Trabajo Final de Grado

Grado en Ingeniería en Tecnologías Industriales

Reconocimiento de voz para interacciones con aplicación en la Realidad Virtual aplicadas al Alzheimer

MEMÓRIA

Autor: Laura Berecibar Gorostiza
Director: Toni Susin
Convocatòria: 07/2018

ETSEIB

Escuela Técnica Superior
Ingeniería Industrial de Barcelona

UPC

ETSEIB

Resumen

Aunque la realidad virtual surgió en el ámbito de los videojuegos y el entretenimiento en general, actualmente y debido a sus infinitas posibilidades, la realidad virtual está en auge en campos y sectores de lo más variados. Entre estos sectores se encuentran la educación, el turismo y museografía, la medicina y el entretenimiento.

Al mismo tiempo, con más de 45 millones de afectados mundialmente y una proyección de enfermos de 130 millones para 2050, el Alzheimer es una de las enfermedades más duras de la sociedad moderna, tanto para el paciente como para la familia. Es por ello que, después de décadas de investigación en busca de su cura, se ha decidido focalizar a los investigadores en prevenirlo.

Centrado en los dos conceptos anteriores, este trabajo, “Reconocimiento de voz para interacciones con aplicación en la Realidad Virtual aplicadas al Alzheimer” tiene como objetivo entretener y ayudar en el ámbito del Alzheimer. No solo se ha creado un juego, que sirve para aumentar la reserva cognitiva del usuario, y así reducir las posibilidades de padecer Alzheimer, sino que se ha creado un sistema capaz de interactuar mediante la voz con la aplicación.

Este tipo de juego se denomina “Serious Game” ya que su propósito se basa en la formación por encima del entretenimiento, y a pesar de ser un término relativamente nuevo, ya se han visto sus efectos en formación o tratamiento de pacientes.

Este proyecto se divide en dos grandes bloques. El primer bloque se basa en la utilización de Unity y la investigación y desarrollo de una tecnología de reconocimiento de voz, innovadora en la UPC. El segundo bloque consiste en el desarrollo del concepto del Alzheimer y en la creación de una aplicación que ejercite la memoria y ayude en la prevención del Alzheimer.

Cabe remarcar que, por el corto periodo de este proyecto, 4 meses, el objetivo del proyecto no era la creación de un juego extremadamente complejo y listo para comercializar, sino de crear algo funcional con los objetivos que se habían fijado desde un principio.

Índice

ETSEIB

1. Glosario

CRV: Centro de Realidad Virtual, en la facultad de Matemáticas donde se ha desarrollado el proyecto.

RV: Realidad Virtual

IA: Inteligencia Artificial.

Asset: modelos, texturas, objetos, sonidos, cualquier contenido que compone un juego en Unity3D.

Escena: Lugar virtual donde se hace la creación del proyecto. Conforma la pantalla principal de la plataforma Unity.

Prefab: Es un tipo de asset que le permite almacenar un objeto GameObject

completamente con componentes y propiedades. Cualquier edición hecha a un

prefab será inmediatamente reflejado en todas las instancias producidas

por él.

Frame-by-frame: imagen-por-imagen. Secuencia de imágenes que a gran velocidad dan sensación de movimiento. Cuanto más frames mejor definición tendrá.

Frame: Imagen instantánea en la visualización de una reproducción.

GameObject: Cada uno de los elementos que forma parte de una escena de Unity.

Plugin: Archivo que hace de puente entre dos programas. Consiste en hacer compatible la información de un archivo en una plataforma.

GUI: Interfaz de Usuario Gráfica.

API: Application Program Interface

IU: Interfaz de Usuario.

Script: Porción de código simple, almacenada en texto plano, que ha de ser interpretada y ejecutada por otro programa externo a él.

Variable booleana: Variable que se utiliza en lenguajes de programación y puede tomar dos valores: "True" (Cert) o "False" (Falso).

Latencia: Tiempo de retraso que tarda en reproducirse una acción en el juego tras haberse realizado.

Path: Recorrido de archivos para llegar a uno concreto.

String: Variable que toma valor en forma de texto.

2. Prefacio

2.1. Origen del proyecto

La decisión de realizar este proyecto fue impulsada por motivos diversos. El primero fue que la realidad virtual estaba en auge y que consideraba sumamente importante sumarme a ese grupo de gente que puede crear algo con esta tecnología. El segundo motivo fue por la ayuda que se podía proporcionar a gente con Alzheimer con esta tecnología. Y la última, y no por ello menos importante, fue que quería realizar un proyecto que supusiera un reto personal, programando en un lenguaje de programación desconocido, con poco tiempo para la imaginación, pero con muchas ganas.

2.2. Motivación

Se puede hablar de una motivación de dobles raíces para este proyecto, ya que esta la parte de la realidad virtual y la parte del Alzheimer. Para la parte de realidad virtual, la motivación fue el reto que suponía, como se ha comentado anteriormente. En el caso del Alzheimer, hace más de 6 años que perdí a un familiar por esta dura enfermedad y me pareció una oportunidad para aprender más del ámbito e intentar aportar algo para su posible prevención.

2.3. Requisitos previos

Por suerte, para este proyecto no había que tener ningún requisito previo. Era un proyecto desde cero y aunque una base de C# hubiera sido realmente beneficiosa, no era necesaria. Lo único que se pedía al estudiante eran ganas de trabajar.

3. Introducción

Antes de empezar el proyecto se fijaron unos objetivos y se analizó su alcance, para una realista y eficaz realización del trabajo.

3.1. Objetivos del proyecto

Los objetivos de este trabajo han sido los siguientes:

- Entender el funcionamiento de la realidad virtual mediante la plataforma Unity
- Encontrar o transformar la tecnología existente para culpabilizar al juego de interactuar con el mediante la voz
- Incrementar la reserva cognitiva del usuario mediante un juego de memoria
- Servir como referencia para usuarios que quieran utilizar esta tecnología

Para que el cumplimiento de estos objetivos fuera posible, se ha utilizado el motor de juegos Unity.

3.2. Alcance del proyecto

Como se ha explicado en el apartado anterior el alcance de este proyecto será la creación de un videojuego capaz de aumentar la reserva cognitiva del usuario, y así poder reducir sus posibilidades de padecer Alzheimer, y la búsqueda y análisis de una tecnología que capacite al programador de crear una aplicación donde se pueda interactuar mediante la voz.

En cuanto al alcance de la aplicación, esta no está pensada para ser comercializada, ya que un proceso de comercialización dura meses e incluso años, tiempo que supera con creces el periodo de realización de proyecto, de unos cuatro meses.

4. Realidad Virtual

A lo largo de su vida un individuo probablemente nunca vaya a Marte, a la Luna, ni pilote un Boeing 747, pero si las expectativas de la realidad virtual se cumplen cualquier persona podrá hacer cada una de estas actividades sin ni siquiera salir de casa. La RV consiste en la inmersión sensorial en un nuevo mundo, basado en entornos reales o no, que ha sido generado de forma artificial, y que podemos percibir gracias a unas gafas de realidad virtual y sus accesorios (cascos de audio, guantes, etc). El objetivo de esta tecnología es crear un mundo ficticio del que puedes formar parte e incluso ser el protagonista: viendo un coche en un concesionario virtual, siendo protagonista de un videojuego o bien practicando como hacer una operación a corazón abierto.

Ilustración 4.1 Realidad Virtual

4.1. ¿Qué es la realidad virtual?

La realidad virtual la define la RAE como la representación de escenas o imágenes de objetos producidos por un sistema informático, que da la sensación de su existencia real.

Como pasó con los teléfonos móviles o internet, la aparición de la realidad virtual supone uno de los cambios tecnológicos más importantes de los últimos tiempos. Aunque aún la sociedad no sea consciente, la Realidad Virtual es una tecnología

revolucionaria y su adaptación a nivel usuario supondrá un antes y un después en la forma de consumo de contenidos multimedia: videojuegos, cine, eventos deportivos, conciertos, documentales… Estas son los elementos característicos de la realidad virtual:

- Mundo Virtual: Un mundo virtual es un entorno en tres dimensiones, donde el usuario puede interactuar con otros y crear objetos como parte de esta interacción. En el mundo virtual, las perspectivas virtuales responden a los cambios en los movimientos del usuario igual que pasaría en el mundo real. Por ello un buen mundo virtual ha de tener dos requisitos:
    - Explorable: Un mundo o escena virtual ha de ser grande y detallado para que así el usuario pueda explorar a sus anchas.
    - Realista/Creíble: Al utilizar un dispositivo de realidad virtual, el usuario tiene que pensar que está verdaderamente en otro lugar, sino la "ilusión" de la realidad virtual desaparecerá.
- Interactivo: A medida que el usuario se mueve por el mundo virtual, la RV tiene que moverse con él.
- Generado por un ordenador: Solo ordenadores suficientemente buenos serán capaces de crear gráficos realistas, interactivos y que cambien en tiempo real a medida que el usuario se mueve.
- Inmersivo: La inmersión en la RV es la percepción de estar físicamente presente en un mundo no físico. Engloba la sensación de presencia, el cual es el punto donde el cerebro humano cree estar en un lugar donde en realidad no está. Hay dos inmersiones que se tienen que cumplir.
    - Inmersión Mental: Un profundo grado de inmersión mental, donde la incredulidad de estar en un mundo virtual desaparece.
    - Inmersión Física: Un profundo grado de inmersión Física, donde la incredulidad de estar en un mundo virtual desaparece.

