TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Emot3D

Desarrollo de un videojuego adaptativo a las emociones del jugador

Autor

Álvaro Domínguez Aguilar

Directores

Francisco Manuel García Moreno

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Julio de 2022

Emot3D
Desarrollo de un videojuego adaptativo a las emociones del jugador

Álvaro Domínguez Aguilar

Palabras clave: diademas EEG, stress, engagement, videojuegos, accesibilidad, adaptabilidad

Resumen

La innovación en el mundo de los videojuegos crece a un ritmo exponencial en la actualidad. Cada vez son más las personas que son usuarias de esta forma de entretenimiento por lo que las empresas de la industria del videojuego buscan nuevas formas de jugar. Emot3D es un videojuego que pretende ofrecer una nueva experiencia de juego a través del uso de dispositivos que cuentan con sensores electroencefalográficos que permiten leer la actividad cerebral del jugador.

El videojuego se desarrollará mediante el motor de videojuegos Unity y los datos se obtendrán mediante uso de los dispositivos Emotiv EPOC X EEG headset. La información sobre el estado emocional de la persona se recopilará a través de la diadema EEG y sus datos se enviarán al videojuego, donde se procesarán para poder adaptar las características y jugabilidad del propio juego en función de los valores obtenidos.

Emot3D
Development of a video game adaptive to the player's emotion

Álvaro Domínguez Aguilar

Keywords: EEG headset, stress, engagement, video games, accessibility, adaptability

Abstract

Innovation in the world of video games is currently growing at an exponential rate. More and more people are using this form of entertainment, which is why companies in the video game industry are looking for new ways to play. Emot3D is a video game that aims to offer a new gaming experience through the use of devices with electroencephalographic sensors that read the player's brain activity.

The video game will be developed using the Unity game engine and the data will be obtained through the use of Emotiv EPOC X EEG headset. The information on the emotional state of the person will be collected through the EEG headset ant the data will be sent to the video game, where it will be processed in order to adapt the characteristics and gameplay of the game according to the values obtained.

Yo, **Álvaro Domínguez Aguilar**, alumno de la titulación INGENIERÍA INFORMÁTICA de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 78982198Z, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Álvaro Domínguez Aguilar

Granada a 8 de julio de 2022

D. **Francisco Manuel García Moreno**, Profesor del Área de Lenguajes y Sistemas Informáticos del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Que el presente trabajo, titulado ***Emot3D, Desarrollo de un videojuego adaptativo a las emociones del jugador***, ha sido realizado bajo su supervisión por **Álvaro Domínguez Aguilar**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 8 de julio de 2022.

El director:

Francisco Manuel García Moreno

Agradecimientos

A mi madre y hermana por mostrarme un apoyo incondicional durante todo este tiempo.

A mi padre, que me ha ayudado a realizar todas las pruebas que han sido necesarias.

A mi tutor, Francisco Manuel, por guiarme y resolverme todas las dudas que me han surgido.

Índice

1. Introducción

En este apartado se planteará una introducción al proyecto desarrollado, describiendo la motivación que ha llevado a desarrollarlo y qué objetivos se han planteado para conseguir su realización. Finalmente, se especificará la planificación temporal que se ha seguido y cuál sería su presupuesto.

1.1. Motivación

En la última década, la industria del juego ha crecido a un ritmo exponencial, no solo en cuanto a nivel económico sino también en cuanto a su innovación. Según la Asociación Española de Videojuegos [1], la facturación de videojuegos en 2021 se ha incrementado un 2,75% respecto al año anterior, con un total de 18,1 millones de jugadores en España, lo que indica el impacto que tiene en la actualidad. El avance tecnológico también ha permitido que se incremente el modo en el que las personas pueden disfrutar de los videojuegos. Este crecimiento ha impulsado a las empresas creadoras de videojuegos a desarrollar nuevas experiencias de juego para atraer a nuevos jugadores.

Para ofrecer una nueva experiencia de juego a las personas, se ha desarrollado este proyecto, Emot3D. En la actualidad, la mayoría de videojuegos del mercado tienen predefinida por parte del creador el modo en el que las personas deben experimentar el juego que han creado mediante una lista de dificultades que cambian valores del mismo según lo que ha seleccionado el jugador. Este proyecto busca llevar más allá la adaptabilidad del juego a través del uso de un dispositivo que permite obtener los datos relativos a los estados cognitivo y emocionales de una persona en tiempo real.

Debido a que no tenía claro qué proyecto desarrollar me informé sobre los trabajos propuestos por los profesores. Entre todos ellos, el que más me llamó la atención fue el propuesto por el profesor Francisco Manuel García Moreno, puesto que siempre he estado interesado en el desarrollo de videojuegos, sobre todo para plataformas de escritorio. Además, también me pareció interesante este tipo de dispositivos debido a que su uso podría ofrecer más accesibilidad, no solo en el mundo de los videojuegos, sino también sobre los aspectos más cotidianos de la vida diaria para las personas que más lo necesitan.

1.2. Objetivos

A continuación, se detalla el objetivo general del proyecto que va a ser desarrollado, así como los objetivos específicos que se han seguido para lograr alcanzarlo.

1.2.1. Objetivo general

El objetivo principal y general de este proyecto se centra en: desarrollar un videojuego complejo mediante el motor de videojuegos Unity que brinde la capacidad de alterar la jugabilidad del mismo, adaptándose a las emociones que transmite el jugador mientras se desarrolla la partida. Estas emociones se capturarán mediante el uso de sensores electroencefalográficos (EEG), colocados en la cabeza del jugador, que registran la actividad cerebral del mismo.

1.2.2. Objetivos específicos

Como objetivos específicos para lograr alcanzar el objetivo principal se plantean los siguientes:

- Indagar sobre los aspectos de los videojuegos que pueden influir en los estados emocionales y cognitivos de las personas.

- Diseñar un videojuego que plantee un desafío a las personas que lo jueguen para obtener una lectura fiable del impacto comentado en el primer objetivo. Esta dificultad no debe generar un alto nivel de estrés en el jugador para prevenir los problemas que pueda generar y mantener la atención del jugador constantemente para que no pierda el enfoque.

- Diseñar un nivel de videojuego con una variabilidad de elementos del juego suficiente, que sea altamente adaptable y configurable, para poder cambiar las dinámicas del juego en cualquier momento según las emociones del jugador.

- Estudiar la información que se obtiene sobre la actividad cerebral de una persona mediante sensores electroencefalográficos para diseñar la jugabilidad del videojuego de forma que se pueda adaptar a estos indicadores.

- Aplicar los datos relativos a los estados emocionales de las personas obtenidos a través de unos sensores electroencefalográficos para poder modificar la jugabilidad de un videojuego.

1.3. Planificación temporal

En este apartado se representa la planificación temporal del proyecto desde su comienzo a finales de marzo hasta su finalización. Para realizar esta planificación, se han hecho uso de diagramas de Gantt (Figuras 1 y 2).

1.3.1. Planificación inicial

La planificación inicial se realizó al comienzo del desarrollo del proyecto, a partir de su asignación definitiva a finales de marzo. En esta planificación se muestran las actividades que se han pretendido realizar desde el comienzo del proyecto hasta su finalización y correspondiente entrega.

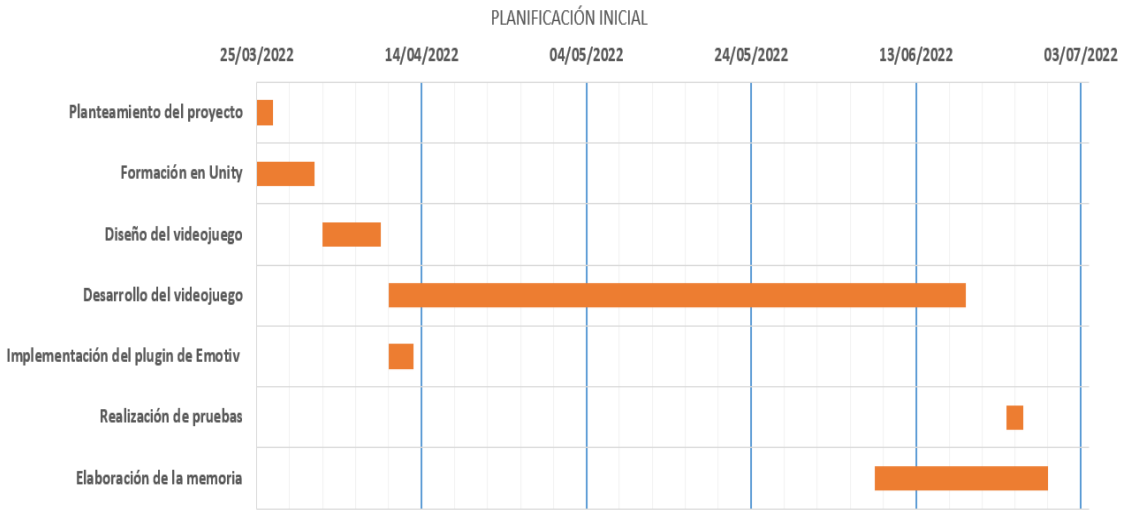

Figura 1. Diagrama de Gantt de la planificación inicial.

En primer lugar, se planteó cómo debería desarrollarse el proyecto y se realizó una pequeña formación en el motor de videojuegos Unity que duraría aproximadamente una semana. Una vez que se hubiese realizado el diseño del videojuego se procedería a su desarrollo. En las fases finales del desarrollo se realizarían las pruebas de validación correspondientes y se procedería a completar la memoria.

1.3.2. Planificación real

La planificación real muestra el tiempo real que se ha necesitado en cada actividad para desarrollar el proyecto. Esta planificación se ha realizado el mismo día en el que este proyecto ha sido entregado.

