4.1) con objetos, que son manejados por el propio sistema de *Unity 3D* y que por tanto escapan a la posibilidad de diseño siendo ésta heredada del motor de gestión propia de *Unity 3D*.

Los patrones que utiliza *Unity 3D* que más se pueden percibir y que son los que más trata el usuario son los siguientes:

- **Composite** Este patrón se caracteriza por su facilidad para implementar herramientas complejas a partir de elementos más sencillos. Suele estar presente en aquellos diseños de interfaces gráficas. En caso de *Unity 3D* no es distinto. Aunque este patrón de diseño no se haya usado en el código de manera estricta, sí ha sido aplicado en el diseño del menú, por ejemplo. El menú se compone de un canvas (panel sobre el que dibujar) al que se le añaden botones [30].
- **Modelo-Vista-Controlador (MVC)** Este patrón se basa en el concepto de la modularidad. La idea es separar la representación física de un componente, de su representación conceptual y de su implementación final. Es el modelo básico de *Unity 3D* dado que es como se enfoca a usar los *GameObject*. Un diseño bajo este patrón se corresponde de tres partes. La primera es el *Modelo*. El *Modelo* corresponde con la estructura de los datos a representar. La segunda parte corresponde con la *Vista*. La *Vista* es la representación de dicho modelo. Por último está el *Controlador* que es el puente de unión entre ambos. En *Unity 3D* el *GameObject* es la vista del elemento que percibe el usuario. Por detrás están los scripts desarrollados que extienden de *Monobehaviour*. Es decir, la clase *Monobehaviour* es el modelo, el *GameObject* de la escena la vista y el software que invoca las funciones heredadas de la clase *Monobehaviour* es el controlador [31].

Estos patrones son los que más afectan al usuario a la hora de realizar una funcionalidad en *Unity 3D* pero no quiere decir que sean los únicos. *Unity 3D* es un sistema muy complejo que en su fero interno aparecen más patrones como *Observer*, *Coroutines*, *Singleton*, etc.

3.6. Diagrama de clases

En las figuras 3.6 y 3.7 se pueden observar los diagramas de clases que corresponden al diseño de este trabajo.

En la figura 3.6 se aprecia las vistas del diseño, según MVC. Las clases que descienden de *GameObject* son los objetos tridimensionales de la escena. Están compuestas por otros componentes que también descienden de la clase *GameObject* y que se omiten por que no aportan valor (botones, entradas de texto, etc). Un ejemplo de la fusión de los patrones *Composite* y *MVC*.

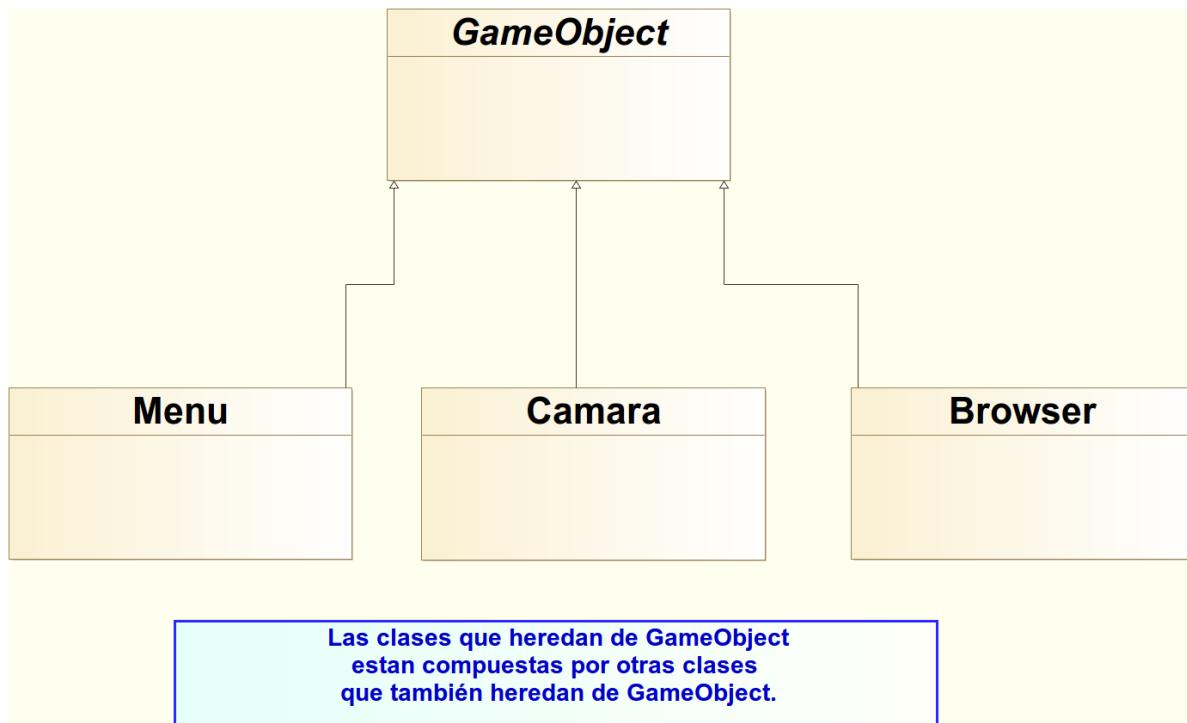
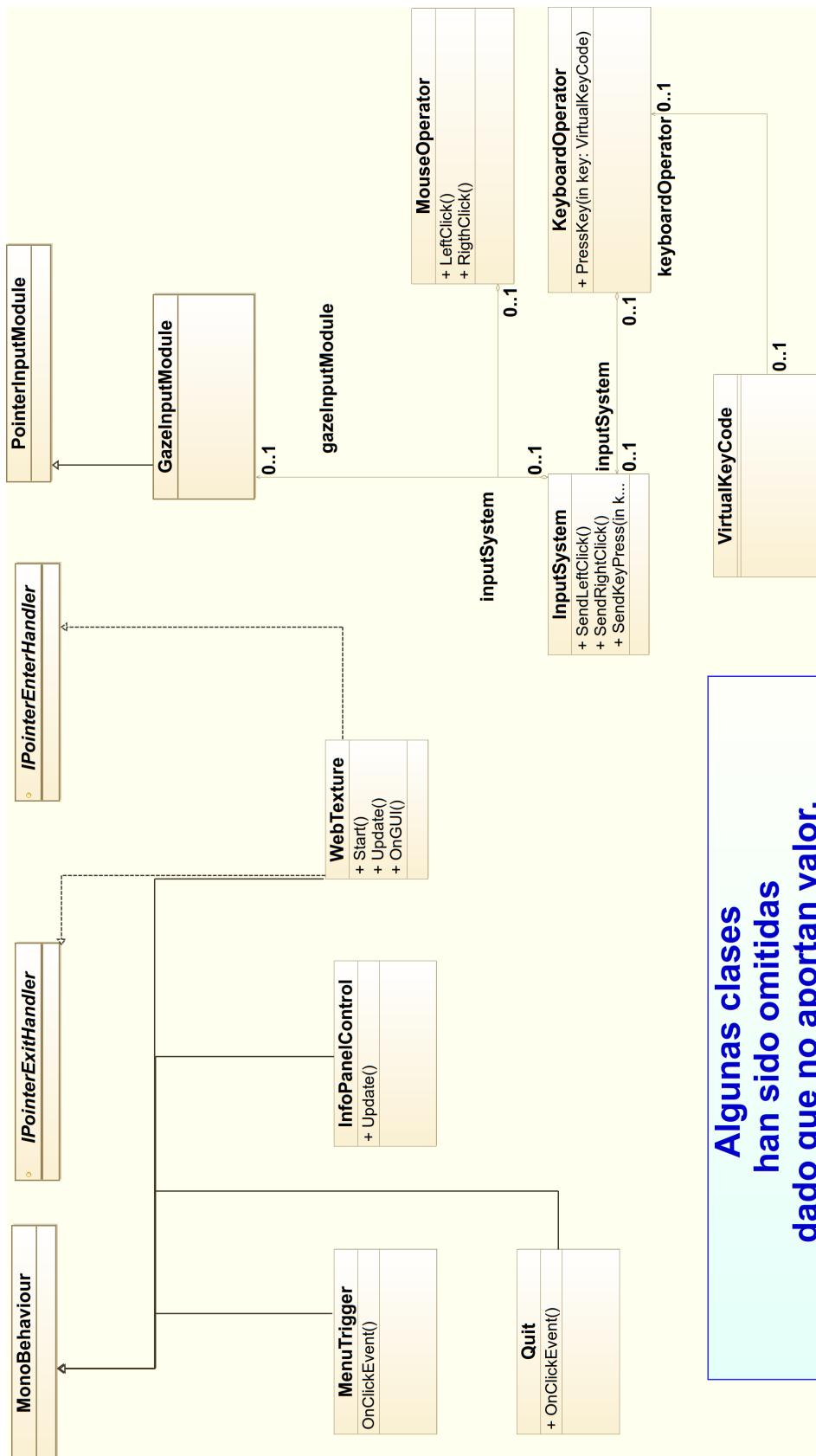


Figura 3.6: Diagrama de clases de la aplicación. Muestra las Vistas del patrón MVC

En la figura 3.7 se observa el diagrama de clases orientado a los controladores, según MVC. A continuación se añade una breve descripción de las clases mas relevantes:

- Las clases que descienden de *Monobehaviour* son independientes porque es el propio sistema el que ejecuta el método *OnUpdate* una vez por frame.
- La clase *InfoPanelController* se encarga de mostrar la hora y la fecha.
- La clase *MenuTrigger* se encarga de controlar que aparezca o desaparezca el menú y de las opciones de éste.
- La clase *Quit* se encarga de cerrar la aplicación. La clase *WebTexture* se encarga de controlar el navegador de Internet.
- Las clase *GazeInputModule* es la que se encarga de la interacción con los GameObject del sistema. Además maneja la funcionalidad para activar la animación de la mirilla. Implementa el patrón *Façade* al usar los módulos que permiten emular el evento físico de click y pulsación de teclado. Se decidió el uso del patrón *Façade* debido a la intención de, en un futuro, sacar una versión para dispositivos móviles. Gracias a este patrón, simplemente cambiando las clases *MouseOperator* y *KeyboardOperator* por dos clases específicas para la plataforma móvil (una que reproduzca un evento táctil en la pantalla y otra que escriba en el buffer de texto del sistema) se consigue adaptar el sistema a dispositivos móviles.



**Algunas clases
han sido omitidas
dado que no aportan valor.**

Figura 3.7: Diagrama de clases de la aplicación. Muestra los Controladores del patrón MVC

3.7. Casos de uso

A continuación se expondrán distintos casos de uso de la aplicación que han servido como referencia de diseño e implementación de este trabajo. En dichos casos de uso se pueden observar dos actores: El actor *Usuario*, es una persona en situación de discapacidad que no puede ponerse por si solo los cascos de VR. Por ello aparece el segundo actor que es un cuidador.

El primero de los casos es un caso de uso simple: el usuario debe ser capaz de interaccionar con un objeto del mundo virtual. Para ello se diseña un caso de uso en el que el usuario tiene que abrir un menú desplegable y a posterior seleccionar la opción *Salir* (figura 3.8).

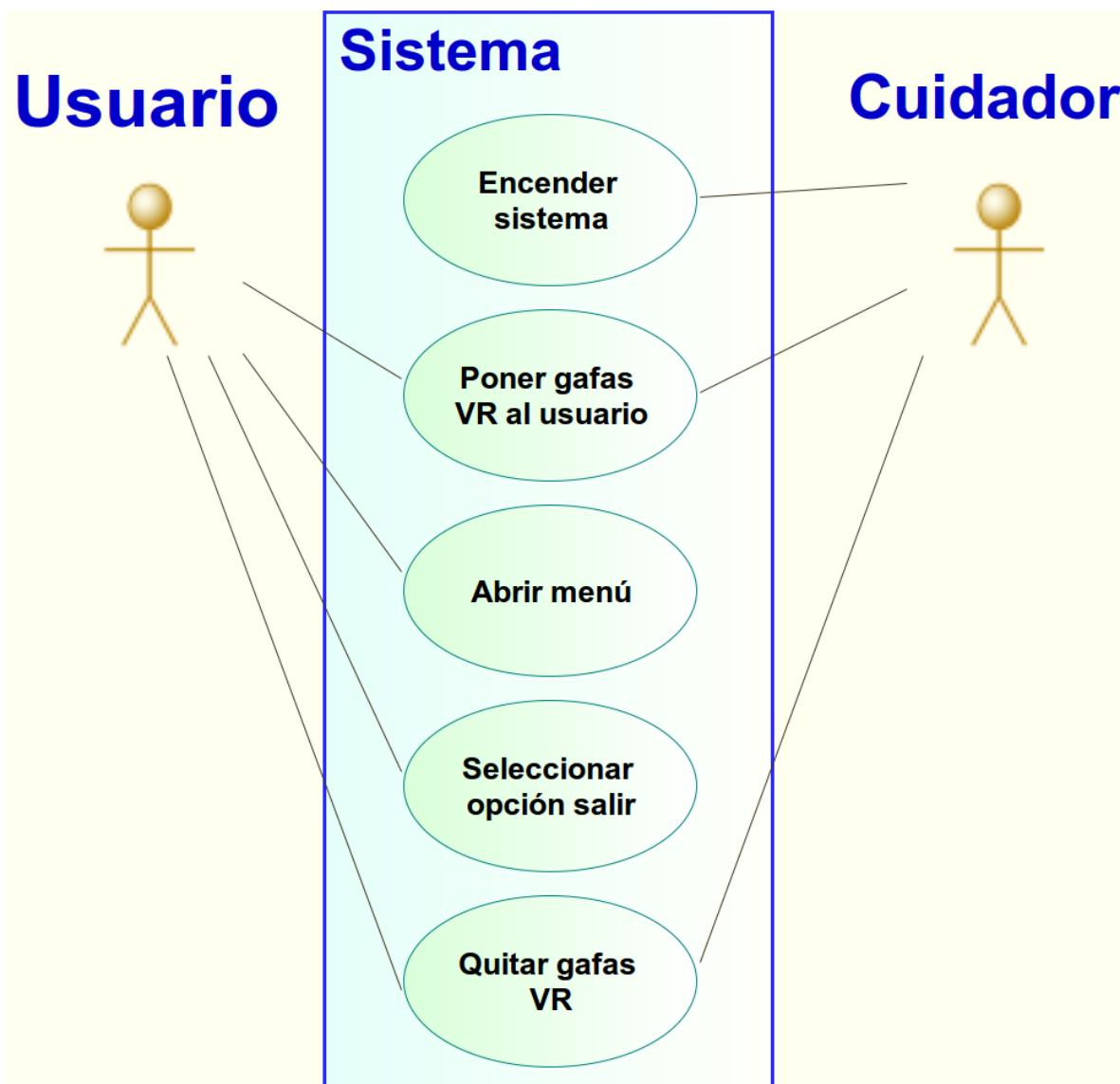


Figura 3.8: Diagrama de caso de uso 1

El segundo caso de uso es en el que se representa la apertura del navegador y el cierre del mismo. El usuario debe abrir el menú desplegable, seleccionar la opción *Open Browser* y esperar a que este se abra. A continuación debe interaccionar con el mismo botón del menú para cerrar dicho navegador (figura 3.9).