4.2. Tipos de realidad virtual

Existen varias categorías de tecnologías de realidad virtual. Los diversos tipos de realidad virtual difieren en sus niveles de inmersión y aplicaciones. A continuación, se exploran algunas de las diferentes categorías de realidad virtual:

No inmersiva

Las simulaciones no inmersivas son la implementación menos inmersiva de la tecnología de realidad virtual. En una simulación no inmersiva, solo se estimula un subconjunto de los sentidos del usuario, lo que permite una percepción periférica de la realidad fuera de la simulación de realidad virtual. Los usuarios ingresan en estos entornos virtuales tridimensionales a través de una ventana, mediante el uso de monitores estándar de alta resolución con poder de procesamiento que normalmente se encuentran en estaciones de trabajo de escritorio convencionales.

Ilustración 4.2 RV No Inmersiva

Semi-inmersiva

Las simulaciones semi-inmersivas brindan una experiencia más inmersiva, en la que el usuario se sumerge en parte, pero no completamente en un entorno virtual. Las simulaciones semi-inmersivas se parecen mucho y utilizan muchas de las mismas tecnologías que se encuentran en la simulación de vuelo. Las simulaciones

semi-inmersivas son impulsadas por sistemas de computación gráfica de alto rendimiento, que a menudo se combinan con sistemas de proyección de pantalla grande o sistemas de proyección de televisión múltiples para estimular adecuadamente las imágenes del usuario.

Ilustración 4.3 RV Semi-inmersiva

Totalmente Inmersiva

Las simulaciones totalmente inmersivas proporcionan la implementación más realista de la tecnología de realidad virtual. En una simulación totalmente inmersiva, el hardware como pantallas montadas en la cabeza y dispositivos de detección de movimiento se utilizan para estimular todos los sentidos de los usuarios. Las simulaciones totalmente inmersivas pueden proporcionar experiencias de usuario muy realistas al ofrecer un amplio campo de visión, altas resoluciones, mayores tasas de actualización (también llamada frecuencia de actualización) y altos niveles de contraste en la pantalla montada en la cabeza del usuario.

Ilustración 4.4 RV Totalmente Inmersiva

4.3. Dispositivos de realidad virtual

1. PC (Personal Computer)/Consola/Smartphone

   El contenido de realidad virtual, aquello que ve el usuario cuando usa los dispositivos de visualización, es tan importante como el dispositivo en sí. Si se quiere realizar un buen mundo virtual con escenas 3D interactivas, se necesita mucha capacidad de computación. Es ahí donde entra el PC, las consolas, y los Smartphone. Estos actúan como el motor que da hace posible la producción del contenido multimedia.

2. HMD (Head-Mounted Display)

   Un HDM es un tipo de dispositivo que contiene una pantalla montada frente a los ojos de un usuario. Esta pantalla generalmente cubre el campo de visión completo del usuario y muestra contenido de realidad virtual. Algunas pantallas montadas en la cabeza de realidad virtual utilizan pantallas de teléfonos inteligentes, como Google Cardboard o Samsung Gear VR. Las pantallas montadas en la cabeza a menudo se acompañan con auriculares para proporcionar estimulación de audio.

3. Dispositivos de entrada

Los dispositivos de entrada son una de las dos categorías que proporcionan a los usuarios una sensación de inmersión (es decir, convencer al cerebro humano para que acepte un entorno artificial como real). Proporcionan a los usuarios una forma más natural de navegar e interactuar dentro de un entorno de realidad virtual. Algunas de las formas más comunes de dispositivos de entrada de realidad virtual incluyen:

- Joysticks
- Force Balls/Tracking Balls
- Controller Wands
- Data Gloves
- Trackpads
- On-Device Control Buttons

5. La memoria

La memoria es un pilar básico para construir la historia de un individuo, definir quién es y para mantener el sentido de la continuidad. Gordon Allport lo resume diciendo que "necesitamos la memoria para asegurarnos que, al levantarnos cada mañana, somos la misma persona que se fue a dormir la noche anterior".

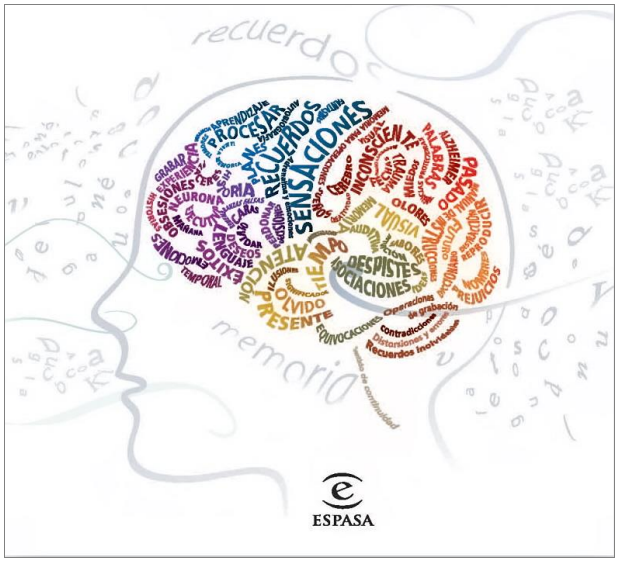

Ilustración 5.1 Memoria

Muchas veces la gente dice que pierde la memoria, o que se les olvida algo. No es que sea un error de la memoria, sino, de atención. Lo que ha pasado es que el individuo no ha estado suficientemente atento a lo que tenía que hacer y se ha distraído debido a algún estimulo externo. Otras veces se producen olvidos porque el tema no interesa lo suficiente o se ha vivido una situación con un componente emocional muy alto que ha dificultado la grabación precisa del hecho, como un accidente. Por ello, por norma general, las personas suelen olvidar experiencias desagradables bloqueándolas

La mayor parte de la gente sufre lapsus de memoria que, a veces, les puede poner en situaciones comprometidas. El factor de la edad también influye, por término medio un joven puede olvidar una vez o dos por semana un nombre propio

mientras por encima de los sesenta le pasará de cuatro a cinco veces.

La memoria humana necesita realizar con eficacia cuatro funciones esenciales:

• Grabar los detalles de las experiencias significativas vividas, las aptitudes y conocimientos que aprendidos y las acciones que se tiene intención de realizar en un futuro.

• La memoria debe ordenar y reconstruir los recuerdos con el fin de adaptarlos al guion de la vida y los cambios que se han ido experimentando en el paso del tiempo.

• Debe evocar, con razonable exactitud y rapidez, los datos que se han guardado y controlar su permanencia.

• Debe olvidar. Tiene que captar y retener una abundante variedad de información relevante y eliminar la que no lo es tanto.

5.1. Cerebro: anatomía

El sistema nervioso (SN), junto con el sistema endocrino, son el rector y el coordinador de todas las actividades conscientes e inconscientes del organismo. Está formado por el sistema nervioso central (SNC), comprendido por el encéfalo y la médula espinal, y el sistema nervioso periférico (SNP), compuesto por el conjunto de los nervios.

El **encéfalo** es la masa nerviosa dentro del cráneo, que está rodeada por meninges, que son tres membranas llamadas: duramadre, piamadre y aracnoides. Consta de tres partes: el cerebro, el cerebelo y el bulbo raquídeo, y otras más pequeñas como el hipotálamo, la hipófisis, los núcleos de la base, el cuerpo calloso, entre otros.

El **cerebro** es la parte más importante y está formado por dos partes. Primero, está la sustancia gris, que es la que está situada más al exterior. Está constituida por mil

millones de neuronas que forman billones de conexiones entre ellas. La sustancia blanca está situada en el interior de la sustancia gris y contiene fibras nerviosas rodeadas de un tipo de grasa blanquecina, llamada mielina, que sirve de soporte y actúa de aislante para facilitar la transmisión de los impulsos entre las neuronas. El cerebro no pesa más de un kilogramo y medio (depende de cada persona) y consume el 20% de todo el oxígeno que se espira. En la mayoría de animales el cerebro se encuentra dentro de la cabeza, por ello se le puso el nombre al encéfalo, ya que viene del griego “dentro de la cabeza” y es ahí donde está el cerebro protegido por el cráneo.

Las **neuronas** están formadas por un núcleo (la cabeza), las dendritas (el cabello), y los axones. Estos están rodeados por mielina, que es una lipoproteína que hace acelerar la velocidad de transmisión de la información. Las neuronas se unen a otras neuronas para 10.000 conexiones sinápticas a través de los axones y de las dendritas. Funcionan a través de descargas eléctricas que transportan la información para acabar en una acción final, como, por ejemplo, levantar el brazo derecho.

Controlan la transición entre los estados de sueño y vigilia, fundamental para el correcto funcionamiento del cerebro ya que, por ejemplo, aprovecha el estado de sueño para organizar la información adquirida durante el estado de vigilia. Un insomnio prolongado puede producir una enfermedad mental e, incluso, alucinaciones. Cada estado cerebral está asociado a unas determinadas ondas de actividad.

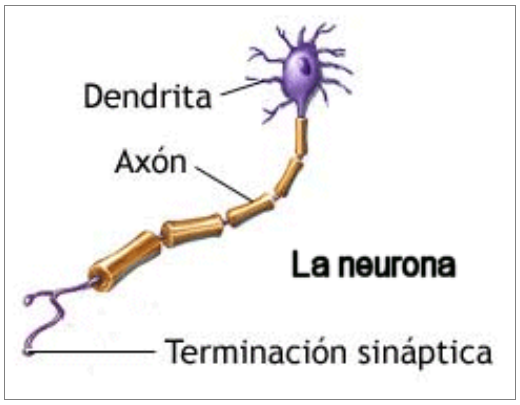

Ilustración 5.2 Neurona

La **cisura interhemisférica** divide el cerebro en dos hemisferios simétricos, unidos por las comisuras interhemisféricas. Cada hemisferio tiene varias cisuras que el subdividen en lóbulos.