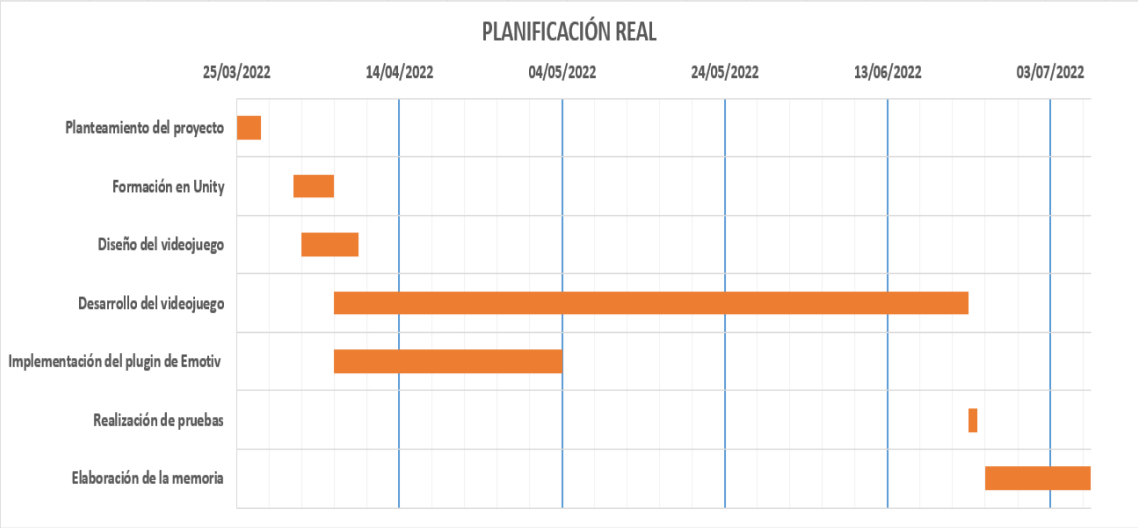

Figura 2. Diagrama de Gantt de la planificación real.

En la planificación real se observa cómo el desarrollo del proyecto comenzó un poco más tarde de lo planificado, lo que obligó a retrasar un poco cada una de las actividades y recortar el tiempo dedicado a otras, por lo que fue necesario acoplar varias actividades y realizarlas a la vez.

La actividad a la que más tiempo se dedicó fue el propio desarrollo del videojuego y la implementación del plugin de Emotiv. Este gran incremento de tiempo para la implementación del plugin se explicará en un apartado posterior.

1.4. Presupuesto

Para llevar a cabo el proyecto es necesario contar con un ordenador con las características mínimas para desarrollar el videojuego de modo fluido, así como el dispositivo necesario para obtener los datos sobre los estados cognitivos y emocionales necesarios para poder adaptar la jugabilidad del juego. El precio del ordenador se ha obtenido mediante la media de precios del mercado de ordenadores con las características mínimas requeridas y el precio del dispositivo se ha obtenido a través del ofrecido en la página web oficial de compra [2].

Debido a que el proyecto se debe completar en una duración de 3 meses, será necesario la contratación de una persona con conocimiento en el diseño y desarrollo de videojuegos para el motor de videojuegos Unity. Este contrato tendrá un valor semejante al de la media del mercado.

Recurso	Unidades	Costo
Ordenador	1	895 €
EMOTIV EPOC X Channel Mobile Brainwear	1	849 €
Desarrollador	1	2350 €/mes * 3 meses = 7050 €
Total		**8794 €**

Tabla 1. Presupuesto estimado del proyecto.

2. Proyectos y trabajos relacionados

En este apartado se presentan los trabajos y proyectos que han sido desarrollados y se relacionan con el tema sobre el que se trata en este proyecto.

2.1. An Analysis of Game-Related Emotions Using EMOTIV EPOC [3]

El objetivo de este experimento, desarrollado en 2018, era analizar los cambios que se producen en las personas cuando jugaban a un videojuego, identificando cuáles eran las emociones que más impactos sufrían según los elementos que constituían el juego. La recopilación de estos estados emocionales se llevó a cabo mediante el uso de una diadema EEG EMOTIV EPOC+.

Para realizar este experimento se registraron los datos de 15 hombres de edades entre 23 y 28 años. Este experimento se dividió en 7 sesiones donde el sujeto era informado con anterioridad sobre la acción que debía realizar:

1) Los primeros experimentos consistían en medir las lecturas del cerebro cuando el sujeto se encontraba meditando, con los ojos cerrados, y cuando participaba en un pequeño juego en equipos con una actividad cerebral moderada.

2) En segundo lugar, los participantes jugaron a dos videojuegos: Rocket League, un juego que combina el fútbol con vehículos, y Super Bomberman, un juego de acción. Todos los participantes jugaron 2 partidas a cada uno de los juegos; la primera partida contra jugadores controlados por el ordenador y la segunda partida jugando con jugadores reales.

3) La última sesión, consistía en ver una partida de 5 minutos de Rocket League jugada por jugadores profesionales.

La conclusión que se obtiene tras realizar este experimento es que los videojuegos causan cambios en la actividad cerebral, siendo estos cambios mayores en algunas partes específicas. La frustración y la excitación de las personas aumentó cuando jugaban con otras personas; y el aburrimiento creció al ver la partida profesional. Sin embargo, el análisis de los datos obtenidos por el dispositivo no era suficiente para medir el impacto que ha tenido el experimento.

2.2. The Emotiv Mind: Investigating the accuracy of the Emotiv EPOC in identifying emotions and its use in an Intelligent Tutoring System [4]

Este estudio fue realizado por el Departamento de Ciencias de la Computación e Ingeniería Software de la Universidad de Canterbury, en 2013, y tenía como objetivo analizar la precisión de los dispositivos Emotiv EPOC a través de tres experimentos:

1) El primer caso de estudio consistía en comparar los datos que se obtenían a través del dispositivo con la información que ofrecía los propios participantes.

2) El segundo caso consistía en un estudio más perfeccionado del primer caso de estudio.

3) Por último, se realizaron pruebas del dispositivo con un Sistema de Tutorías Inteligente.

En el primer experimento, 10 participantes jugaron a una partida de Tetris de 5 minutos, se visualizaron varias imágenes y se completaron cuestionarios relacionados. Debido a errores con el dispositivo, sólo se pudieron obtener los datos de 3 participantes, por lo que los resultados finalizaron con una valoración inconclusa, aunque se pudo observar que existía una relación entre los estados de meditación y aburrimiento captados por el dispositivo con los reportados por los participantes.

En el segundo experimento, se perfeccionó el anterior reduciendo las imágenes y mejorando las respuestas del cuestionario. Debido al ruido provocado por los sensores, los resultados de este experimento tampoco fueron fiables.

En el último experimento, 11 participantes completaron un curso en base de datos mediante un sistema de tutoría. De igual modo que en los otros experimentos, no se encontró ninguna diferencia significativa entre los eventos que se habían predefinido debido a una incorrecta lectura de valores o a la gran varianza entre las respuestas emocionales de los participantes para una misma categoría.

En este estudio se llegó a la conclusión de que el dispositivo Emotiv EPOC no es fiable en la lectura de las emociones debido a que no se encontraron relaciones entre el estado emocional reportado por los participantes y los valores capturados por el dispositivo. Sin embargo, se llegaron a estos resultados debido a la lectura errónea de los datos a causa de la falta de hidratación de los sensores tras el paso del tiempo y el software y modelo de la diadema EEG no eran consistentes en el recibo y envío de estos datos, al tratarse de los primeros modelos creados por Emotiv.

2.3. Conclusiones

Como se ha visto, este campo de investigación es muy reciente y queda mucho por avanzar en el uso de este tipo de dispositivos, que son mucho menos intrusivos que los tradicionales cascos de EEG que necesitan de la conexión de múltiples cables, aparatos específicos que lean las señales que transmiten, etc.

Así que todo esfuerzo en la línea de continuar con la investigación puede contribuir a arrojar luz en este tipo de aplicaciones que podrían beneficiar a los futuros jugadores, donde cada vez más empieza a vislumbrase indicios de que vamos hacia una tecnología de realidad virtual y metaverso, donde podrían jugar un papel muy interesante este tipo de dispositivos EEG para controlar los entornos interactivos.

3. Tecnologías utilizadas

En este apartado se describen las tecnologías que han sido utilizadas para realizar el desarrollo del proyecto y se comentan brevemente cada una de las características con las que cuentan.

3.1. Unity

Unity es un motor de videojuegos, lanzado en 2005 y desarrollado por la empresa Unity Technologies [5]. Unity se encuentra disponible para los sistemas operativos principales: Microsoft Windows, Linux y Mac OS.

La principal virtud de Unity es que permite desarrollar videojuegos en diversidad de plataformas: webs, ordenadores portátiles y de sobremesa, consolas, dispositivos móviles y dispositivos de realidad virtual. Este motor de videojuegos permite diseñar videojuegos tanto en una perspectiva tridimensional (3D) como en una perspectiva bidimensional (2D). Aunque el principal objetivo en Unity es el desarrollo de videojuegos, también se es utilizado en la actualidad en otros ámbitos profesionales como películas o arquitectura.

Unity cuenta con diferentes herramientas dentro del propio editor del motor de videojuegos además de la denominada Unity Asset Store [6], donde los usuarios pueden crear y subir herramientas, utilidades y recursos como modelos 3D o texturas, así como descargar estos recursos creados por otros usuarios o por la propia empresa desarrolladora. Se ha utilizado la versión 2020.3.33f1 de Unity para el desarrollo de este proyecto.

3.2. EMOTIV EPOC X 14 Channel Mobile Brainwear

Emotiv EPOC X [7] es una diadema EEG desarrollada por la empresa Emotiv, diseñada para obtener datos del cerebro humano para realizar investigaciones.

El dispositivo cuenta con 14 sensores electroencefalográficos que permiten medir valores obtenidos a partir de los impulsos eléctricos obtenidos por el mismo: comandos mentales, métricas de rendimiento mental y expresiones faciales. Además, cuenta con sensores de movimiento, giroscopio y acelerómetro, para medir la posición y orientación de la cabeza del usuario.