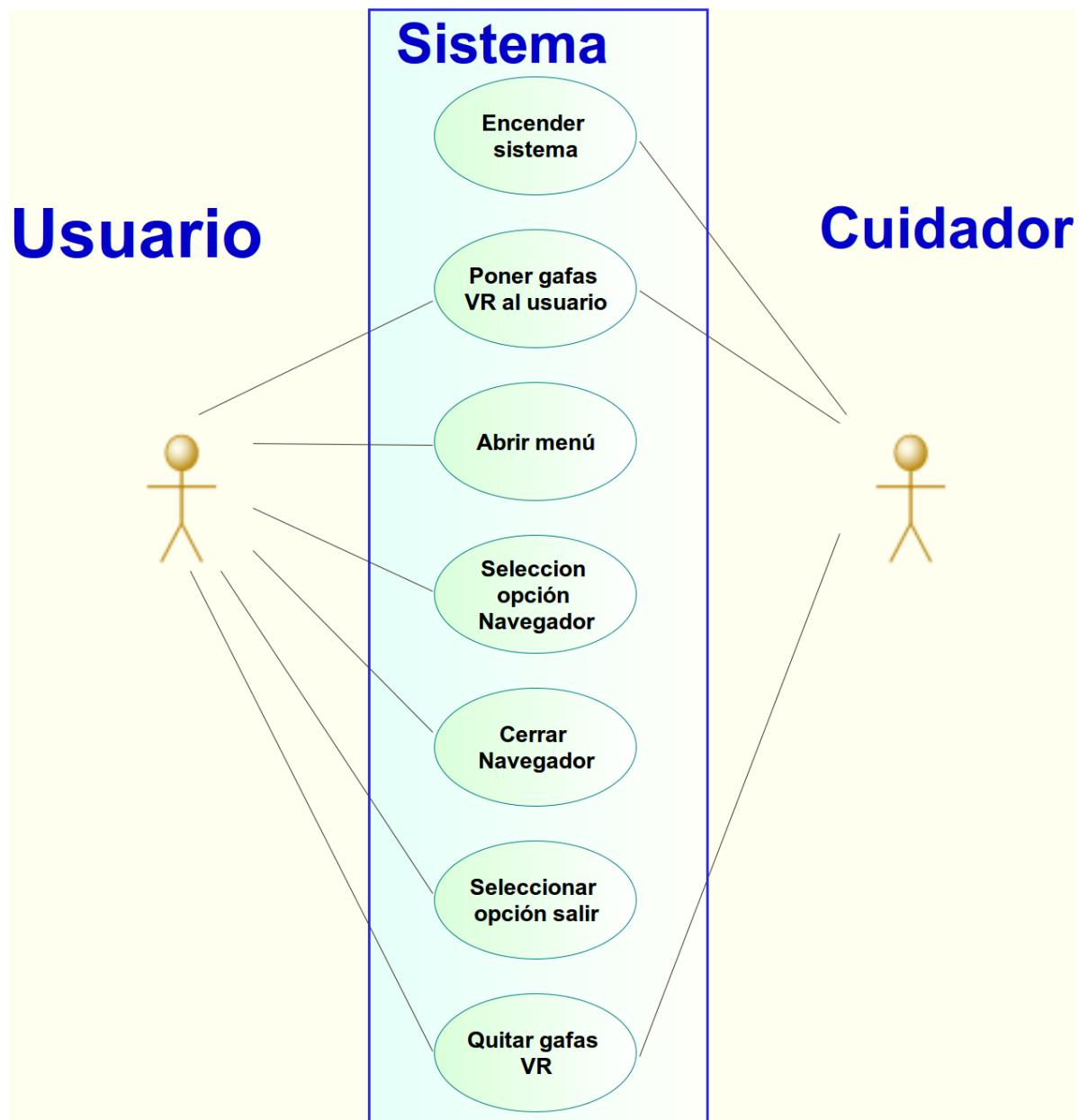


Figura 3.9: Diagrama de caso de uso 2

El tercer caso de uso es en el que se representa la apertura del navegador y la navegación por Internet. Para ello el usuario debe abrir el navegador, seleccionar el campo donde escribir la URL, interaccionar con el teclado para escribir la url www.youtube.es, darle al botón *Go* para cargar la página, seleccionar un vídeo a reproducir y después cerrar el navegador y el sistema (figura 3.10).

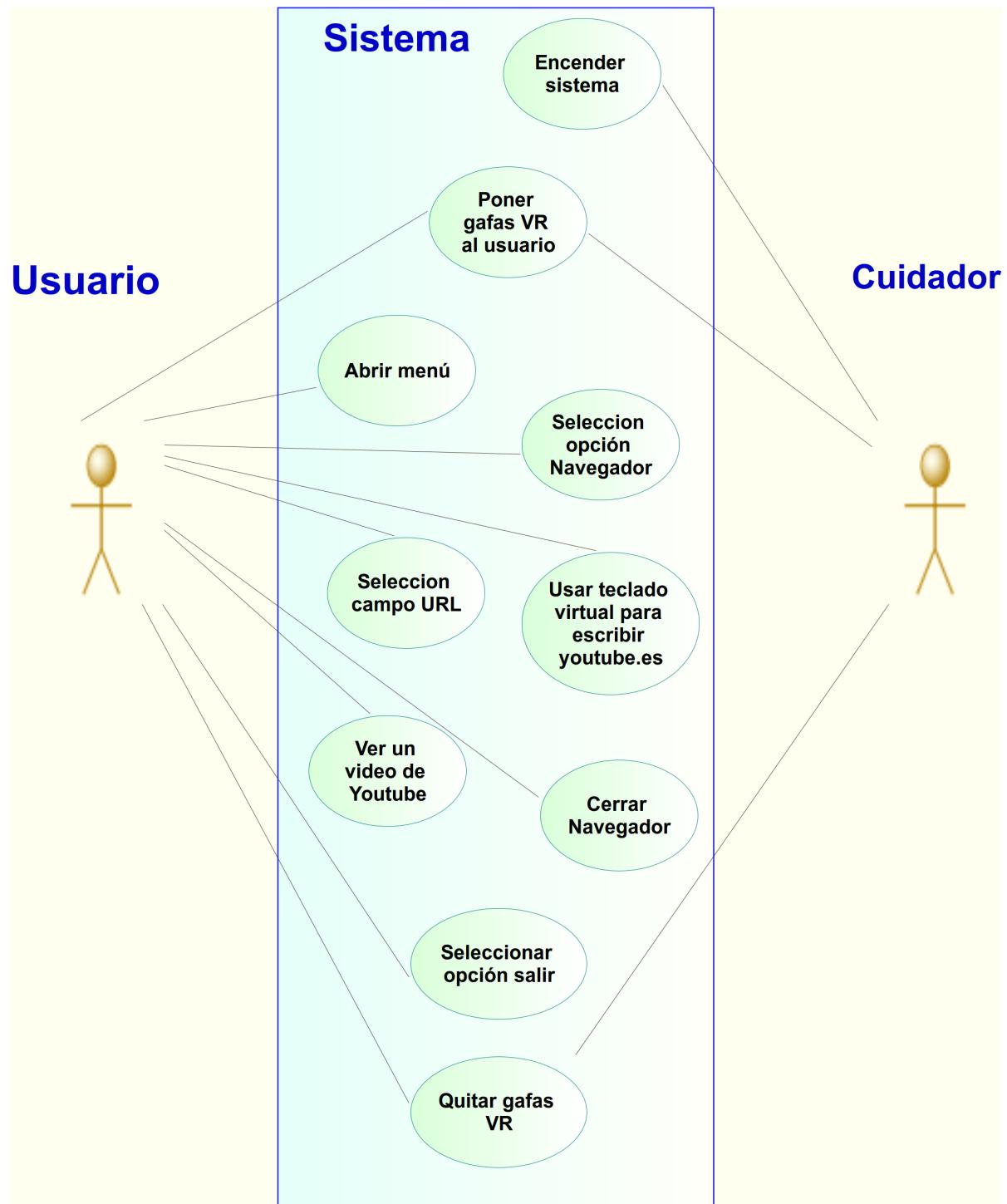


Figura 3.10: Diagrama de caso de uso 3

En el anexo A se encuentra una colección de figuras del sistema que representan cada caso de uso. Dicho anexo está dividido en secciones que corresponden a cada caso descrito.

3.8. Problemas encontrados y soluciones

En el desarrollo de este TFG surgió un contratiempo que tuvo una gran repercusión en el proyecto. El problema era que la funcionalidad para implementar un navegador web no estaba soportada por *Unity 3D*, lo que obligó a buscar un software de terceros llamado *uWebKit*, pues el desarrollo de un navegador está fuera del ámbito de este TFG dado que su tiempo estimado de desarrollo es demasiado alto. La consecuencia del uso de este software fue tener que dejar de lado el desarrollo para dispositivos móviles al no dar este software de terceros soporte a estas plataformas. Dicho software es de pago, siendo el coste de su licencia de unos 150€. Afortunadamente, tras hablar con la empresa responsable por correo electrónico y contarles las intenciones de este TFG y su finalidad académica, decidieron proporcionar una licencia de desarrollo gratis [27].

Tras esto, hubo que adaptar dicho software a las necesidades de este TFG pues no estaba destinado al uso de un método de entrada como el usado en este trabajo. EL método de entrada esperado para este software era un ratón y un teclado. Por este motivo fue necesario desarrollar un módulo capaz de emular eventos de click en el sistema y otro módulo capaz de emular el evento que produce una pulsación de una tecla del teclado. Para ello se usaron dos *DLL* (una teclado, otra ratón) que invocan llamadas al sistema para emular estas funcionalidades.

El software de terceros utilizado [27] usa la posición del ratón de la siguiente manera: obtiene de *Unity 3D* la posición del ratón respecto a la escena y la adapta al sistema de referencia de la web. Por eso una vez resuelto el contratiempo de la emulación de clicks hubo que enfrentarse al problema de transformar la posición de la mirilla a la posición del ratón. Esto se consiguió gracias a los *RayCaster* de *Unity 3D* que básicamente proporcionaban información sobre las coordenadas a donde se apunta respecto a la escena. Solo fue necesario modificar que en lugar de obtener la posición inicial del ratón, la obtuviera de este *RayCaster*.

Antes de desarrollar la funcionalidad con este software se indagó en otras opciones de software libre como *Awesomium* [32]. Aunque existe un apartado de esta herramienta destinado a *Unity 3D*, está para una versión obsoleta (versión 3.4f) con lo que no se puede usar en la última versión de *Unity 3D* (5.2.1f) que es con la que se ha desarrollado este trabajo. La posibilidad de bajar de versión quedó descartada pues tanto el SDK de *Google Cardboard* como el de *Oculus Rift* no daban soporte a tales versiones de *Unity 3D*.

Se exploraron otras opciones como el proyecto *Chromium* [26] pero no se podía incluir de manera rápida a *Unity 3D* y en concreto a *C#*.

Para versiones móviles existen alternativas adecuadas. Android ofrece un webViewer de manera nativa y en iOS está implementado *Safari* para uso de los desarrolladores con lo que no es difícil implementar estas funcionalidades en dichas plataformas. No se ha implementado debido a la falta de tiempo.

4

Experimentos Realizados y Resultados

En este apartado se explicarán los dos experimentos realizados y su resultado. El primero es un estudio simple sobre los efectos del uso de gafas de realidad virtual y el segundo es un estudio de comodidad y usabilidad del usuario final.