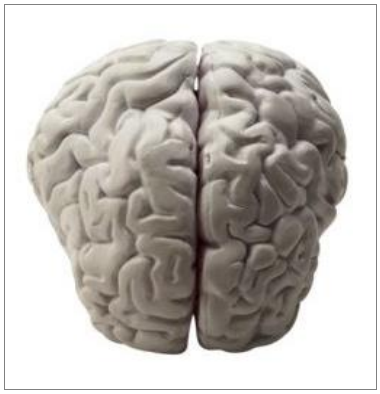

Ilustración 5.3 Cisura Interhemisférica

Hay cuatro lóbulos con funciones específicas:

- Lóbulo frontal: Se relaciona con el control de los impulsos, el sentido común, la producción del lenguaje, la memoria funcional / de trabajo, las funciones motoras, el comportamiento sexual, la socialización y la espontaneidad. También está relacionado con la planificación, la coordinación, el control y la ejecución de las conductas.

- Lóbulo temporal: En este lóbulo residen las principales funciones de la memoria. Está dividido en dos partes:

- Lóbulo temporal dominante, que es el encargado de recordar palabras y los nombres de los objetos, la llamada memoria semántica.

- Lóbulo temporal no dominante, que está relacionado con la memoria visual (recordar caras de personas, imágenes ...)

- Lóbulo occipital: Es donde está la corteza visual, y por lo tanto, está asociado con nuestra capacidad para interpretar lo que vemos.

- Lóbulo parietal: Es el encargado de procesar la información sensorial procedente de varias partes del cuerpo, el conocimiento de los números y sus relaciones, y de la manipulación de los objetos.

El **cerebelo** está situado detrás del cerebro y es más pequeño. También presenta una sustancia gris y una blanca, pero la blanca tiene forma de árbol y por eso también se le llama el árbol de la vida. Es el encargado de coordinar todos los movimientos de los músculos y realizar actividades motoras, como caminar, ir en bicicleta... Está relacionado con la memoria motora.

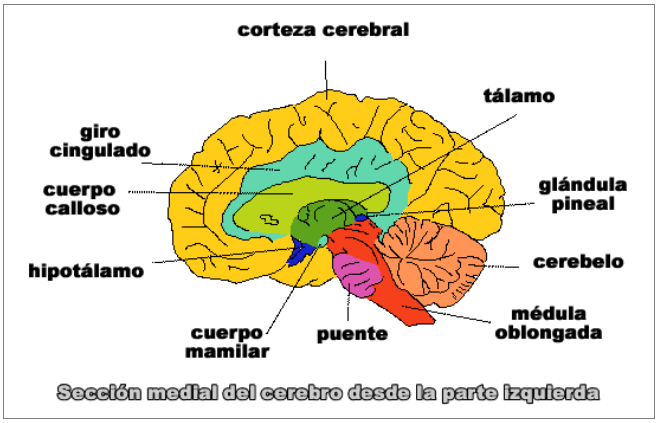

Ilustración 5.4 Partes del cerebro

5.2. Demencia

La demencia se define como un trastorno degenerativo progresivo de la capacidad cognitiva, que es una de las principales causas de la discapacidad en las personas mayores. De momento, no existe un tratamiento que la cure del todo.

Hay cuatro tipos de demencias:

- La demencia vascular
- La demencia vascular mixta
- La demencia frontotemporal
- La enfermedad del Alzheimer

Este proyecto se centra en el último tipo de demencias mencionado, el Alzheimer, por lo que a continuación se profundizará en más detalle.

5.2.1. Alzheimer

La enfermedad de Alzheimer (EA) es una enfermedad neurológica encuadrada en el grupo de las neurodegenerativas, son las que producen degeneración de las células esenciales del sistema nervioso, las neuronas. Es la enfermedad que más frecuentemente causa pérdida de facultades mentales en el anciano.

La enfermedad de Alzheimer tiene un pico de aparición creciente a partir de los 70 años de edad. Es relativamente rara por debajo de los 65 años, aunque hay casos descritos con inicio, excepcionalmente atípico, en la juventud. El hecho de que afecte más a ancianos ha igualado erróneamente envejecimiento a enfermedad de Alzheimer. No todos los ancianos sufren Alzheimer, ni la enfermedad de Alzheimer ocurre solamente en ancianos.

La enfermedad de Alzheimer produce una degeneración neuronal que afecta principalmente a neuronas altamente especializadas encargadas de realizar las

funciones que más caracterizan a los seres humanos. De hecho, la enfermedad de Alzheimer como tal, parece propia de los seres humanos. Se han descrito cambios conductuales en animales de compañía ancianos, pero las alteraciones en sus cerebros no son parecidas a las de la enfermedad de Alzheimer. Hay que tener en cuenta que los humanos son los mamíferos terrestres con la mayor longevidad media.

La enfermedad de Alzheimer afecta a más de medio millón de personas en nuestro país y a más de 45 millones en todo el mundo y es una de las causas mayores de discapacidad. Afecta al núcleo familiar y es causa de afecciones en las personas que los atienden. Plantea también muchos dilemas morales que atañen a la capacidad de decidir, el manejo al final de la vida y la distribución de recursos a la población anciana pensionista. Así que, a la complejidad médica se añade la complejidad familiar, social y, en definitiva, política que esta enfermedad tiene.

5.2.2. Etapas

En las etapas tempranas del Alzheimer, antes de que los síntomas puedan ser detectados por pruebas actuales, las partes del cerebro que primero se ven afectadas son las de Aprender y recordar y Pensar y planear.

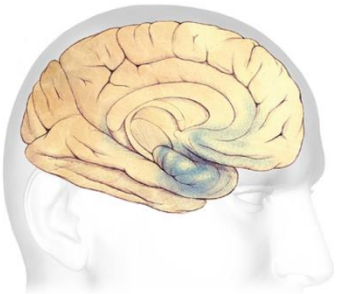

Ilustración 5.5 Estado del cerebro en las etapas tempranas del Alzheimer

En las etapas leves o moderadas del Alzheimer, las regiones del cerebro que son importantes para la memoria, el poder pensar y planear son las que se ven más dañadas. Esto resulta en problemas de la memoria o del pensamiento tan serios

como para interferir con el trabajo o la vida social. Las personas en esta etapa pueden volverse confundidas y tener problemas al manejar el dinero, expresarse y organizar sus pensamientos. Muchas personas con la enfermedad de Alzheimer son diagnosticadas durante estas etapas.

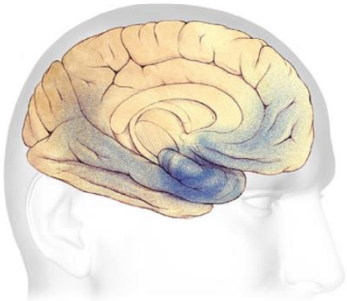

Ilustración 5.6 Estado del cerebro en las etapas leves o moderadas del Alzheimer

En las etapas avanzadas del Alzheimer, la mayoría de la corteza está seriamente dañada. El cerebro se encoge dramáticamente debido a la muerte de un gran número de células. Las personas que padecen de Alzheimer pierden su habilidad de comunicarse, reconocer a su familia y a sus seres queridos y de cuidarse de sí mismo.

Ilustración 5.7 Estado del cerebro en las etapas avanzadas del Alzheimer

5.2.3. Prevención del Alzheimer

Como se ha comentado anteriormente la enfermedad del Alzheimer es la causa más común de la demencia. Es una enfermedad neurodegenerativa, progresiva e

irreversible, causante de grandes costes sociales y económicos en las sociedades modernas.

El impacto social de esta enfermedad no se basa tan sólo en los enfermos, más de 45 millones, sino que supone un desgaste emocional, de salud y económico para sus familiares y amigos. El coste económico anual para cuidar a una persona se estima en 27.000 euros anuales. Las estadísticas advierten que el número de afectados por demencia superará los 135 millones en 2050, lo que supondrá un coste difícilmente asumible por nuestras sociedades, especialmente teniendo en cuenta que un 58% de los afectados viven en países con sueldos medios y bajos.

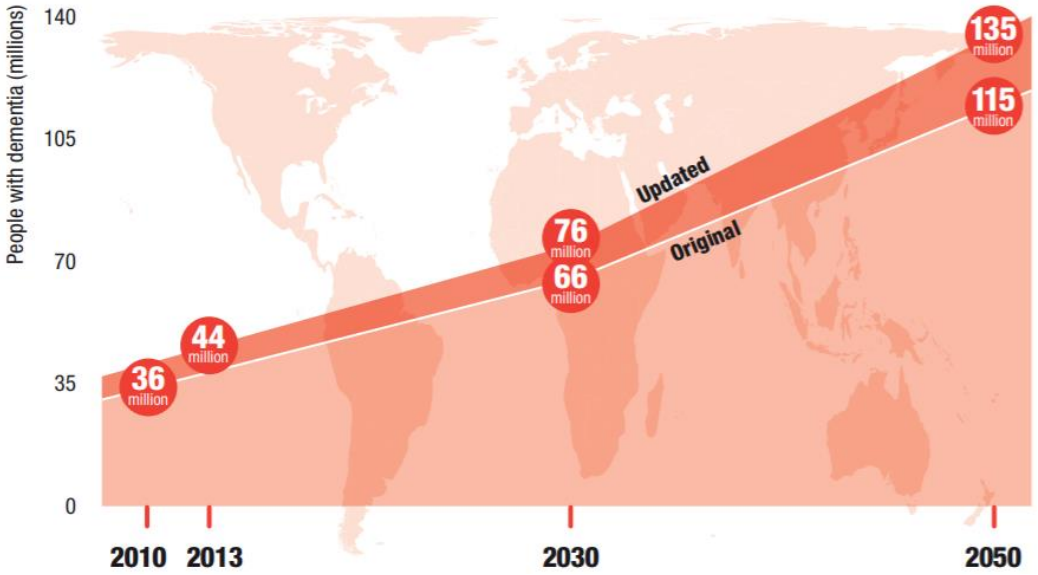

Ilustración 5.8 Comparación del número de enfermos de Alzheimer original y actualizado

Por ello cada vez más investigadores se centran no solo en curar la enfermedad, pero también en prevenirla.