El dispositivo puede conectarse a un ordenador de forma inalámbrica, a través de la tecnología Bluetooth mediante un receptor USB.

Figura 3. EMOTIV EPOC X Headset.

3.3. Emotiv Unity Plugin

Emotiv Unity Plugin [8] es el plugin oficial desarrollado por Emotiv que contiene el código necesario para conectar los servicios de la API de Emotiv y los sensores de los distintos electrodos al motor de videojuegos Unity.

El plugin cuenta con diferentes clases que permiten realizar funcionalidades relacionadas con las diademas EEG: crear sesiones de grabaciones de datos, inyectar marcadores, suscribirse a flujos de datos, crear perfiles de datos y realizar un entrenamiento mediante expresiones faciales y pensamientos. Para hacer uso de todas estas funcionalidades de los electrodos, el plugin cuenta con una clase que actúa como interfaz y contiene todos los métodos necesarios.

3.4. Emotiv Launcher

Emotiv Launcher [9] es la aplicación oficial de Emotiv y es necesario para poder conectar el dispositivo Emotiv EPOC X. Entre las características más importantes de esta aplicación se encuentra la capacidad de crear un dispositivo virtual y cuenta con un menú para modificar distintos estados y flujos de datos del dispositivo, permitiendo la simulación de un dispositivo real en caso de no contar con uno.

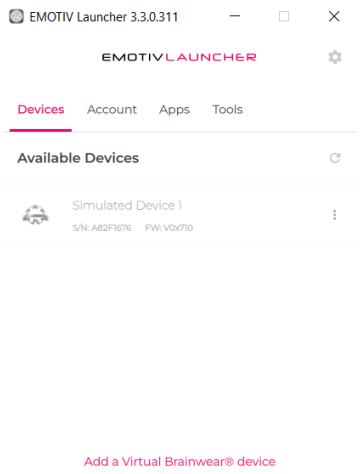

Figura 4. Emotiv Launcher.

Esta aplicación actúa como central para otras aplicaciones de diversas funcionalidades desarrolladas por Emotiv. Para el desarrollo del proyecto se ha hecho uso de la aplicación denominada EmotivBCI debido a que permite la visualización de los flujos de datos que transmite el dispositivo además de poder crear varios perfiles de entrenamiento que permiten guardar el estado del dispositivo en caso de realizarse diferentes pruebas. Cada uno de estos perfiles de entrenamiento almacenan los conjuntos de datos de comandos mentales y expresiones faciales, permitiendo la prueba de diferentes estrategias de entrenamiento o personalizarlos para el uso de diferentes aplicaciones sin necesidad de sobrescribirlos.

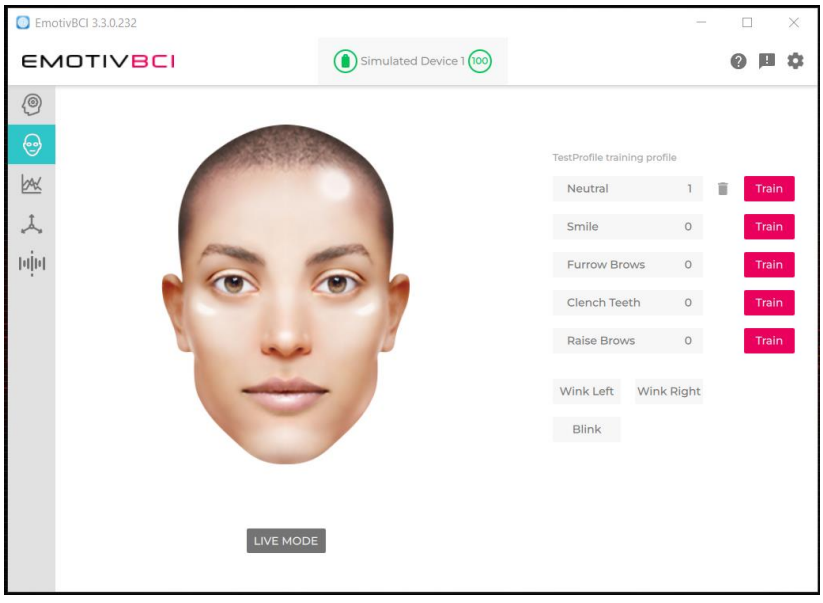

Figura 5. EmotivBCI.

Sin embargo, su mayor desventaja es que para hacer uso de las otras aplicaciones u obtener datos en crudo sobre los datos transmitidos por el dispositivo es necesario realizar el pago de una licencia, por lo que se reduce la capacidad para analizar este flujo de datos y sacar conclusiones correctas. No obstante, en su versión gratuita obtenemos datos que ya han sido procesados por Emotiv a partir de los datos en crudo estableciendo así:

a) Performance metrics: Miden el rendimiento mental obtenido a partir de la actividad mental del usuario mediante un valor entero. Provee de seis medidas básicas:

1) Engagement: Atención que experimenta la persona hacia los estímulos relevantes para completar la tarea que tiene como objetivo.

2) Stress: Medida de la comodidad de la persona respecto al desafío actual. Un alto nivel puede suponer riesgos de salud mientras que un nivel más bajo puede incrementar la productividad de una tarea.

3) Interest: Grado de atracción hacia la tarea que la persona se encuentra realizando. Un valor bajo, indica una fuerte aversión hacia la tarea, mientras que un valor alto muestra atracción para completarla.

4) Excitement: Sensación de excitación fisiológica respecto a un valor positivo. Se caracteriza por la activación de diferentes respuestas fisiológicas como el ensanchamiento de las pupilas o el aumento del ritmo cardíaco.

5) Focus: Medida de la profundidad de atención a una tarea y frecuencia en la que se produce el cambio de tareas. Un valor bajo indica poca concentración y, por lo tanto, un alto nivel de distracción.

6) Relaxation: Capacidad de la persona para poder recuperarse de un estado de concentración intensa.

b) Facial expressions: Detecta las señales eléctricas que emiten los músculos faciales y de los ojos. Puede detectar las expresiones de sonreír, fruncir las cejas, apretar los dientes, levantar las cejas, guiñar ambos ojos y parpadear.

c) **Mental commands:** Algoritmo de reconocimiento de patrones. Una vez que se ha entrenado un estado neutral, el sistema aprende a reconocer la actividad cerebral de un comando comparándola con la actividad captada en el estado neutral. Mientras mayor sea el número de repeticiones del entrenamiento de un comando mental, mayor será la fiabilidad del sistema para detectar el patrón. Los comandos mentales están asociados a los movimientos de un cubo en un espacio vacío: Empujar, Atraer, Izquierda, Derecha, Arriba, Abajo, Rotar y Desaparecer.

d) **Motions sensors:** Datos relativos a la posición y orientación de la diadema EEG mediante el uso de los valores obtenidos por un magnetómetro, un acelerómetro y vectores de rotación.

4. Propuesta del proyecto

En este apartado se expondrá la arquitectura del videojuego desarrollado, detallando cada uno de los componentes y mostrando el funcionamiento de las partes más relevantes.

4.1. Descripción

Emot3D se ha diseñado como un juego de plataformas en 3D cuya jugabilidad varía según el estado emocional del jugador mientras se encuentra equipado con la diadema Emotiv EPOC X. Las características del videojuego y la capacidad que tiene el jugador para experimentarlo son las siguientes:

a) El jugador puede controlar su avatar mediante teclado y ratón. El movimiento del jugador se limita a moverse en un terreno plano o poco inclinado en las cuatro direcciones disponibles. Para sortear obstáculos que el jugador no puede atravesar, dispone de la capacidad de realizar un salto en dirección vertical en la que se encuentra con la posibilidad de realizar otro salto cuando se encuentra saltando.

b) El videojuego cuenta con avatares enemigos que se encuentran dispersos por cada uno de los niveles. Estos enemigos tienen la capacidad de moverse por la escena y atacar al jugador.

c) Los niveles están compuestos por objetos estáticos, objetos con la capacidad de desplazarse o rotar y objetos que interactúan con el jugador.

d) El videojuego cuenta con un sistema de salud y concurrencia. Este sistema se puede ver alterado por los distintos objetos que se encuentran colocados en el nivel.

e) Existe un sistema de penalizaciones que se aplican al jugador y alteran sus propias características. Estas penalizaciones se pueden aplicar mediante la interacción del jugador con los distintos objetos que conforman la escena.

f) El videojuego recibe y procesa los datos obtenidos por la diadema EEG para modificar la jugabilidad del propio juego, añadiendo de ese modo un nivel de dificultad que va variando según los estados emocionales del jugador.

Cada uno de las diferentes funcionales del videojuego se detallan con mayor profundidad en los apartados que se encuentran a continuación.

4.2. Arquitectura del sistema

El sistema se ha diseñado mediante un modelo de descomposición modular. Debido a que el videojuego cuenta con diferentes mecánicas que no están relacionadas entre sí, se ha necesitado desarrollar un amplio número de clases con funcionalidades muy específicas. Este hecho ha provocado un aumento de la complejidad del sistema, por lo que se ha dividido el código del sistema en diferentes *módulos* o *paquetes*, agrupando clases que comparten una funcionalidad específica, decrementando la complejidad del mismo.

Los componentes de cada sistema se explicarán mediante diagramas de clases mientras que el funcionamiento de la adaptabilidad del videojuego se expondrá mediante el uso de diagramas de secuencia.

4.2.1. Diagrama de clases

El código del videojuego se ha dividido en cinco módulos: Controller, GamePlay, Items, Manager y UI. Para representar cada uno de los módulos que conforman el sistema, se ha usado un diagrama de clases que contiene todas las asociaciones entre los diferentes componentes. A continuación, se pasa a detallar cada módulo:

4.2.1.1. CONTROLLER

El módulo Controller se encuentra compuesto por las clases que controlan el movimiento y el comportamiento de los elementos de la escena que cuentan con esta capacidad: avatar del jugador, enemigos, cámara y proyectiles. Las características de estos elementos se ven afectadas por el nivel de estrés y *engagement* del jugador: velocidad del jugador y enemigos del nivel, interacciones de daños y penalizaciones que sufre el jugador y movimiento de las plataformas y trampas situadas en el nivel.