4.1. Estudio de efectos de las gafas de realidad virtual

El objetivo de este estudio es entender cómo afecta el uso de un HDM en las personas que lo van a usar. Existen multitud de estudios que demuestran que el uso de la realidad virtual puede provocar mareos o distorsión de la realidad [2]. En este experimento se ha escogido un conjunto de personas sin ningún problema motor. La idea es proporcionar al usuario un estímulo dentro de un mundo virtual con la intención de marear y provocar una distorsión de la realidad.

Para esta prueba se ha utilizado un videojuego de control de un helicóptero en un PC. El escenario en el que se sitúa el helicóptero es una isla con un portaaviones y una plataforma de aterrizaje en el agua. Además, en el centro de la isla, pero a una considerable altura, hay un globo con cajas colgando y un misil dando vueltas alrededor del globo. Este espacio virtual no ha sido creado por mi sino que es una demo destinada al uso de *Oculus Rift*. Su nombre es *Heli-Heli* [33].

Para este pequeño estudio se ha utilizado las *Oculus Rift DK2* como dispositivo HMD. Para controlar el helicóptero se usa un joystick de la casa *Thrustmaster*. Se utilizan también unos cascos de audio para obtener una sensación mayor de inmersión.

4.1.1. Metodología

La metodología de este experimento es bastante sencilla: primero se ajustan las *Oculus Rift* de manera personalizada creando un nuevo perfil por usuario. Además, se tiene en cuenta si el usuario tiene algún tipo de problema visual para seleccionar las lentes de tipo A o B, siendo las de tipo B para personas con defectos visuales. Se toma nota del tipo de defecto visual que tiene el usuario en caso de tenerlo y de si suele jugar habitualmente a videojuegos (denotado como si es *Gamer*). Se le ponen las *Oculus Rift* al usuario y se le explica en qué consistirá el

experimento. Por último se colocan unos cascos de audio para obtener una mayor sensación de inmersión en el juego.

A continuación se realiza la prueba: se deja al usuario entre cinco y diez minutos de vuelo libre para que se habitúe al entorno virtual. El tiempo varía en función de la pericia del usuario. A continuación, el evaluador toma los controles del helicóptero pudiendo ver el mundo virtual en el monitor del usuario sin tener que quitarle las gafas. El evaluador empezará a realizar movimientos bruscos con el helicóptero hasta provocar que este se descontrolé y se estrelle en el suelo. Esta parte no dura más de tres minutos. Rápidamente, tras estrellar el helicóptero, se le quita al usuario las gafas y los cascos de audio. En la URL del sistema (<https://www.dropbox.com/sh/8t9zp913xyq49dt/AAAY5N9I96hA9H5zjL4hzKUpa?dl=0>) se encuentra un vídeo en el que se muestra un ejemplo de este experimento.

Se le pregunta al usuario si ha sentido vértigo, mareos y/o distorsión de la realidad.

4.1.2. Resultados

Este experimento se realizó a seis individuos que dieron los siguientes resultados:

Resultados				
Tipo Lente	Gamer	Vértigo	Mareos	Distorsión Realidad
B	Sí	No	Sí	No
B	No	No	Sí	Sí
A	No	No	No	Sí
A	Sí	No	Sí	No
B	No	No	Sí	Sí
A	Sí	No	Sí	No

Cuadro 4.1: Resultados de test de efectos de las *Oculus Rift DK2*. La primera columna corresponde al tipo de lente. La segunda columna a la pregunta de si el usuario suele jugar a videojuegos. La tercera columna corresponde a la pregunta de si tras la prueba, el usuario ha sufrido vértigo. La cuarta columna a si a sufrido mareos. La quinta columna representa la respuesta a la pregunta de si el usuario ha sufrido distorsión de la realidad.

4.1.3. Conclusiones

Tras observar los datos de la tabla 4.1, aunque sean poco representativos dado el pequeño tamaño muestral, podemos sacar como conclusión que los usuarios no tienen sensación de vértigo dada la altura del helicóptero virtual. Esto se puede explicar razonando que la sensación de inmersión no es lo suficientemente alta. No tenemos realmente sensación de altura. Por otro lado, casi todos los usuarios se marearon debido a los rápidos movimientos del helicóptero al dar tumbos.

También se observa que los usuarios que tienen tendencia a jugar habitualmente a videojuegos están más acostumbrados a escenarios 3D y no tienen tanta distorsión de la realidad al hacer el cambio repentino del mundo tridimensional al real. Esto demuestra que este sistema tiene una etapa de entrenamiento y que la capacidad para usarlo durante más tiempo va incrementando con su uso.

Estos resultados nos dan a entender que el uso prolongado de un sistema de VR es perjudicial. Si nos fijamos en la tabla 4.1 podemos observar como todos los usuarios menos uno sufrió de mareos. *Oculus Rift* muestra al inicio de cualquier aplicación un mensaje de advertencia del uso prolongado de su sistema. En este trabajo esto es un punto muy importante a tener en cuenta pues una PSD no tiene la capacidad para quitarse por si solo las gafas. Debe estar bajo la supervisión de un cuidador.

4.2. Estudio de comodidad y usabilidad del sistema

Por último se ha realizado un test de usabilidad. Las herramientas de accesibilidad van muy por detrás del uso común del PC por lo que cabe esperar que esta herramienta frente a un uso convencional del PC sea más lenta. Para hacer esta comparativa se han seleccionado otra vez usuarios que no tienen ningún problema o trastorno de movimiento dado que realizar estas pruebas con personas en situación de discapacidad es demasiado difícil debido la preparación necesaria, los permisos y los trámites burocráticos.

Para esta prueba se utiliza el sistema desarrollado. Cuenta de un menú desplegable, al que se accede a través del sistema de interacción con la mirada, con una opción para abrir el navegador. Un ejemplo del despliegue y de la selección de la opción para abrir el navegador se puede ver en el anexo A.3.

El escenario físico de este sistema es un PC y las *Oculus Rift DK2*.

4.2.1. Metodología

La idea es contrastar el sistema frente al uso convencional de un PC. Para ello primero se temporiza el tiempo que tarda en realizar una tarea determinada con un sistema de entrada común y al que está acostumbrado, es decir, teclado y ratón. Luego se hace una medición de lo que tarda el usuario con el sistema actual tras haberle explicado verbalmente, sin previo entrenamiento (SE), como utilizar el sistema. A continuación se le da entre tres y cinco minutos de uso libre con el sistema tras los cuales se realiza otra medición de tiempo. La intención con estos minutos de uso libre es que el usuario se adapte al sistema y entrene con él (CE).

La tarea a realizar es la misma en los tres pasos y consiste en entrar al portal de *Youtube* a través de la URL www.youtube.es, en el buscador escribir la palabra *eps* y seleccionar el primer video.

Para la primera parte se realiza dicha tarea con el navegador *Mozilla Firefox* y el teclado físico. La segunda y tercera parte se realiza con el software presentado para este TFG usando la opción de navegador y el teclado implementado. Un ejemplo de esto puede verse en el anexo A.3.

En la URL del sistema (URL!!!) se encuentra un vídeo en el que se muestra un ejemplo de este experimento.

4.2.2. Resultados

Este experimento se realizó con diez individuos acostumbrados al uso de las TIC pero sin previa noción del sistema presentado en este TFG. Se obtuvieron los siguientes resultados:

Resultados			
Tipo Lente	T.Firefox	T.Sistema (SE)	T.Sistema (CE)
B	19,66	146,26	45,06
A	15,03	142,21	60,26
A	12,49	116,02	59,96
B	13,57	83,33	93,16
A	14,75	134,09	115,31
A	20,39	112,57	111,44
B	16,79	97,07	80,06
A	13,60	106,46	83,22
A	14,71	56,29	55,05
A	16,54	77,17	71,36

Cuadro 4.2: Resultados de test de comodidad y usabilidad. La primera columna corresponde al tipo de lente usada. La segunda columna corresponde al tiempo usando teclado, ratón y el navegador *Mozilla Firefox*, la tercera columna corresponde a los tiempos usando el sistema de este trabajo sin entrenamiento, la cuarta columna corresponde a los tiempos del sistema desarrollado en este trabajo con entrenamiento de tres a cinco minutos. Resultados de tiempos en segundos.

Media de Resultados		
T.Firefox	T.Sistema (SE)	T.Sistema (CE)
$15,75 \pm 2,61$	$107,15 \pm 29,34$	$77,49 \pm 23,72$

Cuadro 4.3: Medias y desviación estándar de los datos obtenidos en la tabla 4.2. Resultados de tiempos en segundos.

4.2.3. Conclusiones

Tras observar los datos de las mediciones mostradas en la tabla 4.2, podemos sacar en claro que, como se esperaba, los tiempos entre el uso de un ratón y teclado son mucho menores que los del sistema desarrollado en este trabajo. Por otro lado también se observa que con solo escasos tres minutos de aprendizaje los tiempos de uso se reducen bastante.

Queda claro que este sistema necesita de entrenamiento por parte del usuario. Aun así permite la interacción con las TIC en unos tiempos dentro de un marco aceptable. Estos tiempos podrían disminuirse aun mas mejorando la algoritmia de determinados componentes así como innovando nuevas formas de interacción que no sean las descritas en este trabajo.

5

Conclusiones, discusión y trabajo futuro

Tras hacer este trabajo se puede sacar como conclusión que el desarrollo de un sistema tan amplio esconde muchos problemas que surgen con el paso del tiempo y el desarrollo del mismo. Aun así he podido sentar las bases y generar una herramienta útil y que ayude a las personas. Los dos experimentos realizados han demostrado que, aunque aun le queda mucho camino por recorrer a este sistema, actualmente puede ser utilizado con la esperanza de buenos resultados. El primer experimento a confirmado que hay que tener cuidado con los efectos secundarios de sistemas de VR. El segundo experimento ha confirmado que con este sistema se interacciona con las TIC de siete a cinco veces más lento en valor promedio respecto al uso de teclado y ratón. Aunque, teniendo en cuenta el sector de la sociedad para el que va dirigido y sus dificultades, es un rango aceptable aunque mejorable. La solución al problema de acercar las TIC a las PSD es difícil pero no imposible y la tecnología avanza sin cesar.

No sería posible realizar un trabajo como este sin la previa documentación y estudio de los sistemas a utilizar. Se parte de la base del desconocimiento total de dichos sistemas, como por ejemplo *Unity 3D*. Esta tarea de documentación y aprendizaje ha sido muy costosa en tiempo y en esfuerzo, siendo imposible de realizar si no llegara a ser por cursos online y videotutoriales de Internet [34]. Aun así, el desarrollo de este sistema me ha permitido dar un paso adelante en mi formación. Estos son los conocimientos adquiridos en desarrollo de este trabajo:

- **Unity 3D**
- **C#**
- **JavaScript en profundidad**
- **Teoría del diseño de videojuegos y espacios tridimensionales**
- **Nuevos patrones de diseño (Facade,Proxy,etc)**
- **Uso de sistemas de realidad virtual completa como Oculus Rift**
- **Latex**

He podido comprobar de primera mano cómo una buena planificación y un buen diseño pueden llegar a ser fundamentales. Cuando inicié este TFG y decidí hacer un módulo para la

cámara de Android y otro para la cámara de *Oculus Rift* no sabía que tendría que dejar a medias el desarrollo para Android debido al problema del navegador mencionado. Esto podría haberse vuelto un problema verdaderamente complicado si no llega a ser por el diseño modular.

Por otro lado, he podido profundizar más en un tema que me interesa como es la realidad virtual y encauzar mi futuro hacia la investigación. Quiero recalcar en la cantidad de investigación y pruebas que he tenido que superar y realizar para terminar con el software que se describe en esta memoria.

Como comenté en la introducción de esta memoria, este trabajo tiene la férrea intención de marcar las bases de una herramienta que ayude a las personas. Por este motivo, su desarrollo debe continuar tanto por mi parte como por la de la comunidad, o al menos eso espero. Teniendo en cuenta los problemas que me han surgido en el desarrollo, establezco a través de este apartado los que yo creo que deberían ser los pasos a seguir para el crecimiento y trabajo futuro de esta herramienta:

- **Algoritmo predictivo de movimiento en ciclo cerrado.** Las PSD sufren distintos tipos de impedimentos. Hay casos de parálisis parcial de los movimientos cefálicos que produce que estos movimientos no sean fluidos. El desarrollo de un algoritmo de aprendizaje y adaptación para cada caso aportaría al sistema una funcionalidad que ayudaría muchísimo a este sector.
- **Desarrollar un navegador para móviles.** Este problema fue una desilusión por mi parte, pues tenía la intención de poder sacar una versión para móvil a la par que para ordenador. Lamentablemente, esta funcionalidad no está integrada en *Unity 3D* y el coste en tiempo que supone para su desarrollo es amplio. Herramientas como *Awesomium* [32] y *Chromium* [26] pueden ser un buen punto de partida. La idea de generalizar este trabajo a un dispositivo móvil a mi parecer es muy atractiva, dado que el coste del sistema completo (gafas vr + software) se reduce drásticamente.
- **Desarrollar nuevos módulos como clientes de correo, editores de texto,etc** Esta herramienta proporciona una funcionalidad básica: navegar por Internet. Aun así se podrían desarrollar más módulos que proporcionen nuevas herramientas como un cliente de correo, un editor de textos o un sistema de archivos.
- **Incluir nuevos sistemas de entrada.** Se puede incluir más hardware que permita añadir más funcionalidades. Como se ha visto en la sección 2.5, existen muchos proyectos que ayudan a las personas con discapacidad. Se podría incluir un sistema de seguimiento de ojos (más conocido en inglés como *Eye Tracking*) que permita ver a dónde se está dirigiendo la mirada y mover la cámara acorde a ello. También se pueden incluir sistemas con *BCI* que permitan detectar determinados tipos de señales en las ondas del cerebro ante estímulos proporcionados en el mundo virtual.
- **Mejorar la adaptación del módulo web.** Aunque el módulo encargado de cargar las webs es operativo, la algoritmia encargada de interaccionar con el es escasa permitiendo emular solamente eventos de click izquierdo. La web evoluciona de manera dinámica y eventos como arrastrar y soltar, mantener pulsado o hacer click derecho están a la orden del día.