Hoy en día se sabe que el Alzheimer es una enfermedad de larga duración, que empieza a producir daños en el cerebro 15 años antes de que se muestran los primeros síntomas. Consecuentemente las estrategias actuales de prevención tienen como objetivo retrasar o incluso frenar la aparición de los síntomas. Para ello es esencial el ejercicio físico, la actividad cognitiva, las relaciones sociales y la

dieta.

Este proyecto se centra en ejercitar la actividad cognitiva aumentando así la reserva cognitiva, a continuación se hablará más detalladamente al respecto.

5.2.3.1. Reserva cognitiva

Una de las claves para mejorar la salud cerebral es mantenerse cognitivamente activos a lo largo de toda la vida. El cerebro sufre el paso de los años como cualquier otro órgano, pero una alta reserva cognitiva puede ser una muy buena arma contra ello.

La acumulación de la experiencia y la estimulación de las capacidades mentales a lo largo de la vida se refleja en lo que se llama “reserva cognitiva”. Esta se puede considerar como un capital mental, cuanto mayor sea más fácilmente compensará los efectos en la eficiencia cognitiva del envejecimiento y del Alzheimer.

Una de las mejores maneras de aumentar la reserva cognitiva es mediante juegos. Estos permitirán al usuario entrenar distintas habilidades cognitivas en función de las características del juego, se estimulará el cálculo, la memoria reciente, la lógica, la capacidad de planificación, el vocabulario o la creatividad, entre otras capacidades.

Esta es la razón por la que se decidió centrarse en la creación de un videojuego para prevenir el Alzheimer, ya que es una de las mejores maneras para aumentar la reserva cognitiva.

6. Unity

Para el desarrollo de esta aplicación se ha empleado el motor de videojuegos Unity3D, el cual ha sido desarrollado por Unity Technologies. Aunque pueda parecer que la opción óptima hubiera sido trabajar directamente con el sistema operativo en el que se fuera a utilizar esta aplicación, eso es un error, ya que las aplicaciones creadas con Unity se pueden utilizar en los sistemas operativos: Windows, OS X y Linux y se pueden utilizar en plataformas de sobremesa como: PlayStation3, PlayStation4, XBOX 360, XBOX One, Wii U y 3DS. El hecho de que Unity de soporte a tantas plataformas y tenga una interface tan sencilla para usuarios no expertos, es la razón por la que esta plataforma es de las más utilizadas en el mundo. El lenguaje de programación con el que se ha trabajado en Unity es el C#, C Sharp.

Por otro lado, cabe mencionar que Unity3D da la al usuario la opción de escoger entre tres planes: Personal, Plus, Pro y Enterprise. En este proyecto se empleó el plan Personal ya que no se contemplaba la opción de ingresar más de 100.000$ con él, ya que entonces se tendría que haber utilizado el plan Plus.

Hay cierta terminología del programa que se empleará a lo largo del documento y que es imprescindible que el lector entienda:

Término	Definición
Asset	Es la representación de cualquier elemento que pueda usarse en el juego o proyecto. Puede provenir de un archivo creado fuera de Unity, como un modelo 3D, un archivo de audio, una imagen o cualquiera de los otros tipos de archivos que admite Unity. También hay algunos tipos de assets que se pueden crear dentro de Unity, como un Controlador de Animator, un Mezclador de Audio o una Textura de Procesamiento.

Escena	Una escena corresponde a un nivel o área en un juego, como puede ser un menú. El funcionamiento de Unity se basa en emplear escenas para los distintos escenarios que tendrán lugar a lo largo del programa. Por ejemplo, una escena puede ser “Menú Principal” que pase a la escena “Inicio Juego” con un click.
Game Object	Son objetos fundamentales en Unity que representan personajes, propiedades, y el escenario. Estos no logran nada por sí mismos, pero funcionan como contenedoras para Componentes, que implementan la verdadera funcionalidad.
Elemento prefabricado	Este nombre hace referencia a un objeto reutilizable, es decir, que puede emplearse en una o más escenas a partir de un modelo de almacenado. Esto se consigue creando una instancia del mismo. Su principal característica consiste en que al modificar el elemento inicial el resto de elementos del programa también se modifican.

Tabla 6.1 Terminología de Unity

Por otro lado, también es importante saber cuáles son los elementos principales de editor de Unity. A continuación, se hará un esquema de los diferentes elementos y se desarrollará cada uno con más detalle.

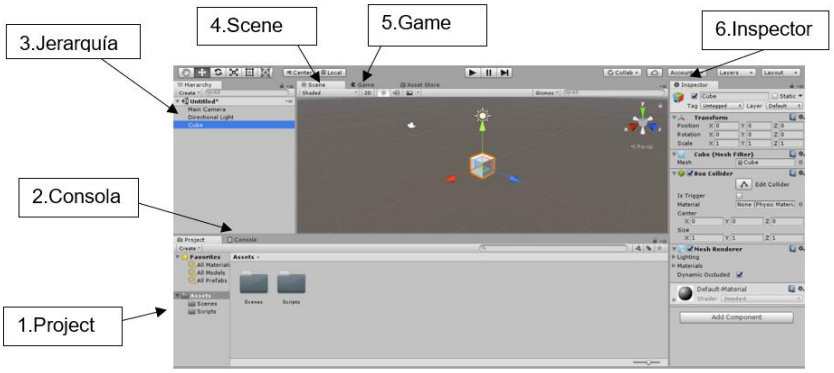

Ilustración 6.1 Esquema de diferentes elementos de Unity

6.1. Pestaña Project

El panel izquierdo del navegador muestra la estructura de carpetas del proyecto como una lista de jerarquía. Cuando una carpeta es seleccionada de una lista haciendo click, su contenido va a ser mostrado en el panel a la derecha. Los assets individuales son mostrados en el panel de la mano derecha como iconos que indican su tipo (script, material, sub-carpeta, etc). El espacio a la izquierda del deslizador muestra el elemento actualmente seleccionado, incluyendo una ruta completa al elemento si se está realizando una búsqueda.

6.2. Consola

La ventana de la consola es una de las más importantes del editor. Aquí es donde se mostrarán los errores que pueda haber en el código o mensajes que se consiguen a través de la comanda Debug.Log (“mensaje”).

6.3. Jerarquía

Toda escena de Unity funciona por objetos, siendo estos los elementos principales y con los cuales interactúa el usuario. Cómo estos objetos no se encuentran sólo en el juego, necesitan tener una relación con el resto de objetos. Esta relación se visualiza en la pestaña Jerarquía.

Hay dos relaciones posibles entre objetos. La primera es la llamada relación padre-hijo, es decir, un objeto se encuentra posicionado en referencia a otro, de forma que todo el que le pasa al objeto padre le ocurre al hijo, pero modificar el hijo no afecta al padre.

La otra relación es la relación de hermanos, donde los dos objetos comparten el mismo padre y, por lo tanto, el mismo punto de referencia. En esta relación la modificación de un no implica la modificación del otro. La relación padre-hijo es

importante puesto que reduce la necesidad de tener una gran cantidad de formas geométricas, sino que con planos esferas y cubos (o elipsoides y prismas si se escalan de forma no equilátera) se pueden crear los objetos deseados.

6.4. Escena

Cuando se genera un proyecto, este crea de forma automática lo que se denomina como una escena, un mundo virtual donde el usuario podrá colocar todos los objetos que quiera.

La creación de una escena comporta también la adición de dos elementos básicos a esta, una cámara desde donde se puede ver la visión que tendrá el usuario al hacer uso de la aplicación y una luz direccional, simulando una estrella en el firmamento. Unity permite la creación de más de una escena dentro del mismo proyecto y permite de cambiar forma rápida entre estas, siendo cada una un mundo virtual independiente del resto con sus propios objetos.

La función de la pestaña Scene es ver la escena concreta seleccionada, haciendo los cambios de la manera más intuitiva. Además, un conjunto de 6 botones permite moverte por la escena y hacer las modificaciones más básicas de los objetos.

Ilustración 6.2 Botones para fácil movimiento y edición

El botón en forma de mano permite, si no se ha clicado en ningún objeto, mover la escena sin girarla, permitiendo la visualización de cualquier punto de esta. Para girar la escena hay que mantener pulsado el botón derecho del ratón y desplazar el ratón, pero para esta acción no es necesario el botón de la mano.

El segundo botón permite al usuario seleccionar un objeto y cambiar su orientación, esto se hace de manera intuitiva ya que aparecen los tres ejes cartesianos, cada uno con un color.