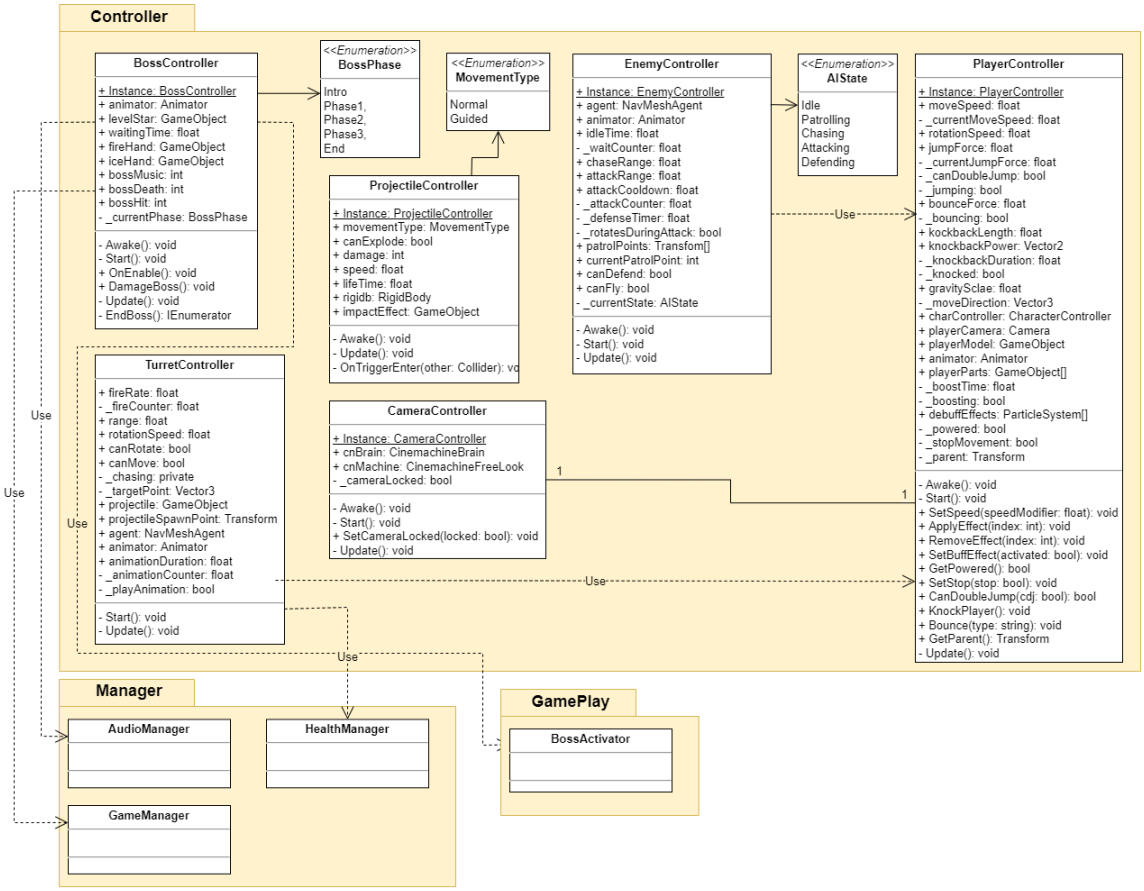

Figura 6. Diagrama de clases del módulo Controller.

a) BossController

La clase BossController se encarga de controlar las animaciones del jefe final del nivel cuando el jugador llega a la zona donde se encuentra. Cuando el jugador golpea uno de los puntos designados para recibir daño del jefe final, se cambia la animación de ataque que realiza. Una vez que el jefe final ha sido derrotado tras ser golpeado un total de 3 veces, este desaparece de la escena revelando la estrella necesaria para finalizar y completar el nivel en el que se encuentra el jugador.

b) CameraController

La cámara se ha desarrollado utilizando el paquete Cinemachine [10], desarrollado por Unity, en lugar de la cámara predeterminada que viene incluida en el propio editor. Este paquete permite crear una cámara más inteligente con una mayor opción de personalización y sencillez de código.

La cámara está posicionada detrás del avatar del jugador para ofrecer una perspectiva en tercera persona, con la opción de fijar la cámara en su eje horizontal para controlar la capacidad de visión del jugador.

c) EnemyController

Esta clase controla el comportamiento de los enemigos que se encuentran situados por la zona de juego. Para el modelado del comportamiento se ha hecho uso de una sencilla inteligencia artificial que incluye un área de navegación, denominada NavMesh [11], que permite crear una zona donde los agentes pueden circular, aproximando las superficies estáticas de los objetos que se encuentran incluidos en la escena. Cada agente cuenta con cinco comportamientos distintos:

1) **Idle:** Comportamiento predeterminado del agente. Cuando un agente se encuentra en estado *Idle*, se queda parado en la posición en la que se encuentra actualmente durante un tiempo predefinido, observando el entorno en busca del jugador.

2) **Patrolling:** Cada agente cuenta con una lista de puntos en dos dimensiones por los que puede navegar. Cuando un agente llega a la posición que le corresponde, cambia al comportamiento *Idle* y cuando finaliza el tiempo de espera, vuelve a avanzar hasta la siguiente posición en la lista.

3) **Chasing:** Cuando el jugador entra en el rango de detección del agente, este comienza a perseguirlo por el área de navegación de la escena, previamente definido. Si el jugador sale del área de detección del agente o se mueve a una zona que no forma parte del área de navegación, el agente deja de seguirlo y vuelve al estado *Patrolling*.

4) **Attacking:** Este comportamiento se activa cuando el jugador entra en el rango de ataque del agente. Cuando el jugador entra en este rango, el agente ejecuta la animación completa de ataque. Una vez que ha finalizado la animación, el agente espera un tiempo predefinido para volver a realizar la animación, en caso de que el jugador siga en el rango de ataque. En caso contrario, el agente vuelve al estado en el que se encontraba anteriormente.

5) **Defending:** El comportamiento de *Defending* solo se encuentra disponible para algunos de los agentes del juego. Este comportamiento se activa una vez ha finalizado la animación de ataque, en el que durante otro tiempo predeterminado ejecuta una animación en el que el agente se vuelve invulnerable al daño que puede recibir por parte del jugador.

d) PlayerController

La clase PlayerController cuenta con el modo en el que el jugador puede moverse por el juego y cómo interactúa con los objectos de la escena.

El jugador puede moverse en las cuatro direcciones: hacia delante, hacia atrás, izquierda y derecha, además de realizar un salto vertical que se ve afectado por la capacidad de salto del jugador y por la gravedad de la escena. En casos especiales, el jugador puede realizar un salto doble en el que mientras está en el aire, pudiendo volver a saltar con una potencia de salto un poco menor que la del salto inicial.

```
float yStore = moveDirection.y;

// Player movement
moveDirection = (transform.forward * Input.GetAxisRaw("Vertical")) + (transform.right * Input.GetAxisRaw("Horizontal"));
moveDirection.Normalize();
moveDirection = moveDirection * _currentMoveSpeed;
moveDirection.y = yStore;

// Player is on the ground => can jump
if (charController.isGrounded) {
    moveDirection.y = -jumpForce;

    // Double jump
    if (Input.GetButtonDown("Jump")) {
        moveDirection.y = _currentJumpForce;
        _jumping = true;
    }
} else if (_canDoubleJump && _jumping) {
    if (Input.GetButtonDown("Jump")) {
        moveDirection.y = _currentJumpForce / 1.25f;
        _jumping = false;
    }
}
```

Figura 7. Función de movimiento básico del jugador.

Esta clase también controla la interacción del jugador con algunos objetos en la escena: potenciadores, mecanismos y trampas, modificando el comportamiento del movimiento del avatar del jugador.

e) ProjectileController

Esta clase controla el comportamiento de los proyectiles que se instancian en el juego. Los proyectiles solo pueden ser creados si un enemigo que puede atacar a distancia ejecuta la animación de ataque cuando el jugador entra en su rango de ataque.

Los proyectiles que se crean pueden ser de dos tipos: normales y guiados. Los proyectiles normales avanzan en línea recta desde la posición en la que se han creado hasta la posición del jugador y los proyectiles guiados siguen al jugador mientras este se mueve por la escena.

Ambos tipos de proyectiles tienen la capacidad de generar una explosión, que se produce cuando el proyectil impacta con un objeto de la escena o el propio avatar del jugador, generando un área a su alrededor donde el jugador recibe daño.

Para evitar la sobrecarga de objetos en caso de que un proyectil no impacte sobre el jugador o con otro objeto en la escena y se pierda de forma indefinida, cada proyectil cuenta con un tiempo de vida en el que son destruidos cuando este se ha agotado.