Glosario de acrónimos

- **3D:** Tercera Dimensión
- **BCI:** Brain Control Interface (Interfaz control cerebral)
- **CE:** Con Entrenamiento
- **CLR:** Common Language Runtime (Entorno en tiempo de ejecución de lenguaje común)
- **DDR5:** Double Data Rate type 5 (Memoria RAM)
- **DK1:** Development Kit 1
- **DK2:** Development Kit 2
- **EEG:** Electroencefalograma
- **FPS:** Frames Per Second (Frames por segundo)
- **Gb:** Gigabyte
- **HDD:** Hard Disk Drive (Disco duro)
- **HMD:** Head-Mounted Display (Pantalla con casco)
- **HW:** Hardware
- **IDE:** Entorno de programación
- **MVC:** Modelo-Vista-Controlador
- **NFC:** Near Field Communication (Comunicación de campo cercano)
- **PC:** Ordenador Personal (Personal Computer)
- **PSD:** Personas en Situación de Discapacidad
- **SDK:** Software Development Kit (Conjunto de herramientas de desarrollo)
- **SE:** Sin Entrenamiento
- **SO:** Sistema Operativo
- **SW:** Software
- **Tb:** Terabyte
- **TFG:** Trabajo de Fin de Grado
- **URL:** Localizador de Recursos Uniforme (Uniform Resource Locator)
- **VR:** Realidad Virtual (Virtual Reality)

Bibliografía

- [1] Psico Smart Apps, S.L. Sistema Psious. <https://psious.com/en>. Accedido: 16-05-2016.
- [2] Seizo Ohyamaa, Suetaka Nishiikeo, Hiroshi Watanabec, Katsunori Matsuokac, Hironori Aizukia, Noriaki Takedaa, and Tamotsu Haradab. Autonomic responses during motion sickness induced by virtual reality. *Auris Nasus Larynx, Volume 34, Issue 3*, page 4303?306, 2007.
- [3] Real Academia Española. Definición realidad virtual. <http://dle.rae.es/?id=VH7cofQ>. Accedido: 17-05-2016.
- [4] HTC. Gafas de vr Htc Vive. <https://www.htcvive.com/eu/>. Accedido: 24-06-2016.
- [5] Google. Gafas de vr Google Cardboard. <https://vr.google.com/cardboard/index.html/>. Accedido: 24-06-2016.
- [6] Virtuix. Sistema de desplazamiento en vr Virtuix Omni. <http://www.virtuix.com/>. Accedido: 24-06-2016.
- [7] The Void 2016. Centro de recreación The Void. <https://thevoid.com/>. Accedido: 17-05-2016.
- [8] Unity Technologies. Motor gráfico Unity 3D. <https://unity3d.com/es>. Accedido: 24-06-2016.
- [9] Blender. Software de diseño 3d Blender. <https://www.blender.org/>. Accedido: 24-06-2016.
- [10] Unity Technologies. Tutoriales oficiales de Unity 3D. <http://unity3d.com/learn>. Accedido: 24-06-2016.
- [11] Manuel Amosa Delgado. Neurología y neurocirugía. *Manual CTO de Medicina y Cirugía*, pages 3–6, 2014. ISBN: 978-84-16153-23-7.
- [12] José Luis Pérez Arellano. Manual de patología general, 6.^a edición. pages 639–641, 2006. ISBN: 978-84-458-1540-3.
- [13] Robert Leeb, Doron Friedman, Mel Slater, and Gert Pfurtscheller. A tetraplegic patient controls a wheelchair in virtual reality. *Hindawi Publishing Corporation, Computational Intelligence and Neuroscience, Volume 2007, Article ID 79642*, page 8.
- [14] Semyon M. Slobounov, William Ray, Brian Johnson, Elena Slobounov, and Karl M. Newell. Modulation of cortical activity in 2d versus 3d virtual reality environments: An EEG study. *International Journal of Psychophysiology 95 (2015)*, pages 254–260.
- [15] Mario Saiano, Laura Pellegrino, Maura Casadio, Susanna Summa, Eleonora Garbarino, Valentina Rossi, Daniela Dall'Agata, and Vittorio Sanguineti. Natural interfaces and virtual environments for the acquisition of street crossing and path following skills in adults with autism spectrum disorders: a feasibility study. *Journal of NeuroEngineering and Rehabilitation (2015)*.

- [16] José Miguel Guerrero Hernández. Técnicas de procesamiento de imágenes estereoscópicas. *Trabajo de fin de máster en ingeniería informática para la industria. Universidad Complutense de Madrid*, pages 1–4, 2010.
- [17] Wikipedia. Definición OLED wikipedia. https://es.wikipedia.org/wiki/Diodo_org%C3%A1nico_de_emisi%C3%B3n_de_luz. Accedido: 18-05-2016.
- [18] Varios. Articulo wikipedia sobre High Frame Rate y el efecto Judder. https://es.wikipedia.org/wiki/High_Frame_Rate. Accedido: 24-06-2016.
- [19] Varios. Articulo wikipedia sobre Motion Blur. https://es.wikipedia.org/wiki/Motion_blur. Accedido: 24-06-2016.
- [20] Oculus VR, LLC. Centro descargas de software para *Oculus Rift*. <https://developer.oculus.com/downloads/>. Accedido: 23-05-2016.
- [21] Wikipedia. Definición Virtual Studio Technology wikipedia. https://es.wikipedia.org/wiki/Virtual_Studio_Technology. Accedido: 23-05-2016.
- [22] Mono Project. Interprete Mono Runtime. <http://www.mono-project.com/docs/>. Accedido: 24-06-2016.
- [23] Unity Technologies. Documentación a la API de GameObjects de Unity 3D. <http://docs.unity3d.com/ScriptReference/GameObject.html>. Accedido: 23-05-2016.
- [24] Unity Technologies. Documentación a la API de MonoBehaviour de Unity 3D. <http://docs.unity3d.com/ScriptReference/MonoBehaviour.html>. Accedido: 23-05-2016.
- [25] Unity Technologies. Orden de ejecución dentro de un GameObject en Unity 3D. <http://docs.unity3d.com/es/current/Manual/ExecutionOrder.html>. Accedido: 25-05-2016.
- [26] Comunitario, varios autores, Open Source. Projecto Chromium. <https://www.chromium.org/>. Accedido: 09-06-2016.
- [27] Thunderbeast Games LLC. Herramienta uWebKit para visualizar contenido html dentro de Unity 3D. <http://uwebkit.com/>. Accedido: 05-02-2016.
- [28] Peter Koch. Sistema gaze input. <http://talesfromtherift.com/vr-gaze-input/>. Accedido: 16-01-2016.
- [29] Craig Larman. UML y Patrones. Introducción al análisis y diseño orientado a objetos. 2.^a Edición. Pearson, pages 418–420, 2003. ISBN: 978-842-05-3438-1.
- [30] Craig Larman. UML y Patrones. Introducción al análisis y diseño orientado a objetos. 2.^a Edición. Pearson, pages 218–224, 2003. ISBN: 978-842-05-3438-1.
- [31] Craig Larman. Patrón Composite. UML y Patrones. Introducción al análisis y diseño orientado a objetos. 2.^a Edición. Pearson, page 359, 2003. ISBN: 978-842-05-3438-1.
- [32] Awesomium. Herramienta Awesomium para visualizar contenido HTML en C# y .NET. <http://www.awesomium.com/>. Accedido: 09-06-2016.
- [33] GriN Multimedia. Simulador de helicóptero Heli-Heli. <http://www.grin.be/blog/?p=975>. Accedido: 11-09-2015.
- [34] Danilo Giardina. Curso Unity 3D impartido por Danilo Giardina. <https://www.tutellus.com/tecnologia/videojuegos/creacion-de-videojuegos-con-unity-3d-3291>. Accedido: 20-09-2015.



Casos de Uso

En este anexo se presenta a través de imágenes los tres casos de uso comentados en esta memoria.

A.1. Caso 1

Recordemos este caso de uso. El usuario debe abrir el menú desplegable y apagar el sistema. Su finalidad es probar el sistema de interacción del sistema.



Figura A.1: Pantalla de inicio del sistema.



Figura A.2: Usuario seleccionado la opción para abrir el menú.

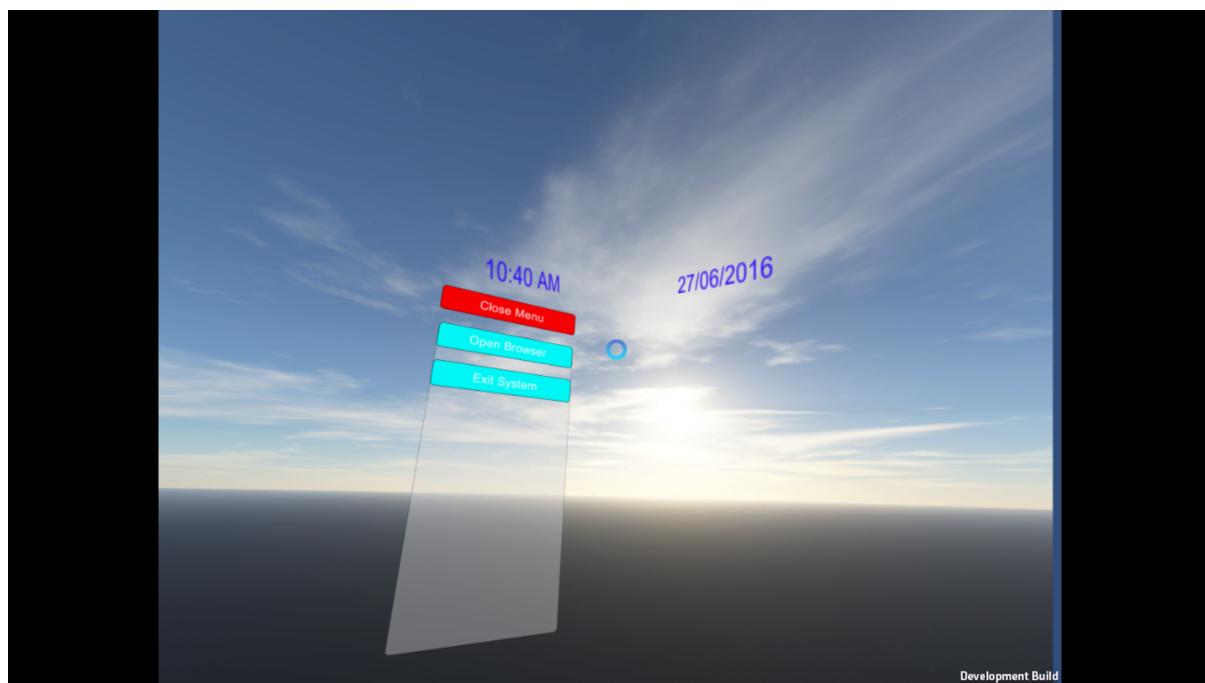


Figura A.3: Menú desplegado.



Figura A.4: Usuario seleccionado la opción para apagar el sistema.

A.2. Caso 2

Recordemos este caso de uso. El usuario debe abrir el menú desplegable y seleccionar la opción del navegador. Tras eso debe cerrar dicho navegador y el sistema.



Figura A.5: Pantalla de inicio del sistema.

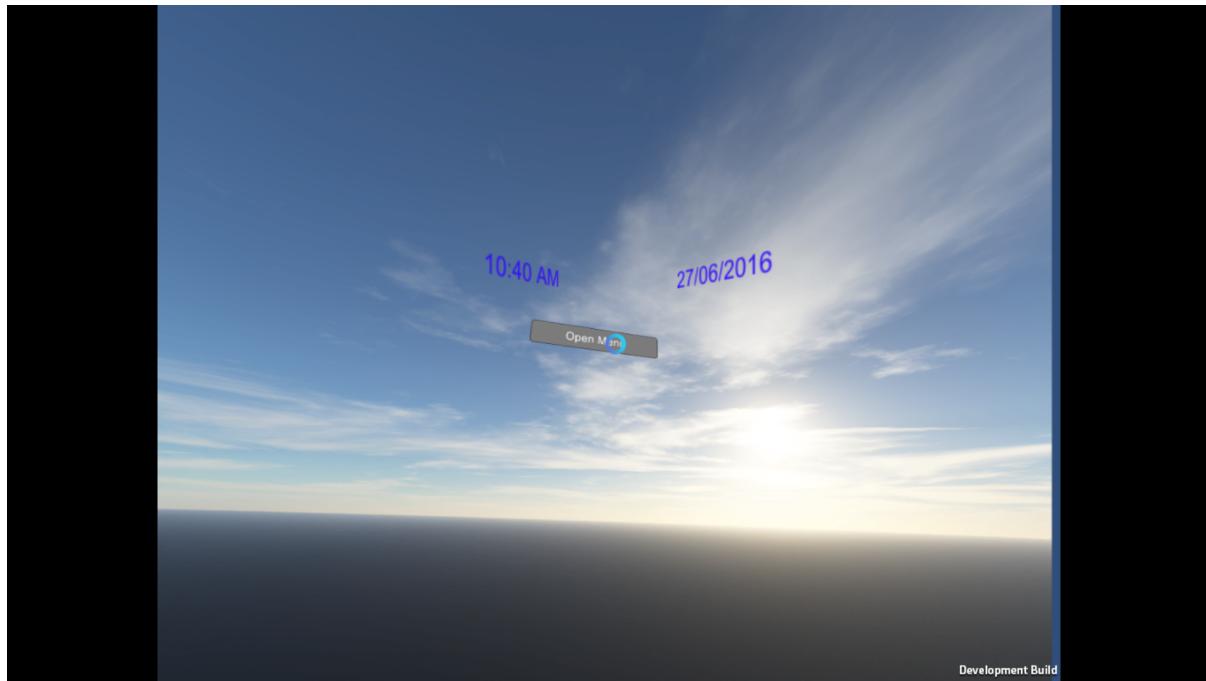


Figura A.6: Usuario seleccionado la opción para abrir el menú.