Los últimos tres botones permiten una edición más compleja y detallada de los objetos, el último de ellos es según muchos usuarios, el más útil.

6.5. Game

Cómo ya se ha comentado anteriormente, cada escena tiene de forma predeterminada una cámara desde la cual se verá la escena al activar la aplicación. Desde la pestaña Game se puede ver la escena desde esta cámara, a sabiendas de la visión que tendrán los usuarios que utilicen la aplicación.

Si no existiera esta cámara, la pestaña se vería completamente negra y si existe más de una, hay una opción para escoger cuál utilizar. Esta cámara se comporta como cualquiera otro objeto y no tiene por qué estar fijo. Por ejemplo, para algunos juegos puede ser importante que la cámara siga al avatar.

6.6. Inspector

Los objetos de cualquier escena tienen propiedades (Componentes en Unity), cómo la posición, el material, la colisión con otros objetos... Estas propiedades se pueden añadir, manipular o eliminar desde la pestaña Inspector. Es en la cantidad de propiedades predeterminadas de Unity donde se ve el potencial de este motor de juegos. Cómo ya se ha comentado, propiedades como que un objeto tenga gravedad o que este choque con otro ya han sido creadas, y estas son muy complejas de realizar desde cualquier otra aplicación.

Pero esta pestaña no se reduce sólo a propiedades, sino que también se pueden añadir acciones a los objetos, accionados por un click o una tecla. Por último, y es aquí donde surge la magia de los juegos, se pueden añadir acciones concretas mediante ficheros, escritos en lenguaje C# o UnityScript, un lenguaje diseñado específicamente para uso con Unity y modelado tras JavaScript.

6.6.1. Scripts

Cómo se ha comentado anteriormente, aunque el Inspector tenga una gran cantidad de propiedades predeterminadas, no es posible hacerlo todo a partir de él. Para crear acciones más complejas y no predeterminadas existen los Scripts, ficheros que se adjuntarán a los objetos con el fin de controlarlos al gusto del programador.

Estos Scripts se pueden escribir en dos lenguajes de programación, Java o C#. Antes de crear el Script se tiene que definir en qué lenguaje se diseñará. C# es un lenguaje de programación desarrollado por Microsoft que deriva de C y C++ y que está orientado a objetos. El lenguaje UnityScript proviene de Java. Este fue creado por Sun Microsystems, es de propósito general y orientado también a objetos. En este proyecto solo se ha utilizado código de lenguaje C#.

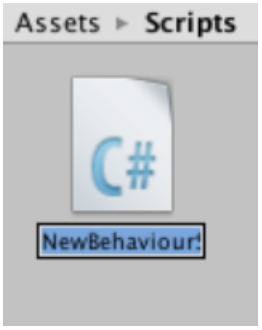

Ilustración 6.3 Visión de un script en Unity

7. Reconocimiento de voz

El reconocimiento de voz es útil para R no solo para simular conversaciones con agentes IA, sino también para que el usuario se comunique con cualquier aplicación que requiera una gran cantidad de opciones. Escribir una respuesta o un comando puede ser demasiado poco práctico, y la saturación de la aplicación con botones puede confundir muy rápido. El limitado uso de los botones se puede apreciar claramente en las aplicaciones de móvil de realidad virtual, ya que por lo general, debido a que la aplicación está diseñada para ser dirigida de manera táctil, al añadirles el elemento auxiliar para su visualización, las gafas, estas quedan inaccesibles y pierden toda su funcionalidad.

La organización mundial de la salud, la ONU, indica que un 98% de la población es capaz de hablar, por lo que pueden utilizar esta herramienta con facilidad mientras están en una experiencia de realidad virtual. Es por ello que los comandos de voz son el futuro. La ciencia ficción los ha tenido durante décadas y, sin embargo, todavía se tiene que utilizar el mando para encender el televisor o activar una alarma.

Agregar una interfaz de voz a una aplicación o dispositivo debería ser simple, por ello al crear esta aplicación lo primero que se analizó fueron diferentes maneras de incluir las comandas mediante órdenes orales. El reconocimiento por voz se puede apreciar en aplicaciones de grandes compañías, tales como Siri o Bixby.

Por desgracia, la UPC no contaba con un sistema para ello y es por esta razón por la que se realizó un exhaustivo análisis de diferentes sistemas antes de elegir el mecanismo óptimo para ello. Cabe mencionar que para este proyecto se buscaba un método de reconocimiento de voz que detectara palabras en concreto y que fuera accesible para el máximo número de usuarios.

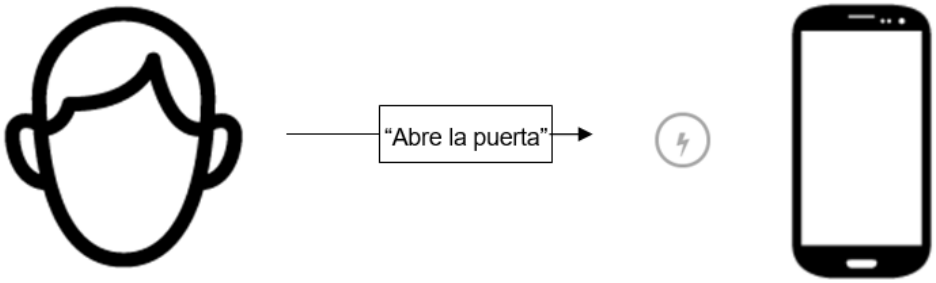

Ilustración 7.1 Reconocimiento de voz

A continuación, se muestran algunas de las alternativas planteadas durante el proceso de aprendizaje y la elección final.

- IBM Watson Speech to text: El servicio Watson Speech to Text permite convertir fácilmente audio y voz en texto para un entendimiento rápido del contenido.

  Por desgracia se notaron dos significativos problemas en este sistema. El primero era la necesidad de estar conectado a la red, ya que eso restringía el uso de la aplicación. El segundo problema consistía en la precisión y la latencia del sistema, al no saber que palabras tenía que reconocer, el sistema iba lento y no siempre acertaba con el comando pedido.

- Wit.ai: El funcionamiento de esta herramienta es parecida al reconocimiento de voz mediante IBM Watson, con la notable mejora de que aquí el creador de la aplicación puede especificar que palabras quiere que sean reconocidas. Lamentablemente en este segundo sistema también es necesaria la conexión a internet.

Por último, se analizó el Windows speech-to-text, y dentro de este se encontraron tres alternativas:

- Dictation Recognizer: Este reconocimiento de voz se parece mucho a los dos anteriores. Puede reconocer cualquier frase, pero depende de la conexión a internet, por lo que no sería la opción óptima para reconocer palabras en concreto y llegar al máximo número de usuarios.

- Grammar Recognizer: El reconocimiento por este sistema está pensado para reconocer frases en lenguaje natural, pero de manera programada. Se programa mediante reglas gramaticales que frases puede decir el usuario utilizando un archivo SRGS. Este método es muy útil ya que funciona sin necesidad de internet, pero abarca más de lo que se necesita para este proyecto.

- Keyword Recognizer: Este último método ha sido el elegido para este proyecto ya que permite reconocer palabras o series de palabras de una lista determinada por el programador. Keywordrecognizer funciona sin necesidad de internet lo que termina convirtiendo este método la opción óptima.

A continuación, se explicará con más profundidad la opción elegida para que se entienda de cara a su uso por un futuro estudiante.

7.1. KeywordRecognizer

Al descubrir este método y tras exhaustiva investigación el primer código probado, y el más sencillo, fue este. Este código es muy visual ya que con la simple creación de una lista y la introducción de las palabras del juego, ya detectaba lo que el usuario decía.

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

public class EjemploKeyWordRecognizer : MonoBehaviour
{
    KeywordRecognizer keywordRecognizer;

    // Creación de lista de palabras

    public string[] Keywords_array;

    // Inicialización del código

    void Start()
    {
        // Cambio del tamaño de la lista a gusto del programador e
introducción de las frases (Ej: longitud lista 4)

        Keywords_array = new string[4];
        Keywords_array[0] = "Frase uno";
        Keywords_array[1] = "Frase dos";
        Keywords_array[2] = "Frase tres";
        Keywords_array[3] = "frase cuatro";

        // creación de instancias de KeywordRecognizer

        keywordRecognizer = new KeywordRecognizer(Keywords_array);
        keywordRecognizer.OnPhraseRecognized += OnKeywordsRecognized;

        // Comienzo del reconocimiento de palabras

        keywordRecognizer.Start();
    }
    void OnKeywordsRecognized(PhraseRecognizedEventArgs args)

      // Se envía a la consola para una comprobación de la palabra
reconocida
    {
        Debug.Log("Keyword: " + args.text + "; Confidence: " + args.c
onfidence + "; Start Time: " + args.phraseStartTime + "; Duration: "
+ args.phraseDuration)
```

A pesar de la poca complejidad del código, este script no cumplía con todas sus funciones, ya que no solo se quería que pudiera reconocer palabras, sino que se buscaba que estás palabras pudieran variar.

Es por ello que se elaboró un segundo código más extenso, que se puede encontrar en el anexo, que permitía ir variando la lista de palabras partiendo de la herramienta de Unity Keywordrecognizer.

7.2. KeywordRecognizer Personalizado

Para poder crear este nuevo se utilizaron los mismos conceptos que el código anterior y las siguientes herramientas.

- El patrón Singleton: Garantiza que una clase sólo tenga una instancia (objeto) y proporciona un punto de acceso global, desde cualquier script, a ella.