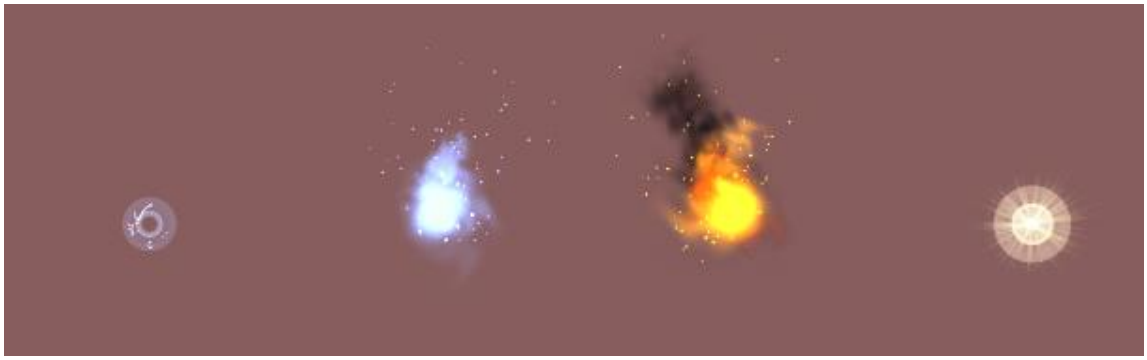

Figura 8. Proyectiles del juego.

f) TurretController

La clase TurretController controla a un tipo específico de enemigo que tiene la capacidad de atacar a distancia. De igual modo que el enemigo común, posee un agente que permite su movimiento por la escena del juego con un comportamiento similar al descrito anteriormente, exceptuando los estados de *Patrolling* y *Defending*.

Este enemigo tiene un tiempo de enfriamiento que limita el tiempo que debe esperar para poder crear un proyectil nuevo y así evitar la sobrecarga de objetos de la escena.

4.2.1.2. GAMEPLAY

El módulo GamePlay está formado por las clases que gestionan los comportamientos de los elementos del juego. Debido a que cada clase gestiona un comportamiento muy específico, los componentes de este módulo son completamente independientes entre ellos.

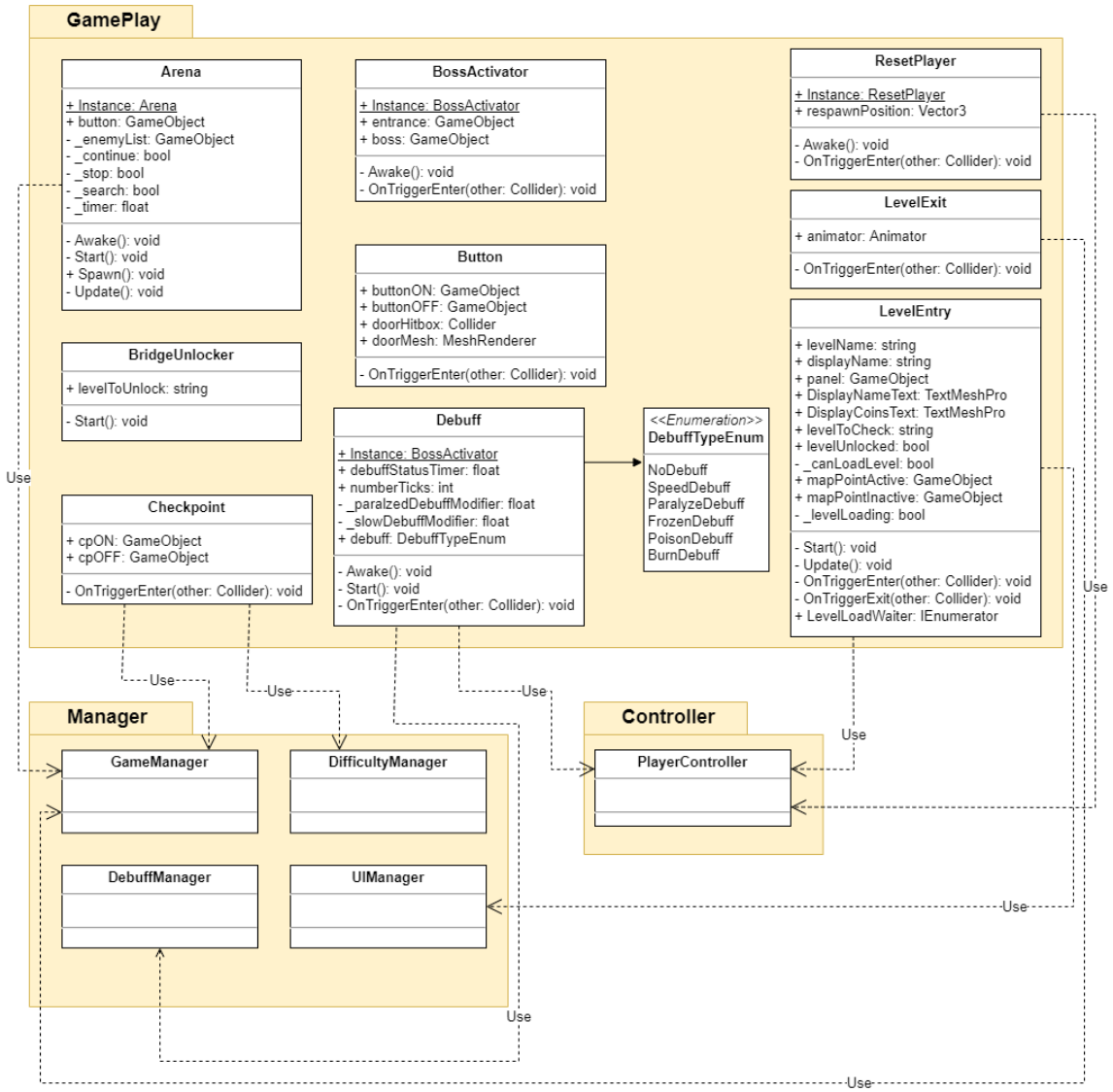

Figura 9. Diagrama de clases del módulo GamePlay.

a) Arena

La Arena es una zona especial del nivel del juego que contiene una lista de enemigos específica y se encuentra cerrada en todas las direcciones.

Cuando el jugador consigue derrotar a todos los enemigos que se encuentran en la zona se genera un botón que, al pulsarlo, permite desbloquear la salida de la Arena. En caso de que la vida del jugador llegue a cero y sea derrotado, los enemigos dentro de la Arena vuelven a generarse.

b) BossActivator

Una vez que el jugador interactúa con esta zona, hace aparecer al jefe final del nivel, además de destruir la zona que comunica el resto del nivel con la zona del jefe final.

c) BridgeUnlocker

Esta clase desbloquea el puente que uno dos mundos en el nivel de selección de niveles cuando el jugador ha completado todos los niveles que forman el mundo en el que se encuentran.

d) Button

El botón es un mecanismo que está asociado a una puerta del nivel. Cuando el botón es presionado por el jugador, la puerta a la que está asociada se abre, dejando paso al jugador.

Figura 10. Botón.

e) Checkpoint

El checkpoint permite guardar la posición del jugador. Cuando el jugador interactúa con este checkpoint y se activa, el resto de checkpoints de la escena por los que ha pasado el jugador se desactivan, en caso de que haya sido activado alguno de ellos.

Si el jugador ha activado algún checkpoint de la escena y su vida llega a cero y es destruido, vuelve a aparecer en la posición del checkpoint que ha activado tras esperar un tiempo de reaparición. En caso de no haber activado ningún checkpoint del nivel, este reaparece en la posición inicial en la que comenzó el nivel.

Figura 11. Checkpoint desactivado / activado

f) Debuff

La clase debuff controla los efectos que se produce cuando un objeto o enemigo de la escena interactúa con el avatar jugador. Cuando se produce esta interacción, se genera un efecto visual alrededor del jugador que indica la clase de penalización que se le ha aplicado. En el juego existen cinco tipos de penalizaciones que puede sufrir el jugador durante un tiempo predefinido:

1) **SpeedDebuff:** Reduce la velocidad de movimiento del jugador en una cantidad predeterminada. Esta reducción debe ser menor que la mitad de velocidad inicial.

2) **ParalyzedDebuff:** De igual modo que la penalización anterior, reduce a la mitad la velocidad de movimiento del jugador.

3) **FrozenDebuff:** Inmoviliza al jugador durante una cantidad de tiempo, reduciendo su velocidad de movimiento a cero, impidiendo que el avatar del jugador pueda moverse hasta que se haya agotado el tiempo.

4) **PoisonDebuff:** Reduce la salud del jugador en intervalos constantes de tiempo.

5) **BurnDebuff:** De igual modo que la penalización anterior, también reduce la salud del enemigo en intervalos constantes de tiempo.

g) LevelEntry

Muestra información relativa a los niveles que se encuentran en el nivel de selección de niveles. Cada nivel tiene asociado un panel de información que muestra el nombre del nivel y la cantidad de monedas que ha adquirido el jugador al completar el nivel. Si el jugador no ha completado el nivel, esta cantidad mostrará un valor indefinido.

La primera vez que se inicia el juego, todos los niveles del juego se marcan como bloqueados, excepto el primer nivel. Una vez que se ha completado el nivel, se marca como completado y el siguiente nivel en la lista de niveles se marca como desbloqueado. Cada vez que el jugador interactúa con la entrada, se realiza una carga de la escena que contiene el nivel que ha seleccionado el jugador.

h) LevelExit

Una vez que el jugador completa un nivel e interactúa con la estrella que se encuentra al final del mismo, se activa esta clase. Tras activarse, marca el nivel actual en el que se encuentra el jugador como completado e informa sobre el número de monedas del nivel que ha recogido. Una vez que ha finalizado este procedimiento, retorna al jugador al nivel de selección de niveles.

i) ResetPlayer

Esta clase se realiza la asignación del punto de reaparición del jugador en caso de que el avatar sea destruido.

4.2.1.3. ITEMS

Los *ítems* son objetos que se encuentran colocados sobre el terreno de juego o pueden ser soltados por los enemigos cuando estos son derrotados y han sido destruidos. Una vez que el jugador interactúa con estos objetos, recibe bonificaciones según el tipo de objeto del que se trate, pudiendo ser de tres tipos:

1) **Monedas (CoinPickup):** Es la divisa principal del juego. Cada nivel del juego cuenta con un número específico de monedas y se van sumando al contador de monedas del nivel cada vez que el jugador interactúa con uno de estos objetos.

2) **Corazones (HealthPickup):** Los corazones restauran la salud del jugador. Cada corazón puede curar una cantidad de 1 de salud o restaurar por completo la salud del jugador, sin importar el valor en el que se encuentra actualmente. En caso de que la salud del jugador se encuentre al máximo, esta interacción no produce ningún efecto.