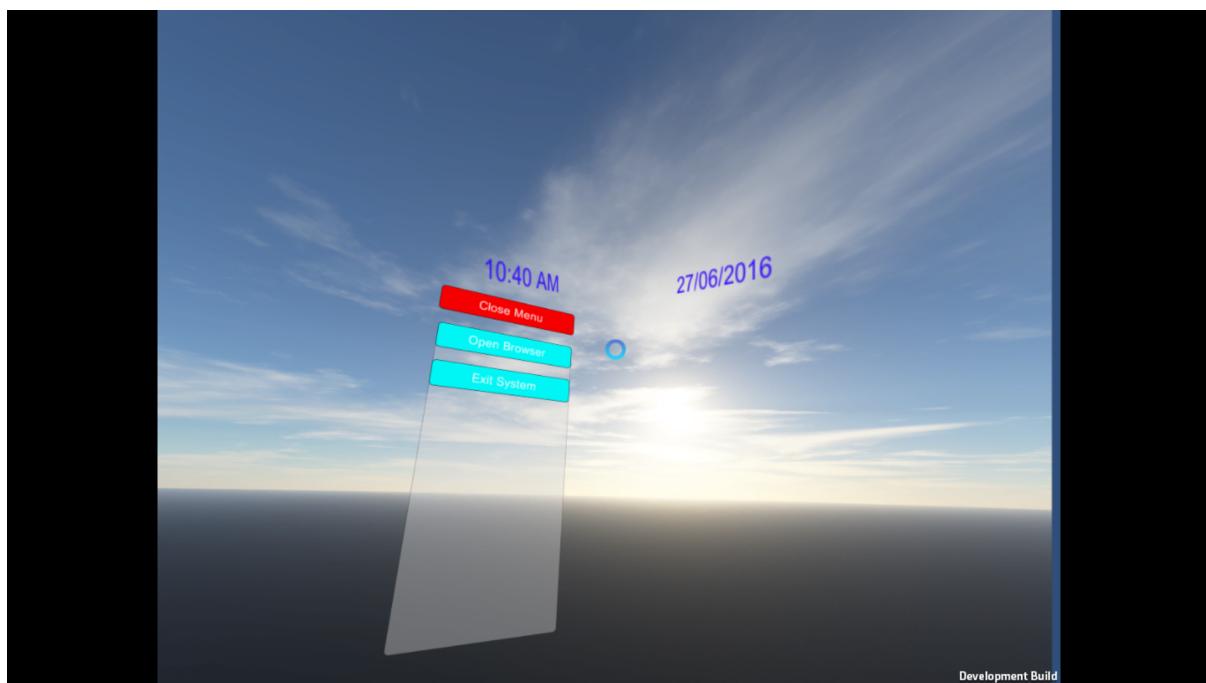


Figura A.7: Menú desplegado.

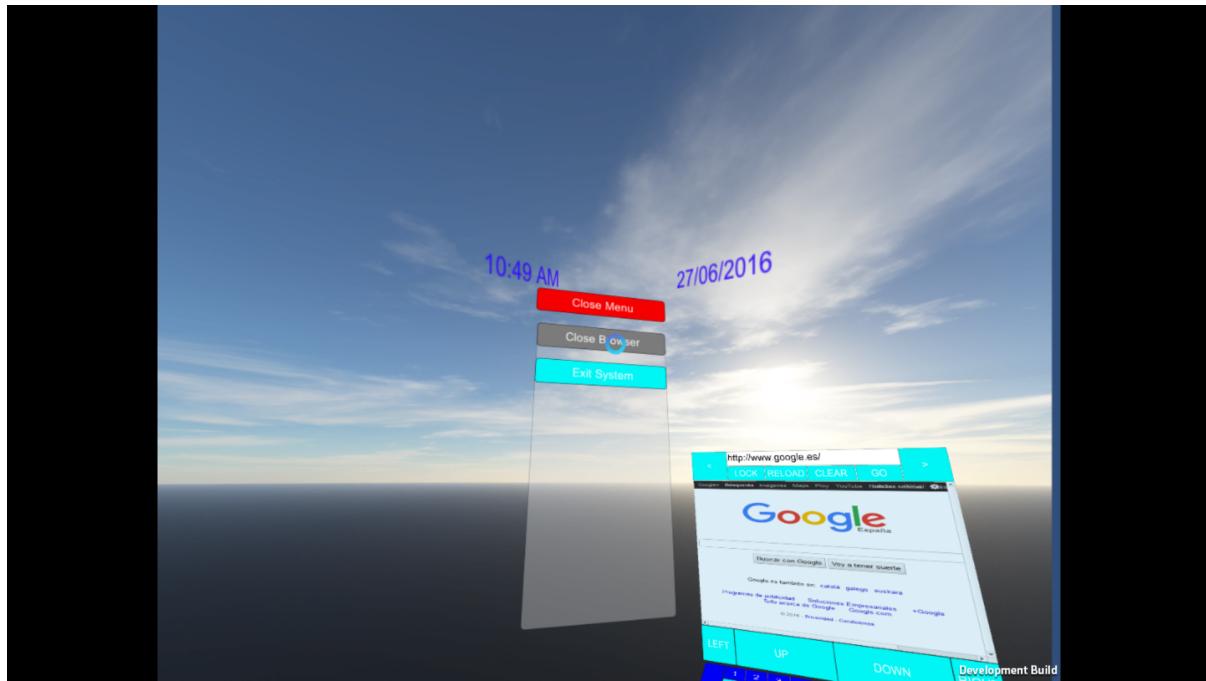


Figura A.8: Usuario seleccionando la opción para abrir el navegador.



Figura A.9: Navegador del sistema.



Figura A.10: Usuario seleccionado la opción para cerrar el navegador.

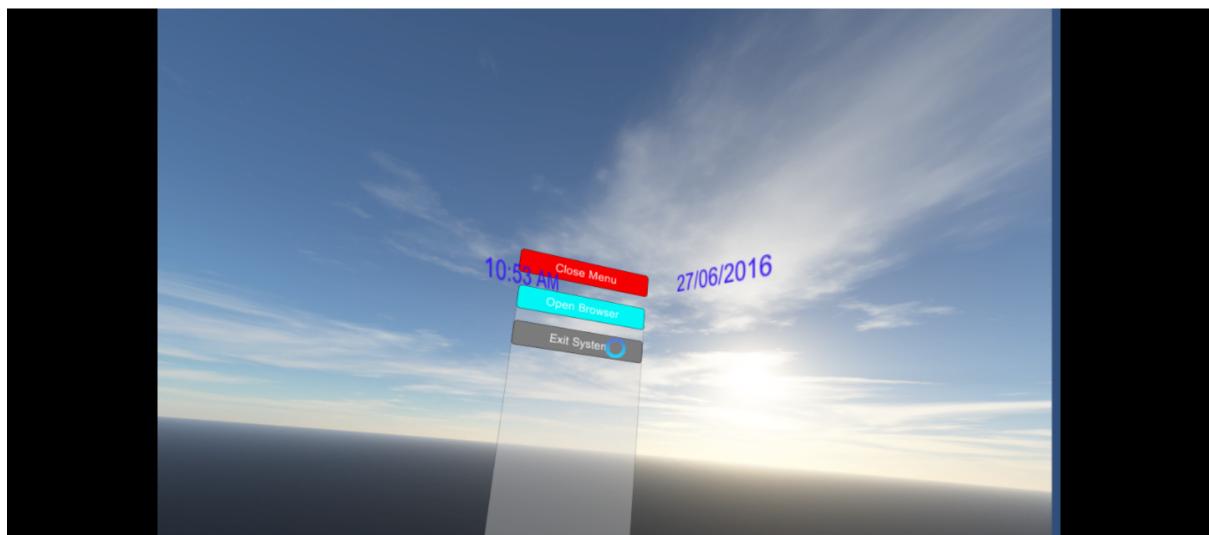


Figura A.11: Usuario seleccionado la opción para apagar el sistema

A.3. Caso 3

Recordemos este caso de uso. El usuario debe abrir el menú desplegable y seleccionar la opción del navegador. A posteriori debe introducir la URL www.youtube.es, cargar la página, seleccionar el primer vídeo que aparezca y cerrar todo el sistema.



Figura A.12: Pantalla de inicio del sistema.



Figura A.13: Usuario seleccionado la opción para abrir el menú.

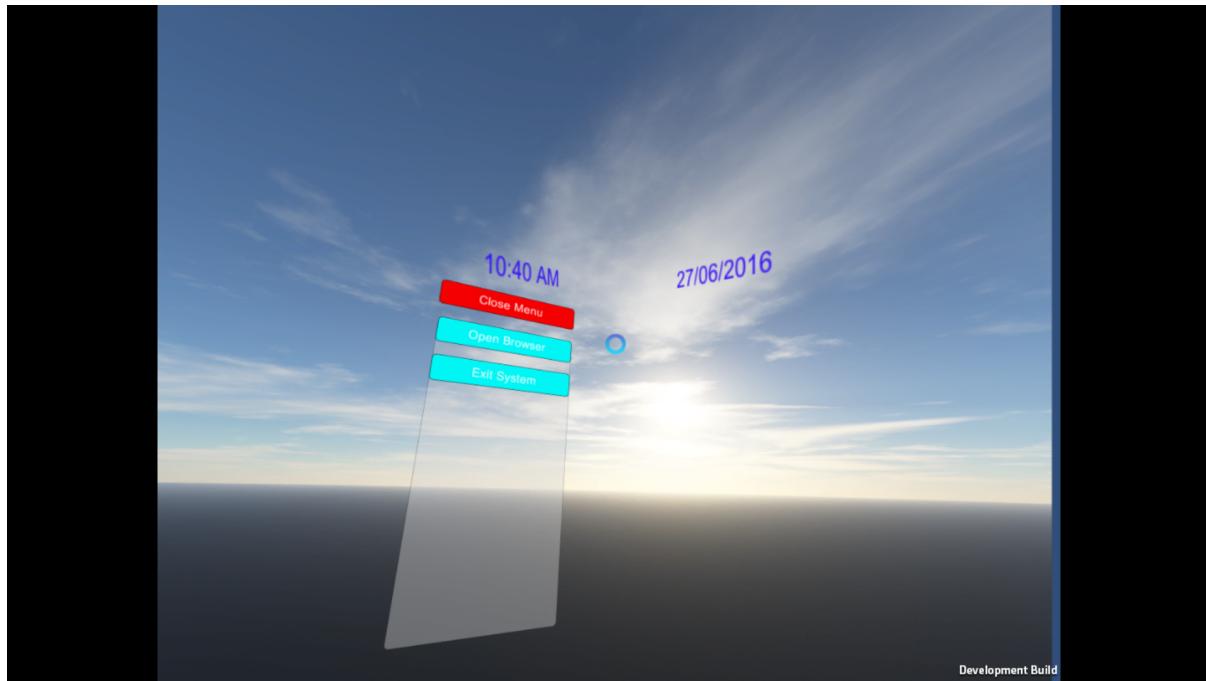


Figura A.14: Menú desplegado.



Figura A.15: Usuario seleccionando la opción para abrir el navegador.



Figura A.16: Navegador del sistema.



Figura A.17: Limpiando campo para escribir la URL.

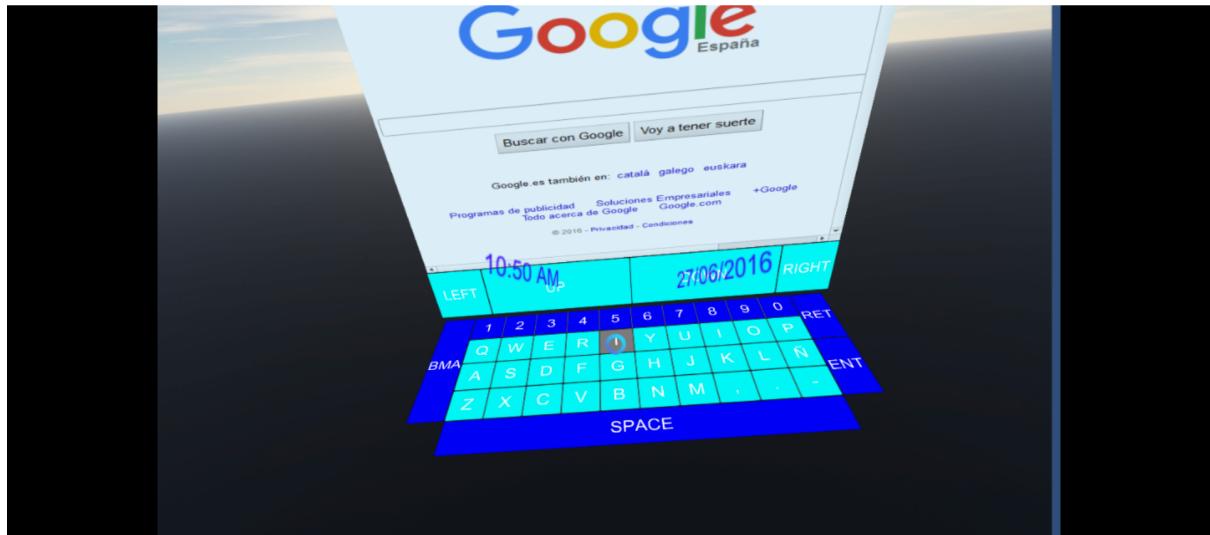


Figura A.18: Escribiendo con el teclado virtual.