```
public class KeyWordRecognizer{
    static KeyWordRecognizer _singleton;
    public static KeyWordRecognizer singleton {
        get {
            if (_singleton == null)
                _singleton = new KeyWordRecognizer();
            return _singleton;
        }
    }
```

- Diccionario: Se utilizará un diccionario {clave:Valor} para poder guardar una palabra con su acción. Esto permitirá que cuando el sistema reconozca una palabra pueda llamar a la acción de inmediato.

```
Dictionary<string, Action> keywords = new Dictionary<string, Act
ion> ();
```

- Delegados: Un delegate es un tipo de referencia que puede usarse para encapsular un método con nombre o anónimo. En este caso

consideramos la acción del diccionario nuestro delegado, y lo llamamos. Con la función Debug.log enseñamos en la consola "Keyword recognized: palabra reconocida"

```
void OnKeywordRecognized(PhraseRecognizedEventArgs args){
        Action @delegate;
        Debug.Log ("Keyword recognized: " + args.text);
        if (keywords.TryGetValue (args.text, out @delegate))
            @delegate ();}
```

Tras contemplar las diferentes herramientas utilizadas la mejor manera de entender como funcionara el programa es con un ejemplo que lo utilice. La funcionalidad se entenderá a continuación, en la sección de la explicación del juego.

8. Descripción de la aplicación

Este proyecto, como bien se ha explicado varias veces a lo largo del documento, es un juego con dos claras cosas a resaltar, el fin de ayudar en la prevención del Alzheimer aumentando la reserva cognitiva del jugador y la incorporación de la voz en cada nivel del juego.

El juego consiste en entrar en una habitación, donde los diferentes objetos han sido creados de manera independiente, y darle al jugador un tiempo para mirarla y analizarla.

Tras ese periodo de tiempo y tras la aparición de una pantalla negra, ciertos objetos desaparecerán y serán substituidos por cubos grises numerados. El jugador tendrá que ir llamando los diferentes cubos por sus respectivos números, si se acuerda de lo que eran, y tras un zoom a dicho objeto, el jugador tendrá que elegir cuál de las imágenes del inventario mostrado en pantalla coincide con el objeto que busca.

Tras llamar el objeto de la imagen y acertar, la cámara deshará el zoom y el jugador podrá probar a acertar el contenido inicial de otro de los cubos grises. Si cuando el jugador llama a uno de los números se arrepiente, podrá volver al punto de partida diciendo “atrás”

En cuanto a la puntuación, el juego no se basa en ganar o perder, sino en conseguir la mayor puntuación posible. Es por ello por lo que el juego lleva incorporado una pantalla con las puntuaciones máximas de cada nivel. Para ello, y según en qué nivel de dificultad se encuentre el jugador, el juego partirá de unos puntos u otros, disminuyendo estos a medida que pasa el tiempo y restando puntos o sumando si se acierta el objeto buscado dentro del inventario.

Para una comprensión más visual del juego, a continuación, se hablará en más detalle de las diferentes pantallas de este. El juego cuenta con tres pantallas principales:

- Pantalla de inicio
- Pantalla de selección de nivel
- Pantalla habitación

8.1. Pantalla de inicio

Será la primera pantalla que vea el jugador. Aquí el jugador podrá “entrar” en la casa o “salir” del juego (Ilustración 8.1). Es una pantalla muy sencilla que solo permite un leve ángulo de visión.

Ilustración 8.2 Pantalla de inicio

8.2. Pantalla de selección de nivel

Tras decir “entrar” en la pantalla aparecerá esta segunda pantalla (Ilustración 8.2). Aquí el jugador deberá elegir el nivel en el que desea jugar. Cada nivel equivale a una combinación de más objetos a desaparecer, menos tiempo de memorización y más opciones en el inventario final. En esta misma ilustración también se puede apreciar que, debajo de cada nivel existe una puntuación que equivale a la

puntuación máxima de ese nivel.

Ilustración 8.3 Selección de nivel

8.3. Pantalla habitación

La pantalla habitación es la principal del juego, donde apareceran y desaparecerán objetos como se ha mencionado anteriormente. Dentro de esta pantalla se encuentran diferentes fases y subfases, es por ello que es conveniente hacer una breve explicación de cada una.

- Memorización: En la fase de memorización el usuarió tendra cierto tiempo, variable que depende del nivel, para memorizar los objetos que hay en la sala. El tiempo que le queda al jugador aparecerá en la parte superior de la pantalla (ilustración 8.3), y se ira moviendo a la vez que el jugador para que este vea cuanto tiempo le queda.

Ilustración 8.4 Pantalla de memorización

- Juego: La fase de "juego" es la parte activa de este juego. Como su nombre bien lo indica es el momento donde el jugador se tendrá que poner a pensar y a memorizar, para así ganar el mayor número de puntos posibles y superar otras puntuaciones.

Ilustración 8.5 Habitación

  - DummyObjects (cubos grises): esta es la primera subfase, ya que es en la cual, tras el tiempo de memorización, los objetos son substituidos por cubos grises también llamados DummyObjects en el script. Cada cubo irá numerado para que, si el jugador se acuerda, lo pueda llamar refieriendose a este.

- Inventario: Como se ha dicho anteriormente, cuando el jugador llama al número del cubo, la cámara hace un zoom hacia dicho objeto y aparece un inventario en pantalla de donde el jugador tendrá que elegir cual es el objeto correcto para ese lugar. En el caso de acertar la cámara quitaría el zoom y volvería a la posición inicial de DummyObjects y el cubo se intercambiará por el objeto real.

Ejemplo del inventario dentro del Dummy 3, que se seleccionará diciendo “Silla”

Ilustración 8.6 DummyObject e inventario

- GameOver: Una vez encontrados todos los objetos es cuando interviene esta pantalla, la cual permite ir al “Menu” inicial, “Reintentar” la partida o “Salir” del juego y cómo durante todo el juego, se hará oralmente (Ilustración 8.6).

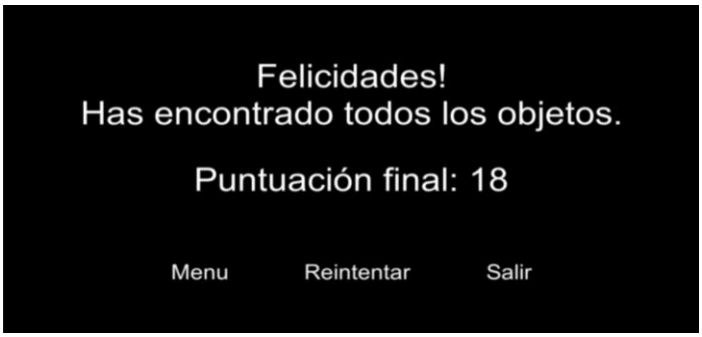

Ilustración 8.7 GameOver

8.4. Puntuación

El funcionamiento para la puntuación es muy sencillo. El jugador parte de 100 puntos base si ha escogido el nivel fácil, 200 en el nivel medio y 300 en el alto. Hay que tener en cuenta que estas puntuaciones se basan en que en el nivel bajo hay 30 segundos de memorización, en el nivel medio 25 y en el nivel alto 15. Y añadir que cuanto mayor es la dificultad, más objetos hay para memorizar.

Una vez en el juego cada objeto acertado suma 10 puntos y los intentos fallidos restan 10 puntos.

Por último, mencionar que desde el primer instante los puntos disminuyen debido a que baja un punto por segundo.

9. Retos del proyecto

Los dos grandes retos de este proyecto han sido el tiempo y la necesidad de aprender cosas de manera autónoma, aunque ambas van estrechamente ligadas.

La creación de un videojuego con la plataforma Unity en el cual se pudieran dar órdenes mediante la voz, implicaba el aprendizaje de un lenguaje de programación nunca utilizado antes, el C# (C Sharp) y una exhaustiva investigación sobre el reconocimiento de voz, sin el cual no se podía empezar la aplicación.

Tener que encontrar una nueva tecnología nunca es fácil, y especialmente si no se ha hecho antes. Si se le añade a este hecho la corta duración del proyecto, de unos 4 meses, aún es más complejo.

Por lo general en los proyectos importantes nunca sobra tiempo, ya que se intenta utilizar al máximo cada minuto con el fin entregar el mejor resultado posible. Así pasó en este proyecto, y por ello el tiempo se convirtió en uno de los mayores retos de este.

10. Presupuesto

Para evaluar el coste económico de este proyecto se ha calculado la suma de los costes individuales. Estos costes se pueden dividir en costes de recursos humanos, horas invertidas por los diferentes componentes del proyecto, y en el material utilizado para este, ya sea virtual o físico.

Para ello primero se han calculado las diferentes fases y el tiempo invertido en cada una de ellas:

FASES	HORAS
Definición y planificación del proyecto	15
Aprendizaje Unity	20
Aprendizaje C# y programación	230
Memoria	70
Reuniones con el director	10
Total	345

Tabla 10.1 Fases del proyecto

Para trabajar en las diferentes fases fue necesaria la participación de un ingeniero y un director, cobrando 40€/h y 60€/h respectivamente. Como se ha visto anteriormente se ha dedicado unas 345 horas al proyecto junto con unas 40 del director.

CONCEPTO	COSTE UNITARIO (€/h)	DEDICACIÓN (h)	COSTE (€)
Ingeniero	40	345	13.800
Director	60	50	3.000
Total	80	395	16.800

Tabla 10.2 Presupuesto recursos humanos

Por otro lado, se han tenido en cuenta los diferentes programas utilizados y que hay que cuantificar.

NOMBRE	COSTE (€)
Unity 2017.3.1f1 (64-bit)	0 (versión gratuita)
HTC VIVE	800
Ordenador	800
TOTAL	1.600

Tabla 10.3 Presupuesto material

Por último, solo queda sumar el coste de los recursos humanos y el coste del material:

NOMBRE	COSTE (€)
Recursos humanos	11.125
Material	1.600
TOTAL	18.400

Tabla 10.4 Presupuesto total

Por lo tanto, se deduce que el coste total aproximado de esta aplicación es de 18.400€ a lo que hay que aplicar el impuesto sobre el valor añadido, IVA, del 21%, por lo que el precio final queda de 22.264€.

11. Impacto ambiental

El impacto ambiental, por definición, es el conjunto de posibles efectos sobre el medio ambiente de una modificación del entorno natural, como consecuencia de obras o actividades.

El hecho de que este proyecto ayude a prevenir el Alzheimer influirá en disminuir los pacientes que padezcan esta enfermedad y por lo tanto a reducir los medicamentos que utilizan estos.

Por otro lado, para este proyecto se necesitan gafas de realidad virtual HTC VIVE, los cuales están hechos de plástico no siempre reciclable. Pero se puede considerar como un impacto ambiental indirecto ya que es un efecto producido por terceros.

Por lo tanto, se puede concluir que este proyecto tiene un impacto ambiental positivo.

12. Conclusiones

Este proyecto ha sido un gran reto ya que se ha tenido que aprender a utilizar el motor de videojuegos Unity 3D, aprender a programar en lenguaje C#, investigar un innovador método de dar comandos mediante la voz, hacer un videojuego que ayudará en la prevención del Alzheimer y realizar todo lo dicho anteriormente en unos cuatro meses.

Una vez terminado el proyecto se puede asegurar que se han cumplido los objetivos fijados al principio de este. Esto ha sido gracias a que Unity es una herramienta muy útil para crear juegos de Realidad Virtual y para la investigación de herramientas personalizadas en general.

El resultado obtenido es muy satisfactorio ya que, aunque el juego no esté listo para comercializar, ha cumplido con creces las funcionalidades que se le habían fijado y todo ello ayudando en la enfermedad del Alzheimer.

Durante estos últimos meses se ha trabajado mucho, pero con muchas ganas. Ha servido para adquirir conocimientos de informática y descubrir un nuevo mundo. La Realidad Virtual es una revolución tecnológica y trabajar en el Centro de Realidad Virtual ha sido una gran oportunidad para descubrir este mundo.

13. Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor del TFG, el profesor del departamento de matemáticas Toni Susin, quien me permitió realizar este proyecto y quien me ha ido guiando y aconsejando a lo largo de la creación de este.

Además, quería hacer una mención especial al centro de realidad virtual de la facultad de matemáticas y en especial a Jordi, quien estuvo siempre disponible para ayudar.

Por último, me gustaría agradecer a mi familia y amigos, ya que fueron ellos los que me animaron a hacer un proyecto de un tema de tanta sensibilidad en nuestra sociedad y han sido ellos los que han opinado sobre cada detalle y modificación del juego y sobre todo los que han aguantado el estrés sufrido en el proceso de aprendizaje de programación.

14. Bibliografia

Bibliográfia sobre el motor de videojuegos Unity:

Unity Technologies – Unity [Fecha de consulta: 27/03/2018] Recuperado de: https://unity3d.com/es

Speech Recognition and VR– Dioselin Gonzalez [Fecha de consulta: 5/04/2018] Recuperado de: https://blogs.unity3d.com/es/2016/08/02/speech-recognition-and-vr/

Microsoft – C# Guide [Fecha de consulta: 20/04/2018] Recuperado de: https://docs.microsoft.com/en-us/dotnet/csharp/

Bibliográfia sobre el reconocimiento de voz:

Harvard University - Harvard Sentences [Fecha de consulta: 5/04/2018] Recuperado de: http://www.cs.columbia.edu/~hgs/audio/harvard.html

Netguru - Voice recognition tools review [Fecha de consulta: 06/04/2018] Recuperado de: https://www.netguru.co/blog/voice-recognition-tools-review

Bibliográfia sobre el Alzheimer:

Alzheimer Association - Waht is Alzheimer? [Fecha de consulta: 20/05/2018] Recuperado de: https://www.alz.org/alzheimers_disease_what_is_alzheimers.asp

Alzheimer Association - Early sings of Alzheimer [Fecha de consulta: 28/05/2018] Recuperado de: https://www.alz.org/alzheimers_disease_10_signs_of_alzheimers.asp

Alzheimer´s Disease International - The global impact of dementia [Fecha de consulta: 01/06/2018] Recuperado de: https://www.alz.co.uk/research/GlobalImpactDementia2013.pdf

15. Anexos

A continuación, se pondrán los scripts más importantes:

KeyWordRecognizer