3) **Potenciador de electricidad (ElectricityPickup):** Este objeto provee al jugador de un nuevo modo de dañar a los enemigos que no se ven afectados por los ataques normales del jugador. Al contrario que los objetos anteriores, este potenciador cuenta con un contador de tiempo, eliminándose su bonificación cuando este se ha agotado.

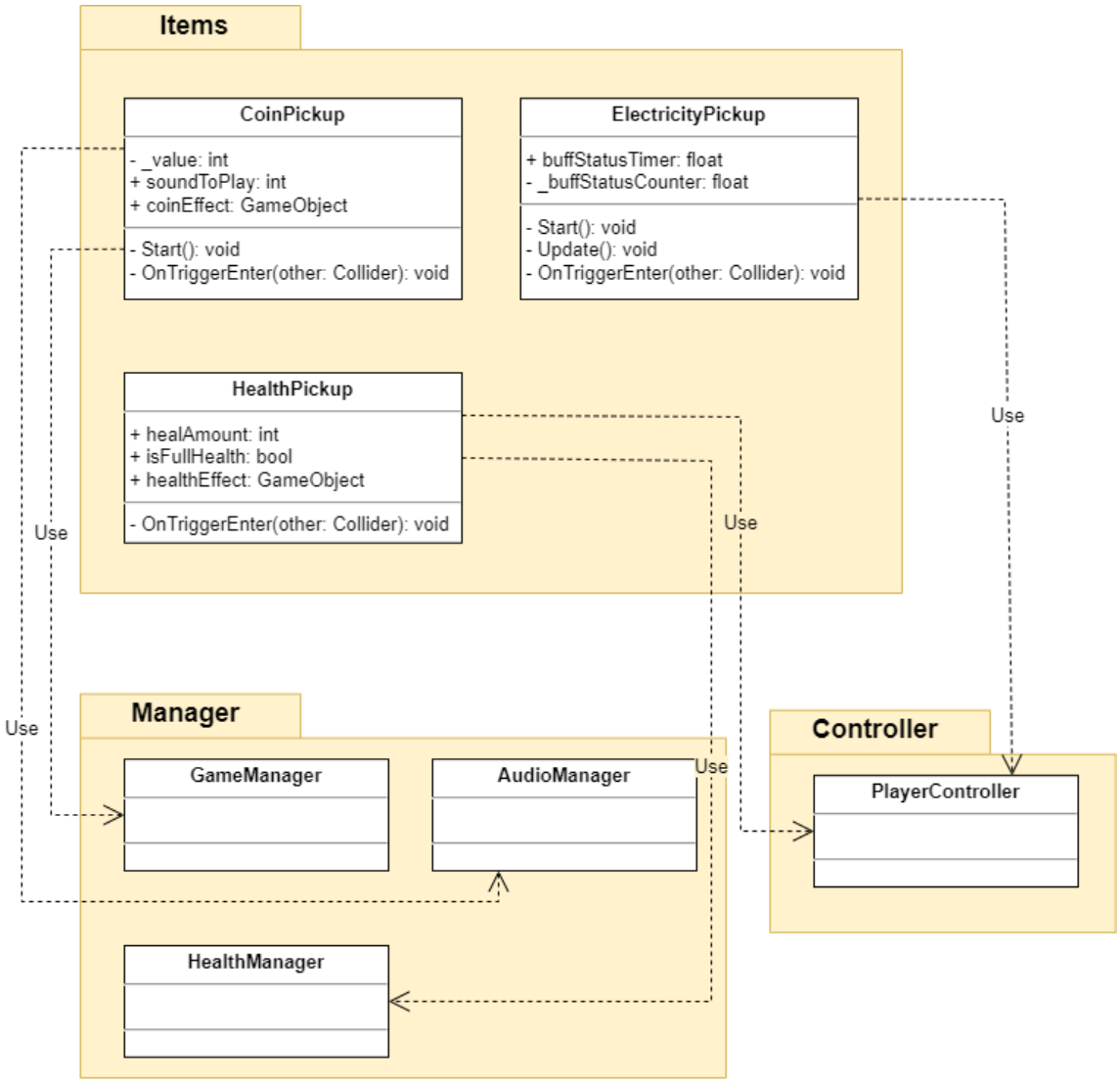

Figura 12. Diagrama de clase del módulo Items.

4.2.1.4. MANAGER

Las clases del módulo Manager gestionan las funcionalidades principales del juego y como las clases de los diferentes módulos interactúan entre ellos.

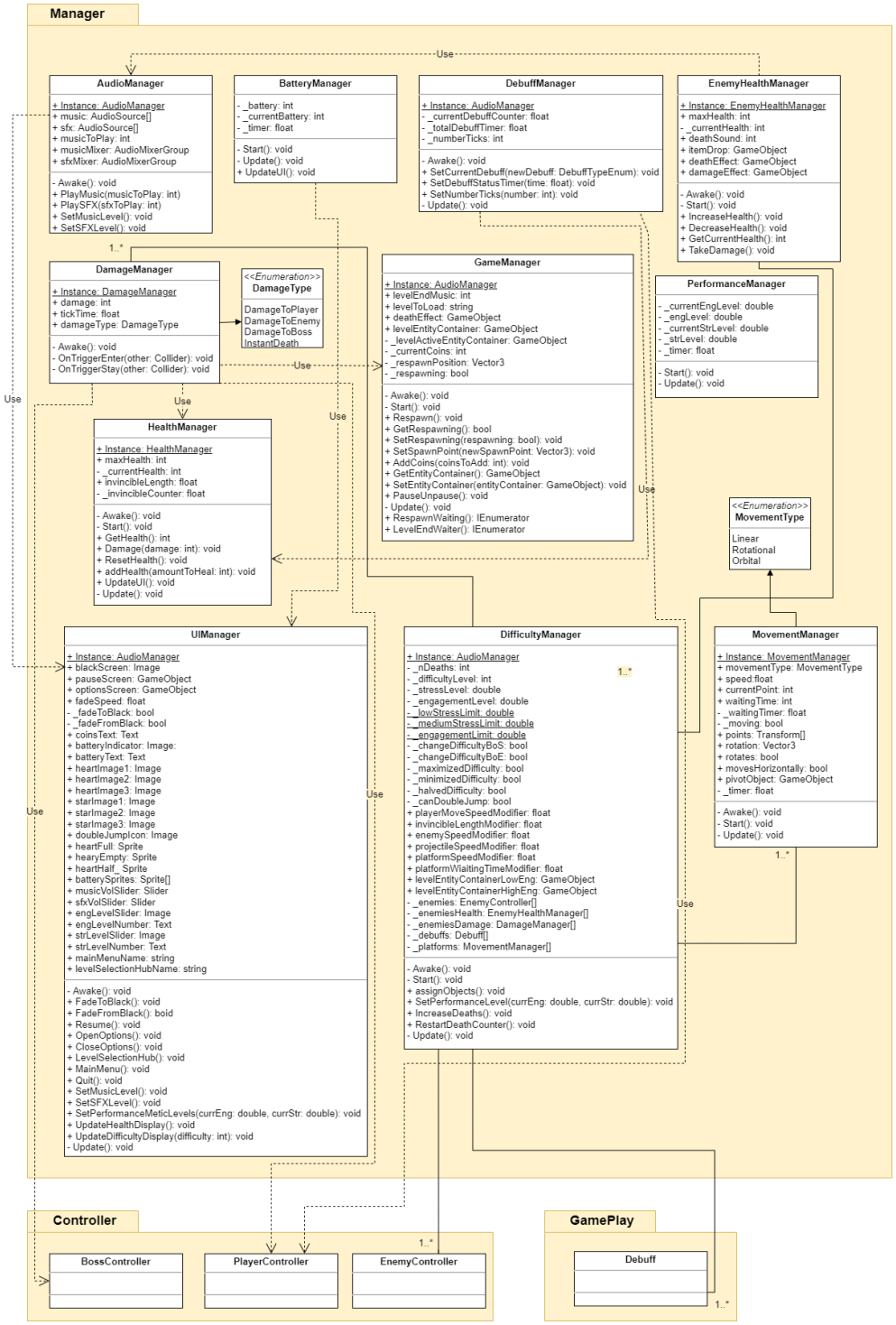

Figura 13. Diagrama de clases del módulo Manager.[1]

[1] Debido a que la clase GameManager interactúa con la mayoría de clases del videojuego, no se ha indicado las asociaciones de esta para aumentar la claridad del diagrama.

a) AudioManager

Esta clase controla el nivel de música y efectos especiales del juego, así como la música que se reproducirá de fondo mientras el jugador juega a un nivel y los efectos de sonido que emiten los distintos objetos de la escena.

b) BatteryManager

Debido a que el plugin de Emotiv cuenta con un método para obtener el valor de la batería de la diadema, sólo es necesario realizar una llamada a dicho método y almacenar el valor devuelto en una variable.

Para poder realizar la llamada a este método, en primer lugar, es necesario suscribirse al flujo de datos correspondiente. Debido a que la tasa de frecuencia del método Update() de Unity es mayor que la de la diadema, ha sido necesario tener un contador que limita cuando se puede realizar la llamada al método para que no devuelva un valor indefinido. Una vez que se ha obtenido el valor, se actualiza la interfaz de usuario con el valor de batería correspondiente.

```
void Update()
{
    _timer += Time.deltaTime;

    if (_timer >= 1f) {
        battery = ((int) DataStreamManager.Instance.Battery());

        if (_currentBattery != battery && battery > 0) {
            _currentBattery = battery;
            UpdateUI();
        } else
            _timer = 0f;
    }
}
```

Figura 14. Método Update().

c) DamageManager

Esta clase gestiona el daño que se produce en el juego. Este daño puede ser de dos tipos: daño al jugador y daño a los enemigos. Cuando un jugador golpea a un enemigo o viceversa, se reduce la vida correspondiente en una cantidad determinada.

d) DebuffManager

Los *debuff* son penalizaciones en las características del jugador que se producen cuando este interactúa con un enemigo u objeto de la escena que puede aplicarlo. Esta clase se encarga de restaurar las características originales del jugador y ocultar los efectos visuales alrededor de este, una vez que el tiempo de penalización ha finalizado.

e) DifficultyManager

Esta clase controla la variabilidad del juego y su dificultad en función de los valores de *stress* y *engagement*[2] devueltos por la diadema EEG. El modo en el que se realizan estas modificaciones se detallará a continuación en los diagramas de secuencia correspondientes.