Figura A.19: Cargando la URL.



Figura A.20: Visualizando un vídeo.



Figura A.21: Usuario seleccionado la opción para cerrar el navegador.



Figura A.22: Usuario seleccionando la opción para apagar el sistema



Código Fuente

En este anexo va incluido el código fuente del desarrollo de este TFG. La codificación no es lo principal en este software sino que hay detrás un trabajo de diseño 3D que encaja con todo este código. Se adjunta la URL desde la que se puede descargar todo el código y el proyecto de *Unity 3D*: https://www.dropbox.com/s/h7nz0ny43y9weu7/TFG_Cristian_Fernandez_Del_Pozo_v1_2016_Source.rar?dl=0

Los cuadros de código reciben la nomenclatura de Listing.

B.1. Menu

En esta sección se incluyen los códigos que dan funcionalidad al menú de la aplicación.

El código descrito en Listing B.1 abre y cierra el menú.

Listing B.1: MenuTrigger

```
1  using UnityEngine;
2  using System.Collections;
3  using UnityEngine.UI;
4
5  public class MenuTrigger : MonoBehaviour {
6
7      [Header("Trigger Click Button")]
8      public Button TriggerButton;
9
10     [Header("MenuPanel")]
11     public GameObject Panel;
12
13     private ColorBlock colorOn;
14     private ColorBlock colorOff;
15
16     void Start (){
```

```
17
18     colorOn = new ColorBlock ();
19     colorOn.normalColor = Color.red;
20     colorOn.highlightedColor = Color.gray;
21     colorOn.colorMultiplier = 1;
22
23     colorOff = new ColorBlock ();
24     colorOff.normalColor = Color.blue;
25     colorOff.highlightedColor = Color.gray;
26     colorOff.colorMultiplier = 1;
27 }
28
29 public void OnClickEvent(){
30     Debug.Log ("Click!");
31
32     if(Panel.GetComponent<Animator>().GetBool("isVisible")){
33         Debug.Log("Ocultate!");
34
35         var results = Panel.GetComponentsInChildren<Button>(true);
36         Panel.GetComponent<Animator>().SetBool("isVisible",false);
37         Panel.GetComponent<Animator>().SetTrigger("PlayClose");
38
39         foreach(Button button in results){
40             button.gameObject.SetActive(false);
41         }
42
43         if (TriggerButton != null) {
44             TriggerButton.colors = colorOff;
45             TriggerButton.GetComponentInChildren<Text> ().text = "←
46             Open Menu";
47
48
49
50     }else{
51         Debug.Log("Muestrate!!!");
52
53         Panel.GetComponent<Animator>().SetBool("isVisible",true);
54         Panel.GetComponent<Animator>().SetTrigger("PlayOpen");
55         var results = Panel.GetComponentsInChildren<Button>(true);
56
57         if (TriggerButton != null) {
58             TriggerButton.colors = colorOn;
59             TriggerButton.GetComponentInChildren<Text> ().text = "←
60             Close Menu";
61
62
63         foreach(Button button in results){
64             button.gameObject.SetActive(true);
65     }
```

```
66     }  
67     }  
68 }
```

El código descrito en Listing B.2 abre y cierra el navegador.

Listing B.2: BrowserTrigger

```
1  using UnityEngine;
2  using System.Collections;
3  using UnityEngine.UI;
4
5  public class BrowserTrigger : MonoBehaviour {
6
7      [Header("TriggerButton")]
8      public Button TriggerButton;
9
10     public void OnClickEvent(){
11
12         ColorBlock colorOn = new ColorBlock ();
13         colorOn.normalColor = Color.red;
14         colorOn.highlightedColor = Color.gray;
15         colorOn.colorMultiplier = 1;
16
17         ColorBlock colorOff = new ColorBlock ();
18         colorOff.normalColor = Color.cyan;
19         colorOff.highlightedColor = Color.gray;
20         colorOff.colorMultiplier = 1;
21
22         Debug.Log ("Click!");
23
24         if(gameObject.GetComponent<Animator>().GetBool("isVisible"))←
25             {
26                 Debug.Log ("Ocultate!");
27
28                 if (TriggerButton != null) {
29                     TriggerButton.colors = colorOff;
30                     TriggerButton.GetComponentInChildren<Text> ().text = "←
31                         Open Browser";
32
33                     gameObject.GetComponent<Animator>().SetBool("isVisible",←
34                         false);
35                     gameObject.GetComponent<Animator>().SetTrigger("←
36                         CloseBrowser");
37
38 //                     var results = gameObject.GetComponentsInChildren<←
39 //                     GameObject>(true);
40 //                     foreach(GameObject button in results){
41 //                         button.SetActive(false);
42 //                     }
43 //
44
45             }else{
46                 Debug.Log ("Muestrate!!!");
47
48             }
49
50         }
51
52     }
53 }
```

```
43
44     if (TriggerButton != null) {
45         TriggerButton.colors = colorOn;
46         TriggerButton.GetComponentInChildren<Text> ().text = "←
47             Close Browser";
48     }
49
50     gameObject.GetComponent<Animator>().SetBool("isVisible",←
51         true);
52     gameObject.GetComponent<Animator>().SetTrigger("←
53         OpenBrowser");
54 //     var results = gameObject.GetComponentsInChildren<←
55 //     GameObject>(true);
56 //
57 //
58 //     foreach(GameObject button in results){
59 //         button.SetActive(true);
56 //     }
57     }
58 }
59 }
```

El código descrito en Listing B.3 apaga el sistema entero.

Listing B.3: Quit

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Quit : MonoBehaviour {
5
6     public void OnClickEvent(){
7         Application.Quit ();
8     }
9 }
```

B.2. Panel de información

En esta sección se muestra el código que establece la fecha y la hora del sistema en el panel de información. Listing B.4

Listing B.4: InfoPanelController

```
1 #pragma strict
2
3 var dateText:UI.Text;
4 var timeText:UI.Text;
5
6
```

```
7 function Start () {
8 }
9 }
10
11 function Update () {
12
13
14     dateText.text = System.DateTime.Now.ToString("dd/MM/yyyy");
15     timeText.text = System.DateTime.Now.ToShortTimeString();
16
17
18 }
```

B.3. Navegador

En esta sección se describe el código necesario para la interacción con el navegador.

El Listing B.5 muestra la adaptación del código proporcionado por *uWebKit*.

Listing B.5: webTexture

```
1 ****
2 * uWebKit
3 * (c) 2014 THUNDERBEAST GAMES , LLC
4 * http://www.uwebkit.com
5 * sales@uwebkit.com
6 ****
7 ****
8 * Mod by Cristian Fernández for TFG project
9 * Under University License
10 * cristian.fernandez@estudiante.uam.es
11 * 2016
12 ****
13 ****
14
15 using UnityEngine;
16 using System.Collections;
17 using System.Collections.Generic;
18 using UnityEngine.Events;
19 using UnityEngine.UI;
20
21 /// <summary>
22 /// Basic example of using a UWKWebView on a 3D Unity surface
23 /// </summary>
24
25 // IMPORTANT: Please see the WebGUI.cs example for 2D support
26
27 public class WebTexture : MonoBehaviour, IPointerExitHandler, ←
    IPointerEnterHandler
28 {
```

```
29  
30 #region Inspector Fields  
31  
32 [Header ("Buttons")]  
33 public Button BlockedClicksText;  
34  
35 [Header ("InputFields")]  
36 public InputField URLInput;  
37  
38 [Header ("CrossHair Animator")]  
39 public Animator CrossHairAnimator;  
40  
41 [Header ("Web Options")]  
42 public bool KeyboardEnabled = true;  
43 public bool MouseEnabled = true;  
44 public bool Rotate = false;  
45 public bool HasFocus = true;  
46 public bool AlphaMask = false;  
47  
48 #endregion  
49  
50 private bool _isInit = false;  
51  
52 private Vector3 _previousMousePosition = Vector3.zero;  
53 private Vector3 _previous2MousePosition = Vector3.zero;  
54 private Vector3 _previous3MousePosition = Vector3.zero;  
55  
56 private bool _isBlocked = false;  
57  
58 private bool _hasClicked = false;  
59  
60 UWKWebView view;  
61  
62 Camera _camera = null;  
63 private int _counter = 0;  
64 private int _ClickCounter = 0;  
65 private int _X = 0;  
66 private int _Y = 0;  
67 private int _posX = 0;  
68 private int _posY = 0;  
69  
70  
71 // Use this for initialization  
72 void Start ()  
73 {  
74  
75     view = gameObject.GetComponent<UWKWebView> ();  
76  
77     _camera = Camera.main;  
78  
79     view.SetAlphaMask (AlphaMask);
```

```
80     view.SetFrameRate (60);
81
82     if (URLInput != null) {
83         URLInput.text = view.URL;
84
85         view.URLChanged = new URLChangedDelegate ((v,url)=>{
86
87             URLInput.text = url;
88
89         });
90     }
91
92
93
94 #if !UNITY_4_6
95     if (GetComponent<Renderer> () != null)
96         GetComponent<Renderer> ().material.mainTexture = view.←
97             WebTexture;
98
99     if (GetComponent<GUITexture> () != null)
100        GetComponent<GUITexture> ().texture = view.WebTexture;
101 #else
102     if (renderer != null)
103         renderer.material.mainTexture = view.WebTexture;
104
105     if (guiTexture != null)
106         guiTexture.texture = view.WebTexture;
107 #endif
108
109     Cursor.lockState = CursorLockMode.Locked;
110     _counter = 0;
111     _ClickCounter = 0;
112 }
113
114 // Update is called once per frame
115 void Update ()
116 {
117     if (Rotate)
118         gameObject.transform.Rotate (0, Time.deltaTime * 4.0f, 0);
119
120     if (!HasFocus || !_isInit)
121         return;
122     double tolx = view.CurrentWidth * 0.02;
123     double toly = view.CurrentHeight * 0.02;
124     RaycastHit rcast;
125
126     Ray r2 = _camera.ScreenPointToRay (Input.mousePosition);
127     Debug.DrawRay (r2.origin, r2.direction, Color.red);
128
129     if (_counter % 61 == 0) {
```

```

130
131     _counter = 0;
132
133     if (Physics.Raycast (r2, out rcast)) {
134         if (rcast.collider != GetComponent<MeshCollider> ()) {
135             return;
136         }
137
138         int x = (int)(rcast.textureCoord.x * (float)view.←
139                         MaxWidth);
140         int y = view.MaxHeight - (int)(rcast.textureCoord.y * (←
141                         float)view.MaxHeight);
142
143         Vector3 mousePos = new Vector3 ();
144         mousePos.x = x;
145         mousePos.y = y;
146
147         view.ProcessMouse (mousePos);
148
149         if (!Utils.Compare2Vector3WithTolerance (←
150             _previous.mousePosition, mousePos, tolx, toly)) {
151
152             if (_ClickCounter < 50) {
153                 _ClickCounter++;
154             } else {
155                 _ClickCounter = 0;
156                 if (!_isBlocked && !_hasClicked) {
157                     _hasClicked = true;
158                     CrossHairAnimator.SetTrigger ("Play");
159                     MouseOperations.MouseEvent (MouseOperations.←
160                         MouseEventFlags.LeftUp | MouseOperations.←
161                         MouseEventFlags.LeftDown);
162                     Debug.Log ("Click WEBVIEW");
163                 } else {
164                     _hasClicked = false;
165                 }
166             }
167
168         } else {
169             _counter++;
170         }
171     }
172
173     public void LoadURL(){
174
175         if (view != null && URLInput != null) {

```

```
176
177     string url = Utils.CheckURL (URLInput.text);
178
179     view.LoadURL (url);
180
181
182 }
183
184
185 }
186
187
188 public void ScrollUP(){
189     if (view != null && _Y != 0) {
190         _Y -= 1;
191         _posY = (int)(_Y * (view.ContentHeight * 0.05));
192         view.SetScrollPosition (_posX, _posY);
193     }
194 }
195
196 public void ScrollDown(){
197     if (view != null) {
198
199         _Y +=1;
200
201         _posY = (int)(_Y * (view.ContentHeight * 0.05));
202
203
204         view.SetScrollPosition (_posX, _posY);
205     }
206 }
207
208 public void ScrollLeft(){
209     if (view != null && _X != 0) {
210         _X -= 1;
211         _posX = (int)(_Y * (view.ContentWidth * 0.05));
212         view.SetScrollPosition (_posX, _posY);
213     }
214 }
215
216 public void ScrollRight(){
217     if (view != null) {
218
219         _X +=1;
220
221         _posX = (int)(_X * (view.ContentWidth * 0.05));
222
223
224         view.SetScrollPosition (_posX, _posY);
225     }
226 }
```