```
using UnityEngine;
using System;
using System.Linq;
using System.Collections.Generic;
using UnityEngine.Windows.Speech;

public class KeyWordRecognizer{
    static KeyWordRecognizer _singleton;
    public static KeyWordRecognizer singleton {
        get {
            if (_singleton == null)
                _singleton = new KeyWordRecognizer();
            return _singleton;
        }
    }

    KeywordRecognizer keywordRecognizer;

    Dictionary<string, Action> keywords = new Dictionary<string, Acti
on> ();

    KeyWordRecognizer() { }

    public void RegisterKeyword(string keyword, Action onKeywordRecog
nized){
        keywords.Add (keyword, onKeywordRecognized);
        RestartRecognizer ();
    }
public void UnRegisterKeyword(string keyword){
        keywords.Remove (keyword);
        RestartRecognizer ();
    }
```

```
void RestartRecognizer(){
        if (keywordRecognizer != null)
            keywordRecognizer.Dispose ();
        if (keywords.Count == 0)
            return;

        keywordRecognizer = new KeywordRecognizer (keywords.Keys.ToAr
ray());
        keywordRecognizer.OnPhraseRecognized += OnKeywordRecognized;
        keywordRecognizer.Start ();
    }

    void OnKeywordRecognized(PhraseRecognizedEventArgs args){
        Action @delegate;
        Debug.Log ("Keyword recognized: " + args.text);
        if (keywords.TryGetValue (args.text, out @delegate))
            @delegate ();
    }

    public void Stop(){
        if (keywordRecognizer != null)
            keywordRecognizer.Stop ();
    }

    public void Resume(){
        if (keywordRecognizer != null)
            keywordRecognizer.Start();
    }
}
```

PlayPhaseManager

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

public class PlayPhaseManager : MonoBehaviour {
    [SerializeField]
    GameObject dummy;
    [SerializeField]
    GameObject label;
    [SerializeField]
    Text scoreText;

    List<HiddenObjectData> hiddenObjects = new List<HiddenObjectData>
();
    HiddenObjectData selected;

    string[] numbersToText = new string[]{
        "uno",
        "dos",
        "tres",
        "cuatro",
        "cinco",
        "seis",
        "siete",
        "ocho",
        "nueve",
        "diez",
    };

    public HiddenObject[] objectsToHide { get; private set; }

    int _score;
    public int score {
        get { return _score; }
        private set {
            _score = value;
            if (_score < 0)
                _score = 0;
            scoreText.text = string.Format("Puntos: {0}", _score);
        }
    }
```

```
void Start(){
        objectsToHide = FindObjectsOfType<HiddenObject>();
        dummy.SetActive (false);
        label.SetActive (false);
        scoreText.text = string.Empty;
    }

    public void Setup(){
        List<HiddenObject> objectsToHideCopy = new List<HiddenObject>
 (objectsToHide);

        //Avoid trying to hide more objects than there are to hide
        for (int i = 0; i < Mathf.Min(LevelManager.singleton.current.
objectsToHide, objectsToHide.Length); i++) {
            HiddenObject hiddenObject = objectsToHideCopy [Random.Ran
ge (0, objectsToHideCopy.Count)];
            GameObject dummy = CreateDummy (hiddenObject);
            GameObject label = CreateLabel (dummy, (i+1).ToString());
            HiddenObjectData data = new HiddenObjectData (hiddenObjec
t, dummy, label);
            hiddenObjects.Add(data);
            objectsToHideCopy.Remove (hiddenObject);

            KeyWordRecognizer.singleton.RegisterKeyword (numbersToTex
t [i], () => OnObjectSelected (data));
        }

        StartCoroutine(PointsCoroutine());
    }

    GameObject CreateDummy(HiddenObject hiddenObject){
        GameObject dummy = Instantiate (this.dummy);
        dummy.SetActive (true);
        dummy.transform.position = hiddenObject.meshRenderer.bounds.c
enter;
        dummy.transform.rotation = hiddenObject.transform.rotation;
        float smallestDimension = Mathf.Min (
            hiddenObject.meshRenderer.bounds.size.x,
            hiddenObject.meshRenderer.bounds.size.y,
            hiddenObject.meshRenderer.bounds.size.z
        );
        dummy.transform.localScale = Vector3.one*smallestDimension;
        hiddenObject.gameObject.SetActive (false);
        return dummy;
    }
```

```
GameObject CreateLabel(GameObject dummy, string text){
        GameObject label = Instantiate (this.label);
        label.SetActive (true);

        float distance = Mathf.Sqrt (3 * Mathf.Pow (dummy.transform.l
ocalScale.x, 2));
        Vector3 direction = (Player.singleton.transform.position -
 dummy.transform.position).normalized;
        label.transform.position = dummy.transform.position + directi
on * distance;

        label.transform.rotation = Quaternion.LookRotation (-
direction);

        label.GetComponentInChildren<Text> ().text = text;
        return label;
    }

    void OnObjectSelected(HiddenObjectData hiddenObjectData){
        if (selected!=null)
            return;

        selected = hiddenObjectData;
        Player.singleton.cameraMovement.enabled = false;
        Player.singleton.ZoomTo (hiddenObjectData.dummy);
        Catalog.singleton.Setup (hiddenObjectData.hiddenObject, ()=>O
nSelectionSuccess(hiddenObjectData));

        KeyWordRecognizer.singleton.RegisterKeyword ("atrás", ()=>Can
celSelection());
    }

    void CancelSelection(){
        Player.singleton.cameraMovement.enabled = true;
        KeyWordRecognizer.singleton.UnRegisterKeyword("atrás");
        Player.singleton.RestoreZoom();
        selected = null;
        Catalog.singleton.Hide();
    }
```

```
void OnSelectionSuccess(HiddenObjectData hiddenObjectData){
        hiddenObjectData.hiddenObject.gameObject.SetActive (true);
        Destroy (hiddenObjectData.dummy);
        Destroy (hiddenObjectData.label);
        KeyWordRecognizer.singleton.UnRegisterKeyword (numbersToText
[hiddenObjects.IndexOf (hiddenObjectData)]);
        hiddenObjectData.guessed = true;
        CancelSelection ();

        int counter = 0;
        for (int i = 0; i < hiddenObjects.Count; i++)
            if (hiddenObjects [i].guessed)
                counter += 1;
            if (counter == hiddenObjects.Count) {
                GameManager.singleton.GameOver ();
                 Player.singleton.cameraMovement.enabled = false;
        }
    }
    IEnumerator PointsCoroutine() {
        score = LevelManager.singleton.current.initialPoints;
        while (GameManager.singleton.phase == GameManager.Phase.Play)
 {
            yield return new WaitForSeconds(1);
            score -
= LevelManager.singleton.current.pointsLostPerSecond;
        }
    }
    public void ErrorDone() {
        score -= LevelManager.singleton.current.pointsLostPerError;
    }
    public void ObjectFound() {
        score += LevelManager.singleton.current.pointsPerObjectFound;
    [System.Serializable]
    class HiddenObjectData{
        public HiddenObject hiddenObject;
        public GameObject dummy;
        public GameObject label;
        public bool guessed;
        public HiddenObjectData(HiddenObject hiddenObject, GameObject
 dummy, GameObject label){
            this.hiddenObject = hiddenObject;
            this.dummy = dummy;
            this.label = label;
            guessed = false;
        }
    }
}
```