Esta clase también permite controlar el número de muertes que sufre el jugador mientras se encuentra en una zona del nivel correspondiente. Cuando la salud del jugador llega a cero, este contador se incrementa en una unidad. Una vez que el contador ha alcanzado un valor de cinco, se desbloquea la capacidad del jugador de realizar un doble salto, permitiendo saltar de nuevo con un menor impulso mientras el jugador se encuentra por encima del terreno o de cualquier objeto de la escena. Si el jugador alcanza un checkpoint no activado del nivel, este contador se reinicia y se pierde la capacidad de realizar un doble salto.

f) EnemyHealthManager

La salud del enemigo se controla mediante un valor numérico entero mayor que cero. Cuando el jugador colisiona con la zona específica diseñada para recibir daño del enemigo, este valor numérico se decrementa.

Cuando la salud del enemigo se reduce a cero, el enemigo es destruido y en el lugar en el que ha sido destruido se crea un objeto que puede ser recogido por el jugador, como una moneda, un corazón para restaurar la salud o un potenciador para dañar a algunos tipos específicos de enemigos.

[2] Se indica el nombre en inglés de los valores debido a que la aplicación solo se encuentra disponible en inglés y no existe una traducción oficial

g) GameManager

Es la clase principal del juego y gestiona los estados del juego cuando se inicia un nivel y cuando se finaliza, además de detener el tiempo de juego en el caso en que el jugador lo pause.

En primer lugar, una vez que el jugador ha seleccionado un nivel en el nivel de selección de niveles y este ha sido cargado, en el método Start() de la clase se asigna la posición en la que debe aparecer el jugador al inicio del mismo y genera todos los enemigos y objetos que se encuentran en el nivel, así como inicializar el contador de monedas recogidas en el nivel.

```
void Start()
{
    Cursor.visible = false;
    Cursor.lockState = CursorLockMode.Locked;
    _respawnPosition = PlayerController.Instance.transform.position;

    // Spawn enemies and items
    _levelActiveEntityContainer = Instantiate(levelEntityContainer);

    // Spawn enemies inside arena
    Arena.Instance.Spawn();

    AddCoins(0);
}
```

Figura 15. Método Start().

Esta clase cuenta con el método RespawnWaiting(), que permite reiniciar el estado del juego cuando el jugador ha agotado su vida y necesita reaparecer. Una vez que se ha agotado la salud del jugador, durante 2 segundos se muestra una pantalla en negro en el que se realiza este proceso:

1) Se destruye todos los enemigos y objetos activos de la escena y de la zona de Arena, en caso de que esta exista.

2) Se resetea el estado de los mecanismos de la escena y de las penalizaciones que sufría el jugador a su estado inicial.

3) Se genera de nuevo los enemigos y objetos de la escena, teniendo en cuenta el nivel de dificultad en el que se encuentra actualmente el juego.

4) Finalmente se reinicia el contador de monedas, se incrementa el contador de muertes y se genera el avatar del jugador en la posición de reaparición que tenía almacenada.

Una vez que el jugador ha adquirido la estrella del final del nivel, el método LevelEndWaiter() se encarga de marcar como completado el nivel y añadir el número de monedas que ha recogido el jugador.

```
public IEnumerator LevelEndWaiter()
{
    //AudioManager.Instance.PlayMusic(levelEndMusic);
    PlayerController.Instance.SetStop(true);
    yield return new WaitForSeconds(1f);
    PlayerController.Instance.SetStop(false);

    // Mark current level as completed
    PlayerPrefs.SetInt(SceneManager.GetActiveScene().name + "_unlocked", 1);

    // Update total coins
    if (PlayerPrefs.HasKey(SceneManager.GetActiveScene().name + "_coins")) {
        if (_currentCoins > PlayerPrefs.GetInt(SceneManager.GetActiveScene().name + "_coins"))
            PlayerPrefs.SetInt(SceneManager.GetActiveScene().name + "_coins", _currentCoins);
    } else
        PlayerPrefs.SetInt(SceneManager.GetActiveScene().name + "_coins", _currentCoins);

    SceneManager.LoadScene(levelToLoad);
}
```

Figura 16. Método LevelEndWaiter().

h) HealthManager

De igual modo que la salud de los enemigos, la salud del jugador se controla mediante un valor numérico con un valor máximo de seis. Cuando el jugador colisiona con un enemigo u objeto que puede hacer daño, este ve reducido su salud en una cantidad determinada por el daño que realiza dicho objeto o enemigo.

Una vez que se ha producido este efecto, el jugador adquiere un tiempo de invencibilidad en el que no se ve afectado por ningún tipo de daño hasta que este tiempo se haya agotado.

i) MovementManager

La clase MovementManager gestiona el movimiento de los objetos no animados del nivel por la escena. Este movimiento puede ser de tres tipos distintos:

1) **Linear:** El objeto se desplaza entre dos puntos tridimensionales (x, y, z) de la escena. Todos los puntos por los que se mueve el objeto se almacenan en una lista de puntos. Si el objeto alcanza la posición de un punto, se actualizará el punto en el que se encuentra y comienza a avanzar hacia el siguiente punto de la lista, tras esperar un tiempo de espera en el que el objeto no se mueve. Si el objeto alcanza el último punto de la lista, este retornará al punto que se encuentra al comienzo de la lista, repitiendo la trayectoria que anteriormente ha realizado.

2) **Rotacional:** En este movimiento, el objeto rota en torno al eje central interior que tiene definido en el modelo. Esta rotación se puede producir en los tres ejes del espacio euclídeo (x, y, z).

3) **Orbital:** El objeto rota de la misma forma que en el movimiento rotacional, aunque alrededor de un objeto exterior denominado pivote, siempre orientado a este objeto pivote.

j) PerformanceManager

Esta clase se encarga de gestionar los valores de *stress* y *engagement* devueltos por la diadema EEG. Para obtener este valor, esta clase llama a un método del plugin de Emotiv encargado de transformar el valor de los estados emocionales del dispositivo a un valor reconocible por Unity.

Debido a que la tasa de frecuencia de Unity es más rápida que la tasa de frecuencia de la diadema de Emotiv, se puede producir que esta devuelva un valor inválido, por lo que ha sido necesario añadir un contador de 0.25 segundos para contrarrestar este error, además de una comprobación de que los valores obtenidos se encuentran en un rango entre 0 y 1. Una vez que se han obtenido valores correctos, estos se envían a la clase que gestiona la dificultad del juego y la interfaz gráfica que muestra esta información.

```
_engLevel = DataStreamManager.Instance.GetPMData("eng");
_strLevel = DataStreamManager.Instance.GetPMData("str");
```

Figura 17. Función para obtener los estados emocionales.

k) UIManager

Gestiona la interfaz gráfica que se muestra una vez se ha iniciado el juego. La interfaz gráfica se divide a su vez en tres interfaces diferenciadas:

1) **Interfaz principal:** La interfaz gráfica principal se muestra mientras el jugador juega al nivel. Contiene información sobre el número de monedas que se han recogido en el nivel, la salud en corazones del jugador, la dificultad actual del juego mostrada en una escala de tres estrellas, la batería del dispositivo y los indicadores de *stress* y *engagement* que experimenta el jugador.

2) **Menú de pausa:** Este menú se activa una vez que el jugador presiona la tecla *Escape* del teclado. Mientras este menú se encuentra activo, el juego entra en un modo de pausa en el que no avanza el tiempo y por lo tanto todos los elementos de la escena se mantienen inmóviles. Este menú ofrece botones que permiten volver al nivel actual, continuando desde el punto en el que se pausó, abrir el menú de opciones, volver al nivel de selección de niveles y volver al menú principal del juego.

3) **Menú de opciones**: Permite modificar los niveles de volumen de música y efectos de sonido del juego en una escala de 0 a 100.

4.2.1.5. UI

Cuando se inicial el juego, la primera pantalla que se muestra es la del menú principal, gestionada por la clase UI_MainMenu.

Cada vez que el jugador inicia el juego, es necesario realizar la conexión con el dispositivo debido a que los datos que transmiten son necesarios para el correcto funcionamiento de este. En caso de que no se encuentre ningún dispositivo conectado, el juego se mantendrá en una pantalla de pausa hasta que el jugador reinicie el juego.

Para poder realizar la conexión entre el dispositivo y el juego es necesario, en primer lugar, añadir a la clase AppConfig la información relativa a este: ClientId, ClientSecret, AppVersion y TmpAppDataDir.

Una vez que se han introducido los datos correspondientes se realiza una llamada a los métodos Init(), encargado de inicializar los valores y Start() para comenzar a conectarse a la API de Cortex y autorizar la aplicación.

```
public void ConnectToCortex()
{
    EmotivUnityItf.Instance.Init(AppConfig.ClientId, AppConfig.ClientSecret, AppConfig.AppVersion, AppConfig.TmpAppDataDir);
    EmotivUnityItf.Instance.Start();
}
```

Figura 18. Método para conectarse a la API.

Finalmente, una vez que se ha conectado el dispositivo a la API de Emotiv, es necesario suscribirse a los flujos de datos correspondientes para obtener los datos necesarios para el videojuego: la batería del dispositivo y los datos relativos al a los estados emocionales del jugador.