```
227
228     void OnGUI ()
229     {
230         if (!KeyboardEnabled || !HasFocus)
231             return;
232
233         if (Event.current.isKey) {
234             view.ProcessKeyboard (Event.current);
235         }
236
237     }
238
239 #region IPointerEnterHandler implementation
240
241     public void OnPointerEnter (PointerEventData eventData)
242     {
243         _isInit = true;
244         _hasClicked = false;
245
246     }
247
248 #endregion
249
250
251
252
253 #region IPointerExitHandler implementation
254
255     public void OnPointerExit (PointerEventData eventData)
256     {
257         _isInit = false;
258         _hasClicked = false;
259     }
260
261 #endregion
262
263     public void OnBlockedEvent ()
264     {
265         _isBlocked = !_isBlocked;
266
267         ColorBlock color = new ColorBlock ();
268
269         if (_isBlocked) {
270             BlockedClicksText.GetComponentInChildren<Text> ().text = "←
271             UNLOCK";
272             color.normalColor = Color.red;
273             color.highlightedColor = Color.gray;
274         } else {
275             BlockedClicksText.GetComponentInChildren<Text> ().text = "←
276             LOCK";
277             color.normalColor = Color.cyan;
```

```

276         color.highlightedColor = Color.gray;
277     }
278     color.colorMultiplier = 1;
279     BlockedClicksText.colors = color;
280
281 }
282
283 }
```

El código mostrado en el Listing B.6 corresponde a las funcionalidades del teclado y el campo de introducción de la URL.

Listing B.6: urlController

```

1  using UnityEngine;
2  using System.Collections;
3  using UnityEngine.UI;
4
5  public class URLController : MonoBehaviour {
6
7      private InputField _input;
8
9      // Use this for initialization
10     void Start () {
11         _input = GetComponent<InputField> ();
12     }
13
14     // Update is called once per frame
15     void Update () {
16
17     }
18
19     public void Clear(){
20         _input.text = "";
21     }
22
23     void OnGUI()
24     {
25         if (_input.isFocused) {
26
27             Event e = Event.current;
28
29             if (Input.anyKeyDown && e.isKey) {
30                 Debug.Log ("Detected key code: " + e.keyCode);
31                 if (_input != null && e.keyCode != KeyCode.None) {
32                     if (e.keyCode == KeyCode.Backspace) {
33                         _input.text = _input.text.Substring (0, _input.text.Length - 1);
34                     } else if (e.keyCode == KeyCode.Period) {
35                         _input.text += ".";
36                     } else if (e.keyCode == KeyCode.BackQuote) {
```

```
37         _input.text += "ñ";
38     }else if (e.keyCode == KeyCode.KeypadDivide) {
39         _input.text += "/";
40     } else {
41         _input.text += e.keyCode.ToString ().ToLower ();
42     }
43
44 }
45
46 }
47 }
48
49 }
50 }
```

B.4. Teclado Virtual

El código mostrado en esta sección corresponde a las funcionalidades propias del teclado virtual de la aplicación. Listing B.7.

Listing B.7: Virtual KeyboardController

```
1 using System;
2 using UnityEngine;
3 using UnityEngine.UI;
4 using WindowsInput;
5
6 public class KeyboardController : MonoBehaviour
{
7
8     [Header("Input Text")]
9     public InputField Input;
10
11    [Header("Controller Button")]
12    public Button ClearController;
13    public Button GoController;
14
15    [Header("Letter Buttons")]
16    public Button A;
17    public Button B;
18    public Button C;
19    public Button D;
20    public Button E;
21    public Button F;
22    public Button G;
23    public Button H;
24    public Button I;
25    public Button J;
26    public Button K;
27    public Button L;
28    public Button M;
```

```
29     public Button N;
30     public Button Ñ;
31     public Button O;
32     public Button P;
33     public Button Q;
34     public Button R;
35     public Button S;
36     public Button T;
37     public Button U;
38     public Button V;
39     public Button W;
40     public Button X;
41     public Button Y;
42     public Button Z;
43
44 [Header("Number Buttons")]
45     public Button B1;
46     public Button B2;
47     public Button B3;
48     public Button B4;
49     public Button B5;
50     public Button B6;
51     public Button B7;
52     public Button B8;
53     public Button B9;
54     public Button B0;
55
56 [Header("Simbol Buttons")]
57     public Button Dot;
58     public Button Comma;
59     public Button Guion;
60
61 [Header("Command Buttons")]
62     public Button Mayus;
63     public Button Ret;
64     public Button Enter;
65     public Button Space;
66
67     private bool _urlWrite = false;
68
69     void Start ()
70     {
71         if (ClearController != null) {
72             ClearController.onClick.AddListener (() => {
73                 _urlWrite = true;
74             });
75         }
76
77         if (GoController != null) {
78             GoController.onClick.AddListener (()=>{
79                 _urlWrite = false;
```

```
80     });
81 }
82
83 if (A != null) {
84     A.setOnClickListener ((() => {
85
86         if (Input != null && _urlWrite){
87             Input.text+="a";
88         }else{
89             KeyBoardOperation.PressKey(VirtualKeyCode.VK_A);
90
91         }
92
93     });
94 }
95
96
97
98
99 if (B != null) {
100    B.setOnClickListener ((() => {
101
102        if (Input != null && _urlWrite){
103            Input.text+="b";
104        }else{
105            KeyBoardOperation.PressKey(VirtualKeyCode.VK_B);
106
107        }
108    });
109 }
110
111 if (C != null) {
112     C.setOnClickListener ((() => {
113
114         if (Input != null && _urlWrite){
115             Input.text+="c";
116         }else{
117             KeyBoardOperation.PressKey(VirtualKeyCode.VK_C);
118
119         }
120     });
121 }
122
123 if (D != null) {
124     D.setOnClickListener ((() => {
125
126         if (Input != null && _urlWrite){
127             Input.text+="d";
128         }else{
129             KeyBoardOperation.PressKey(VirtualKeyCode.VK_D);
130
131     });
132 }
```

```
131     }
132
133     });
134 }
135
136 if (E != null) {
137     E.setOnClickListener (() => {
138
139         if (Input != null&& _urlWrite){
140             Input.text+="e";
141         }else{
142             KeyBoardOperation.PressKey(VirtualKeyCode.VK_E);
143
144         }
145     });
146 }
147
148 if (F != null) {
149     F.setOnClickListener (() => {
150
151         if (Input != null&& _urlWrite){
152             Input.text+="f";
153         }else{
154             KeyBoardOperation.PressKey(VirtualKeyCode.VK_F);
155
156         }
157     });
158 }
159
160 if (G != null) {
161     G.setOnClickListener (() => {
162
163         if (Input != null&& _urlWrite){
164             Input.text+="g";
165         }else{
166             KeyBoardOperation.PressKey(VirtualKeyCode.VK_G);
167
168         }
169     });
170
171 if (H != null) {
172     H.setOnClickListener (() => {
173
174         if (Input != null&& _urlWrite){
175             Input.text+="h";
176         }else{
177             KeyBoardOperation.PressKey(VirtualKeyCode.VK_H);
178
179         }
180     });
181 }
```

```
182     if (I != null) {
183         I.onClick.AddListener (() => {
184
185             if (Input != null&& _urlWrite){
186                 Input.text+="i";
187             }else{
188                 KeyBoardOperation.PressKey(VirtualKeyCode.VK_I);
189             }
190
191         });
192     }
193
194     if (J != null) {
195         J.onClick.AddListener (() => {
196
197             if (Input != null&& _urlWrite){
198                 Input.text+="J";
199             }else{
200                 KeyBoardOperation.PressKey(VirtualKeyCode.VK_J);
201             }
202         });
203     }
204
205     if (K != null) {
206         K.onClick.AddListener (() => {
207
208             if (Input != null&& _urlWrite){
209                 Input.text+="K";
210             }else{
211                 KeyBoardOperation.PressKey(VirtualKeyCode.VK_K);
212             }
213
214         });
215     }
216
217     if (L != null) {
218         L.onClick.AddListener (() => {
219
220             if (Input != null&& _urlWrite){
221                 Input.text+="L";
222             }else{
223                 KeyBoardOperation.PressKey(VirtualKeyCode.VK_L);
224             }
225         });
226     }
227
228     if (M != null) {
229         M.onClick.AddListener (() => {
230
231             if (Input != null&& _urlWrite){
232                 Input.text+="m";
```

```
233     }  
234         KeyBoardOperation.PressKey(VirtualKeyCode.VK_M);  
235     }  
236  
237     });  
238 }  
239  
240 if (N != null) {  
241     N.setOnClickListener (() => {  
242  
243         if (Input != null && _urlWrite){  
244             Input.text+="  
245         }  
246         KeyBoardOperation.PressKey(VirtualKeyCode.VK_N);  
247     }  
248  
249     });  
250 }  
251  
252 if (Ñ != null) {  
253     Ñ.setOnClickListener (() => {  
254  
255         if (Input != null && _urlWrite){  
256             Input.text+="  
257         }  
258         KeyBoardOperation.PressSpecialKey("ñ");  
259     }  
260  
261     });  
262 }  
263  
264 if (O != null) {  
265     O.setOnClickListener (() => {  
266  
267         if (Input != null && _urlWrite){  
268             Input.text+="  
269         }  
270         KeyBoardOperation.PressKey(VirtualKeyCode.VK_O);  
271     }  
272  
273     });  
274 }  
275  
276 if (P != null) {  
277     P.setOnClickListener (() => {  
278  
279         if (Input != null && _urlWrite){  
280             Input.text+="  
281         }  
282         KeyBoardOperation.PressKey(VirtualKeyCode.VK_P);  
283     }  
284 }
```

```
284         });
285     }
286 }
287
288 if (Q != null) {
289     Q.setOnClickListener ((() => {
290
291         if (Input != null && _urlWrite){
292             Input.text+="q";
293         }else{
294             KeyBoardOperation.PressKey(VirtualKeyCode.VK_Q);
295         }
296
297     });
298 }
299
300 if (R != null) {
301     R.setOnClickListener ((() => {
302
303         if (Input != null && _urlWrite){
304             Input.text+="r";
305         }else{
306             KeyBoardOperation.PressKey(VirtualKeyCode.VK_R);
307         }
308
309     });
310 }
311
312 if (S != null) {
313     S.setOnClickListener ((() => {
314
315         if (Input != null && _urlWrite){
316             Input.text+="s";
317         }else{
318             KeyBoardOperation.PressKey(VirtualKeyCode.VK_S);
319         }
320
321     });
322 }
323
324 if (T != null) {
325     T.setOnClickListener ((() => {
326
327         if (Input != null && _urlWrite){
328             Input.text+="t";
329         }else{
330             KeyBoardOperation.PressKey(VirtualKeyCode.VK_T);
331         }
332
333     });
334 }
```

```
335
336     if (U != null) {
337         U.onClick.AddListener (() => {
338
339             if (Input != null && _urlWrite){
340                 Input.text+="u";
341             }else{
342                 KeyBoardOperation.PressKey(VirtualKeyCode.VK_U);
343             }
344
345         });
346     }
347
348     if (V != null) {
349         V.onClick.AddListener (() => {
350
351             if (Input != null && _urlWrite){
352                 Input.text+="v";
353             }else{
354                 KeyBoardOperation.PressKey(VirtualKeyCode.VK_V);
355             }
356
357         });
358     }
359
360
361     if (W != null) {
362         W.onClick.AddListener (() => {
363
364             if (Input != null && _urlWrite){
365                 Input.text+="w";
366             }else{
367                 KeyBoardOperation.PressKey(VirtualKeyCode.VK_W);
368             }
369
370         });
371     }
372
373     if (X != null) {
374         X.onClick.AddListener (() => {
375
376             if (Input != null && _urlWrite){
377                 Input.text+="x";
378             }else{
379                 KeyBoardOperation.PressKey(VirtualKeyCode.VK_X);
380             }
381
382         });
383     }
384
385     });
386 }
```

```
386     }
387
388     if (Y != null) {
389         Y.setOnClickListener (() => {
390
391             if (Input != null && _urlWrite){
392                 Input.text+="Y";
393             }else{
394                 KeyBoardOperation.PressKey(VirtualKeyCode.VK_Y);
395             }
396
397         });
398     }
399 }
400
401 if (Z != null) {
402     Z.setOnClickListener (() => {
403
404         if (Input != null && _urlWrite){
405             Input.text+="z";
406         }else{
407             KeyBoardOperation.PressKey(VirtualKeyCode.VK_Z);
408         }
409
410     });
411 }
412
413 if (B0 != null) {
414     B0.setOnClickListener (() => {
415
416         if (Input != null && _urlWrite){
417             Input.text+="0";
418         }else{
419             KeyBoardOperation.PressKey(VirtualKeyCode.VK_0);
420         }
421
422     });
423 }
424
425 }
426
427 if (B1 != null) {
428     B1.setOnClickListener (() => {
429
430         if (Input != null && _urlWrite){
431             Input.text+="1";
432         }else{
433             KeyBoardOperation.PressKey(VirtualKeyCode.VK_1);
434         }
435
436 }
```