CameraMovement.cs

```
using UnityEngine;
using UnityEngine.UI;

public class CatalogItem : MonoBehaviour {
    [SerializeField]
    Image _image;
    [SerializeField]
    Text _keyword;

    public Sprite image {
        get { return _image.sprite; }
        set { _image.sprite = value; }
    }

    public string keyword {
        get { return _keyword.text; }
        set { _keyword.text = value; }
    }

    RectTransform _rectTransform;
    public RectTransform rectTransform {
        get {
            if (_rectTransform == null)
                _rectTransform = transform as RectTransform;
            return _rectTransform;
        }
    }
}
```

CameraMovementMenu.cs

```
using UnityEngine;

public class CameraMovementMenu : MonoBehaviour {
    [SerializeField]
    float sensibility;
    [SerializeField]
    float maxRotationX;
    [SerializeField]
    float maxRotationY;

    Vector3 initialRotation;

    void Awake() {
        initialRotation = transform.eulerAngles;
    }

    void Update(){
        float mouseX = Input.GetAxis ("Mouse X");
        float mouseY = Input.GetAxis ("Mouse Y");

        Vector3 rotation = transform.eulerAngles;
        rotation.x -= mouseY * sensibility * Time.deltaTime;
        rotation.x = Mathf.Clamp(rotation.x, 360+initialRotation.x -
 maxRotationX, 360+initialRotation.x + maxRotationX);
        transform.rotation = Quaternion.Euler(rotation);
    }
}
```

Catalog.cs

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

public class Catalog : MonoBehaviour {
    [SerializeField]
    CatalogItem itemPrefab;
    [SerializeField]
    GameObject catalogView;

    static Catalog _singleton;
    public static Catalog singleton {
        get {
            if (_singleton == null)
                _singleton = FindObjectOfType<Catalog>();
            return _singleton;
        }
    }

    CatalogItem[] items;
    string[] keywordIndexes = new string[]{
        "primero",
        "segundo",
        "tercero",
        "cuarto",
        "quinto",
    };
    int itemsToShow { get { return LevelManager.singleton.current.cat
alogItemsToShow; } }

    void Awake(){
        items = new CatalogItem[itemsToShow];
        for (int i = 0; i < itemsToShow; i++) {
            items [i] = Instantiate (itemPrefab);
            items [i].transform.SetParent (itemPrefab.transform.paren
t);
            items [i].rectTransform.localScale = itemPrefab.rectTrans
form.localScale;
        }
        itemPrefab.gameObject.SetActive (false);
        catalogView.SetActive (false);
     }
```

```
public void Setup(HiddenObject hiddenObject, Action onSelectionSucces
s){
        catalogView.SetActive (true);
        int random = Random.Range (0, itemsToShow);
        List<HiddenObject> objectsToHideCopy = new List<HiddenObject>
 (GameManager.singleton.playPhaseManager.objectsToHide);
        objectsToHideCopy.Remove (hiddenObject);

        for (int i = 0; i < itemsToShow; i++) {
            bool isCorrect = i == random;
            HiddenObject objectToShow = isCorrect ? hiddenObject : ob
jectsToHideCopy [Random.Range (0, objectsToHideCopy.Count)];
            objectsToHideCopy.Remove (objectToShow);

            items[i].image = objectToShow.image;
            items[i].keyword = string.IsNullOrEmpty(objectToShow.keyw
ord)?keywordIndexes[i]:objectToShow.keyword;

            KeyWordRecognizer.singleton.RegisterKeyword (items[i].key
word, () => {
                if (!isCorrect) {
                    GameManager.singleton.playPhaseManager.ErrorDone(
);
                    return;
                }

                for (int j = 0; j < itemsToShow; j++)
                    KeyWordRecognizer.singleton.UnRegisterKeyword(ite
ms[j].keyword);

                GameManager.singleton.playPhaseManager.ObjectFound();
                onSelectionSuccess();
            });
        }
    }

    public void Hide(){
        catalogView.SetActive (false);
        for (int i = 0; i < itemsToShow; i++)
            KeyWordRecognizer.singleton.UnRegisterKeyword(items[i].ke
yword);
    }
```

CatalogItem.cs

```
using UnityEngine;
using UnityEngine.UI;

public class CatalogItem : MonoBehaviour {
    [SerializeField]
    Image _image;
    [SerializeField]
    Text _keyword;

    public Sprite image {
        get { return _image.sprite; }
        set { _image.sprite = value; }
    }

    public string keyword {
        get { return _keyword.text; }
        set { _keyword.text = value; }
    }

    RectTransform _rectTransform;
    public RectTransform rectTransform {
        get {
            if (_rectTransform == null)
                _rectTransform = transform as RectTransform;
            return _rectTransform;
        }
    }
}
```

Gamemanager.cs

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;

public class GameManager : MonoBehaviour {
    public enum Phase { None = 0, Memorize, Play, GameOver }

    [SerializeField]
    GameOverMenu gameOverMenu;
    [SerializeField]
    ScreenFader fader;
    [SerializeField]
    Text countDownText;

    public Phase phase { get; private set; }

    static GameManager _singleton;
    public static GameManager singleton{
        get{
            if (_singleton == null)
                _singleton = FindObjectOfType<GameManager> ();
            return _singleton;
        }
    }

    PlayPhaseManager _playPhaseManager;
    public PlayPhaseManager playPhaseManager{
        get{
            if (_playPhaseManager == null)
                _playPhaseManager = FindObjectOfType<PlayPhaseManager
> ();
            return _playPhaseManager;
        }
    }

    void Awake() {
        fader.Fade(0, 1, () => {
            countDownText.gameObject.SetActive (false);
            Player.singleton.cameraMovement.enabled = false;
            fader.Fade(1, 0, ()=> StartCoroutine (MemorizePhase ()));
        });
    }
```

```
IEnumerator MemorizePhase(){
        countDownText.gameObject.SetActive (true);
        phase = Phase.Memorize;
        Player.singleton.cameraMovement.enabled = true;

        int seconds = LevelManager.singleton.current.memorizeTime;
        while (seconds >= 0) {
            countDownText.text = string.Format ("Tiempo: {0} seconds"
, seconds);
            yield return new WaitForSeconds (1);
            seconds-=1;
        }

        countDownText.gameObject.SetActive (false);
        fader.Fade(1, 1, ()=>{
            phase = Phase.Play;
            playPhaseManager.Setup();
            fader.Fade(1, 0);
        });
    }

    public void GameOver(){
        Debug.Log ("Game Over");
        StartCoroutine(GameOverCoroutine());

        LevelManager.singleton.current.record = playPhaseManager.scor
e;
    }

    IEnumerator GameOverCoroutine() {
        phase = Phase.GameOver;
        yield return new WaitForSeconds(2);
        fader.Fade(1, 1, gameOverMenu.Show);
    }

    void OnDestroy(){
        _singleton = null;
    }
}
```