```
437     });
438 }
439
440 if (B2 != null) {
441     B2.onClick.AddListener (() => {
442
443         if (Input != null&& _urlWrite){
444             Input.text+="2";
445         }else{
446             KeyBoardOperation.PressKey(VirtualKeyCode.VK_2);
447         }
448
449     });
450 }
451
452 if (B3 != null) {
453     B3.onClick.AddListener (() => {
454
455         if (Input != null&& _urlWrite){
456             Input.text+="3";
457         }else{
458             KeyBoardOperation.PressKey(VirtualKeyCode.VK_3);
459         }
460
461     });
462 }
463
464 if (B4 != null) {
465     B4.onClick.AddListener (() => {
466
467         if (Input != null&& _urlWrite){
468             Input.text+="4";
469         }else{
470             KeyBoardOperation.PressKey(VirtualKeyCode.VK_4);
471         }
472
473     });
474 }
475
476 }
477
478 if (B5 != null) {
479     B5.onClick.AddListener (() => {
480
481         if (Input != null&& _urlWrite){
482             Input.text+="5";
483         }else{
484             KeyBoardOperation.PressKey(VirtualKeyCode.VK_5);
485         }
486
487 }
```

```
488     });
489 }
490
491 if (B6 != null) {
492     B6.onClick.AddListener (() => {
493
494         if (Input != null&& _urlWrite){
495             Input.text+="6";
496         }else{
497             KeyBoardOperation.PressKey(VirtualKeyCode.VK_6);
498         }
499
500     });
501 }
502
503
504 if (B7 != null) {
505     B7.onClick.AddListener (() => {
506
507         if (Input != null&& _urlWrite){
508             Input.text+="7";
509         }else{
510             KeyBoardOperation.PressKey(VirtualKeyCode.VK_7);
511         }
512
513     });
514 }
515
516
517 if (B8 != null) {
518     B8.onClick.AddListener (() => {
519
520         if (Input != null&& _urlWrite){
521             Input.text+="8";
522         }else{
523             KeyBoardOperation.PressKey(VirtualKeyCode.VK_8);
524         }
525
526     });
527 }
528
529 if (B9 != null) {
530     B9.onClick.AddListener (() => {
531
532         if (Input != null&& _urlWrite){
533             Input.text+="9";
534         }else{
535             KeyBoardOperation.PressKey(VirtualKeyCode.VK_9);
536         }
537
538 }
```

```
539     });
540 }
541
542 if (Dot != null) {
543     Dot.onClick.AddListener (() => {
544
545         if (Input != null&& _urlWrite){
546             Input.text+=". ";
547         }else{
548             KeyBoardOperation.PressSpecialKey(".");
549         }
550
551     });
552 }
553
554
555 if (Comma != null) {
556     Comma.onClick.AddListener (() => {
557
558         if (Input != null&& _urlWrite){
559             Input.text+="," ;
560         }else{
561             KeyBoardOperation.PressSpecialKey(",");
562         }
563
564     });
565 }
566
567
568 if (Guion != null) {
569     Guion.onClick.AddListener (() => {
570
571         if (Input != null&& _urlWrite){
572             Input.text+=" - ";
573         }else{
574             KeyBoardOperation.PressSpecialKey("- ");
575         }
576
577     });
578 }
579
580
581 if (Enter != null) {
582     Enter.onClick.AddListener (() => {
583
584         KeyBoardOperation.PressKey(VirtualKeyCode.RETURN);
585
586     });
587 }
588 }
589
```

```
590     if (Ret != null) {
591         Ret.onClick.AddListener (() => {
592             if (Input != null && _urlWrite){
593                 Input.text = Input.text.Substring(0, Input.text.Length - 1);
594             } else{
595                 KeyBoardOperation.PressKey(VirtualKeyCode.BACK);
596             }
597
598         });
599     }
600 }
601
602
603     if (Space != null) {
604         Space.onClick.AddListener (() => {
605
606             KeyBoardOperation.PressKey(VirtualKeyCode.SPACE);
607
608         });
609     }
610
611     if (Mayus != null) {
612
613         ColorBlock colorOn = new ColorBlock ();
614         colorOn.normalColor = Color.red;
615         colorOn.highlightedColor = Color.gray;
616         colorOn.colorMultiplier = 1;
617
618         ColorBlock colorOff = new ColorBlock ();
619         colorOff.normalColor = Color.blue;
620         colorOff.highlightedColor = Color.gray;
621         colorOff.colorMultiplier = 1;
622
623         Mayus.colors = (KeyBoardOperation.IsCapitalLetter()) ? colorOn : colorOff;
624
625         Mayus.onClick.AddListener (() => {
626
627             var prev = KeyBoardOperation.IsCapitalLetter();
628             prev = !prev;
629
630             KeyBoardOperation.PressKey(VirtualKeyCode.CAPITAL);
631
632             Mayus.colors = (prev) ? colorOn : colorOff;
633
634         });
635     }
636
637
638 }
```

639

}

B.5. Sistema de interacción Gaze

En esta sección se muestra el código necesario para adaptar el sistema Gaze Input al trabajo realizado. Listing B.8.

Listing B.8: GazeInput

```

1 // Gaze Input Module by Peter Koch <petercpt@gmail.com>
2
3 ****
4 * Mod by Cristian Fernández for TFG project
5 * Under University License
6 * cristian.fernandez@estudiante.uam.es
7 * 2016
8 ****
9
10 using UnityEngine;
11 using UnityEngine.Events;
12 using System.Collections.Generic;
13
14 // To use:
15 // 1. Drag onto your EventSystem game object.
16 // 2. Disable any other Input Modules (eg: StandaloneInputModule ←
17 //     & TouchInputModule) as they will fight over selections.
18 // 3. Make sure your Canvas is in world space and has a ←
19 //     GraphicRaycaster (should be default).
20 // 4. If you have multiple cameras then make sure to drag your ←
21 //     VR (center eye) camera into the canvas.
22 public class GazeInputModule : PointerInputModule
23 {
24     public enum Mode { Click = 0, Gaze };
25     public Mode mode;
26
27     [Header("Click Settings")]
28     public string ClickInputName = "Submit";
29     [Header("Gaze Settings")]
30     public float GazeTimeInSeconds = 2f;
31
32     [Header("CrossHair Animator")]
33     public Animator CrossHairAnimator;
34
35     public RaycastResult CurrentRaycast;
36
37     private PointerEventData pointerEventData;
38     private GameObject currentLookAtHandler;
39     private float currentLookAtHandlerClickTime;

```

```

38     public override void Process()
39     {
40         HandleLook();
41         HandleSelection();
42     }
43
44     void HandleLook()
45     {
46         if (pointerEventData == null)
47         {
48             pointerEventData = new PointerEventData(eventSystem);
49         }
50         // fake a pointer always being at the center of the screen
51         pointerEventData.position = new Vector2(Screen.width/2, ←
52             Screen.height/2);
53         pointerEventData.delta = Vector2.zero;
54         List<RaycastResult> raycastResults = new List<RaycastResult>←
55             >();
56         eventSystem.RaycastAll(pointerEventData, raycastResults);
57         CurrentRaycast = pointerEventData.pointerCurrentRaycast = ←
58             FindFirstRaycast(raycastResults);
59         ProcessMove(pointerEventData);
60     }
61
62     void HandleSelection()
63     {
64         if (pointerEventData.pointerEnter != null)
65         {
66             // if the ui receiver has changed, reset the gaze delay ←
67             // timer
68             GameObject handler = ExecuteEvents.GetEventHandler<←
69                 IPointerClickHandler>(pointerEventData.pointerEnter);
70             if (currentLookAtHandler != handler)
71             {
72                 currentLookAtHandler = handler;
73
74                 currentLookAtHandlerClickTime = Time.←
75                     realtimeSinceStartup + GazeTimeInSeconds;
76
77                 CrossHairAnimator.SetTrigger("Play");
78             }
79             // if we have a handler and it's time to click, do it now
80             if (currentLookAtHandler != null &&
81                 (mode == Mode.Gaze && Time.realtimeSinceStartup > ←
82                     currentLookAtHandlerClickTime) ||
83                 (mode == Mode.Click && Input.GetButtonDown(←
84                     ClickInputName)))
85             {
86                 if (EventSystem.current.currentSelectedGameObject != ←

```

```
        null)
81    {
82        Debug.Log ("old" + EventSystem.current.←
83            currentSelectedGameObject.name);
84        GameObject currentSelected = EventSystem.current.←
85            currentSelectedGameObject;
86
87        // ExecuteEvents.ExecuteHierarchy(EventSystem.current.←
88        // currentSelectedGameObject, pointerEventData, ←
89        // ExecuteEvents.deselectHandler);
90        }
91
92        EventSystem.current.SetSelectedGameObject(←
93            currentLookAtHandler);
94
95        ExecuteEvents.ExecuteHierarchy(currentLookAtHandler, ←
96            pointerEventData, ExecuteEvents.pointerClickHandler);
97        currentLookAtHandlerClickTime = float.MaxValue;
98        ExecuteEvents.ExecuteHierarchy(EventSystem.current.←
99            .currentSelectedGameObject, pointerEventData, ←
100           ExecuteEvents.deselectHandler);
101    }
102
103
104 }
```

B.6. Sistema de emulación de clicks y teclado

En esta sección se muestra el código para emular clicks y eventos de teclado.

En el Listing B.9 se muestra el código para la emulación del ratón.

Listing B.9: MouseOperation

```
1 using System;
2 using System.Runtime.InteropServices;
3
4 public class MouseOperations
5 {
6     [Flags]
7     public enum MouseEventFlags
8     {
```

```
9     LeftDown = 0x00000002 ,
10    LeftUp = 0x00000004 ,
11    MiddleDown = 0x00000020 ,
12    MiddleUp = 0x00000040 ,
13    Move = 0x00000001 ,
14    Absolute = 0x00008000 ,
15    RightDown = 0x00000008 ,
16    RightUp = 0x00000010
17 }
18
19 [DllImport("user32.dll", EntryPoint = "SetCursorPos")]
20 [return: MarshalAs(UnmanagedType.Bool)]
21 private static extern bool SetCursorPos(int X, int Y);
22
23 [DllImport("user32.dll")]
24 [return: MarshalAs(UnmanagedType.Bool)]
25 private static extern bool GetCursorPos(out MousePoint ←
26     lpMousePoint);
27
28 [DllImport("user32.dll")]
29 private static extern void mouse_event(int dwFlags, int dx, ←
30     int dy, int dwData, int dwExtraInfo);
31
32 public static void SetCursorPosition(int X, int Y)
33 {
34     SetCursorPos(X, Y);
35 }
36
37 public static void SetCursorPosition(MousePoint point)
38 {
39     SetCursorPos(point.X, point.Y);
40 }
41
42 public static MousePoint GetCursorPosition()
43 {
44     MousePoint currentMousePoint;
45     var gotPoint = GetCursorPos(out currentMousePoint);
46     if (!gotPoint) { currentMousePoint = new MousePoint(0, ←
47         0); }
48     return currentMousePoint;
49 }
50
51
52 public static void MouseEvent(MouseEventFlags value)
53 {
54     MousePoint position = GetCursorPosition();
55
56     mouse_event
57         ((int)value,
58          position.X,
59          position.Y,
60          0,
```

```
57             0)
58         ;
59     }
60
61     [StructLayout(LayoutKind.Sequential)]
62     public struct MousePoint
63     {
64         public int X;
65         public int Y;
66
67         public MousePoint(int x, int y)
68         {
69             X = x;
70             Y = y;
71         }
72     }
73 }
74
75 }
```

En el Listing B.10 se muestra el código para la emulación del teclado.

Listing B.10: KeyBoardOperation

```
1  using System;
2  using System.Runtime.InteropServices;
3  using WindowsInput;
4
5  public class KeyBoardOperation
6  {
7      public static void PressKey(VirtualKeyCode KeyCode)
8      {
9          InputSimulator.SimulateKeyPress (KeyCode);
10     }
11
12     public static bool IsCapitalLetter(){
13         return InputSimulator.IsTogglingKeyInEffect (VirtualKeyCode.CAPITAL);
14     }
15
16     public static void PressSpecialKey(string s){
17         if (InputSimulator.IsTogglingKeyInEffect (VirtualKeyCode.CAPITAL)) {
18             s = s.ToUpper ();
19         } else {
20             s = s.ToLower ();
21         }
22
23         InputSimulator.SimulateTextEntry (s);
24     }
25 }
```

B.7. Utilidades

En esta sección se describe la clase utilidades con funciones extra de ayuda. Listing B.11.

Listing B.11: Utils

```

1  using System;
2  using UnityEngine;
3
4
5  public class Utils
6  {
7      public static double EuclideanDistance (double a, double b)
8      {
9
10         var res = Math.Sqrt (Math.Pow (a, 2) - Math.Pow (b, 2));
11
12         return (res != double.NaN) ? res : 0;
13     }
14
15
16     public static bool Compare2Vector3WithTolerance (Vector3 v1, ←
17         Vector3 v2, double Tolerance2VectorsX, double ←
18         Tolerance2VectorsY)
19     {
20
21         if (v1 == Vector3.zero || v2 == Vector3.zero) {
22             return false;
23         }
24
25
26
27         var xEdist = EuclideanDistance (v1.x, v2.x);
28         var yEdist = EuclideanDistance (v1.y, v2.y);
29
30
31         if (xEdist >= Tolerance2VectorsX || yEdist >= ←
32             Tolerance2VectorsY) {
33             return false;
34         }
35
36         public static string CheckURL(string s){
37
38             s = s.ToLower ();
39
40             string result="";
41
42             if (!s.Contains ("http://") || !s.Contains ("https://")) {

```

```
43         result = "http://" ;
44     }
45
46     if (!s.Contains ("www.")) {
47         result += "www." ;
48     }
49
50     result += s ;
51
52
53
54
55     return result ;
56
57 }
58
59 }
```