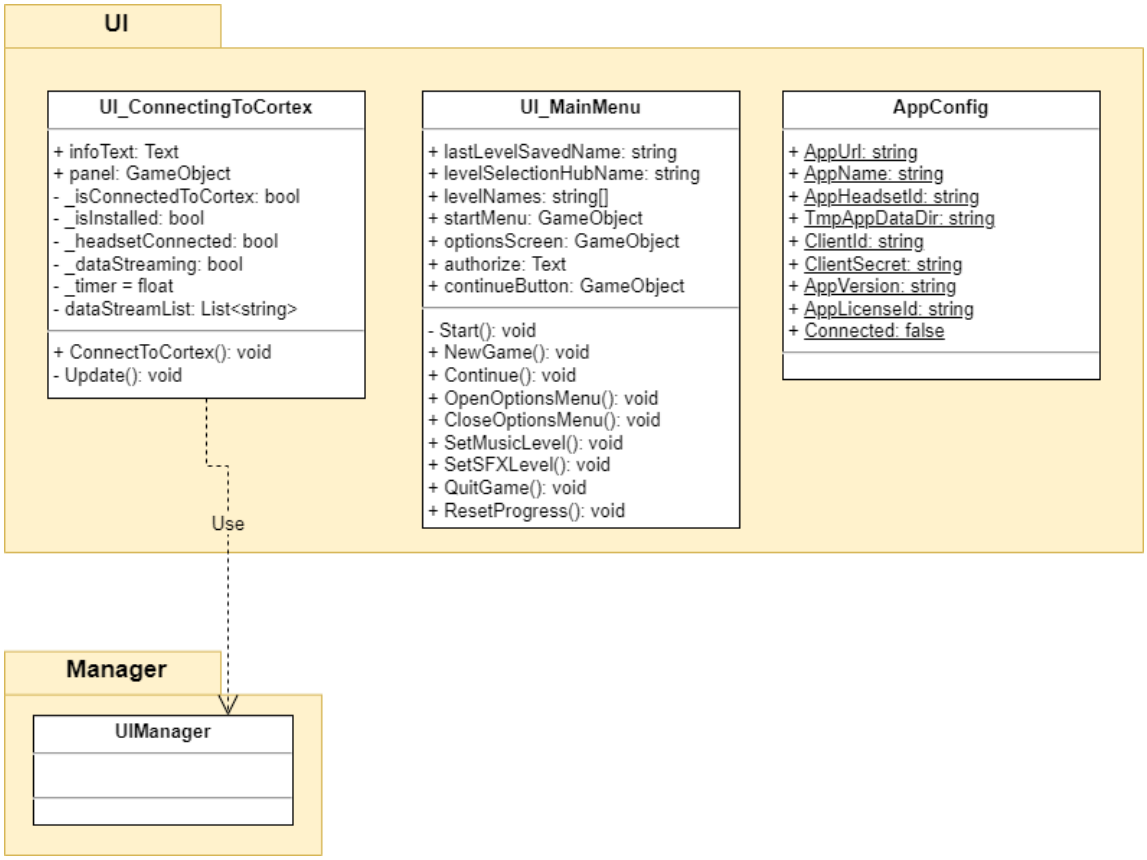

Figura 19. Diagrama de clases del módulo UI.

4.2.2. Diagrama de secuencia

En los diagramas de secuencia, se muestra el comportamiento del juego en función de los estados emocionales de *engagement* y *stress* del jugador mientras lleva equipado el dispositivo y juega al juego. A pesar de que las modificaciones sobre el comportamiento del juego se llevan a cabo en el mismo método, se han desarrollado dos diagramas de secuencia, uno para cada estado emocional, para mostrar de forma más detallada como se producen estos cambios.

4.2.2.1. Diagrama de estrés

En el primer diagrama se muestra como varía el juego en relación con el estado emocional de estrés. El valor de *stress* mide la comodidad del jugador respecto al desafío actual; un nivel de *stress* mayor incrementa la dificultad para realizar una tarea, teniendo efectos negativos sobre la persona. Por consiguiente, se ha utilizado este indicador para modificar los elementos activos del juego que interactúan directamente con el jugador, como se detallan a continuación:

- Velocidad de movimiento del jugador.
- Tiempo de invulnerabilidad del jugador.
- Daño de los enemigos.
- Salud de los enemigos.
- Velocidad de los enemigos
- Tiempo en modo Idle de los enemigos
- Número de veces que se ejecuta una penalización al jugador.

Para manejar la dificultad del juego según el *stress* del jugador, se ha creado una escala de tres dificultades según el valor de este indicador.

1. El nivel de dificultad mayor se presenta cuando el *stress* está por debajo de un umbral de 0.7 sobre 1. En este nivel de dificultad se modifican todos los valores mencionados anteriormente: se decrementa el tiempo en modo *Idle* de los enemigos y el tiempo de invulnerabilidad, mientras que el resto de valores se ven aumentados.

2. Cuando el valor de *stress* se sitúa entre 0.7 y 0.85, se cambia la dificultad del juego a dificultad media. En esta dificultad, se modifican los valores daño y salud de enemigos, además del número de veces que se ejecuta una penalización de igual modo que en el nivel de dificultad del juego mayor, mientras que el resto de características vuelve a sus valores originales.

3. El nivel de dificultad menor se presenta cuando el *stress* es igual o está por encima a un umbral de 0.85 sobre 1. En este segmento de valores, el *stress* ha alcanzado sus máximos valores y es más complicado para el jugador completar la tarea que tiene asignada. En este nivel los valores del juego son iguales a los del comienzo del videojuego. En el caso en que se haya aumentado la dificultad previamente y el valor de *stress* llegue a este umbral, se reducirán los valores correspondientes.

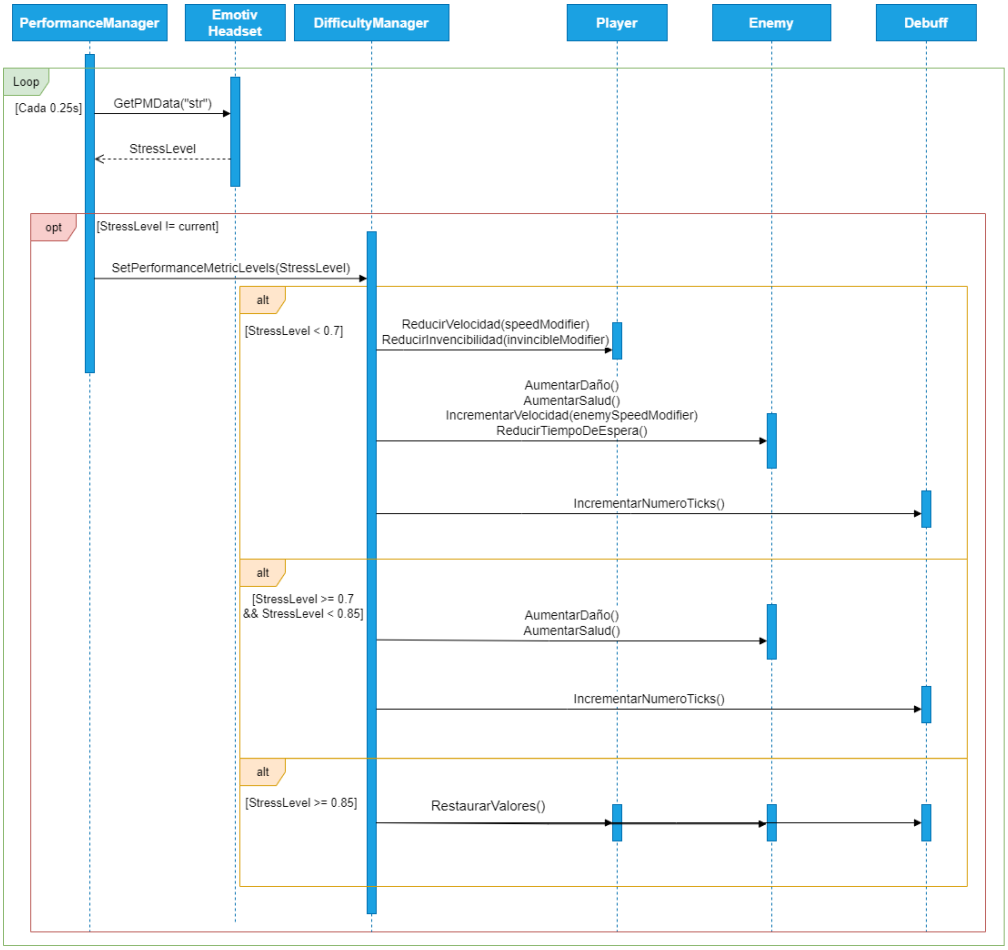

Figura 20. Diagrama de secuencia del indicador stress

4.2.2.2. Diagrama de engagement

En el segundo diagrama se muestra como varía el juego en función del estado emocional de *engagement*. El *engagement* indica la atención que experimenta el usuario hacia los estímulos relevantes para completar la tarea que tiene como objetivo. Cuanto mayor sea la atención y la carga de trabajo que experimenta el jugador, mayor será el valor de este indicador. Este indicador se utiliza para modificar los elementos que conforman la escena:

- Velocidad de movimiento
- Tiempo de espera de las plataformas móviles
- Tiempo de espera de las trampas del nivel
- Enemigos que se generan cuando el jugador se crea en la escena.

Para manejar la dificultad del juego según el *engagement* del jugador, se ha creado una escala de dos dificultades según el valor de este indicador.

1) Si el valor se encuentra por encima de 0.65, entonces el jugador presenta un alto grado de *engagement* y por consiguiente las características de los elementos vuelven a los valores con los que se diseñó originalmente.

2) Si el valor se encuentra por debajo de 0.65, el jugador presenta un grado bajo de *engagement*, por lo que aumenta la velocidad de movimiento y se reduce a la mitad el tiempo de espera. Además, una vez que el jugador muere y reaparece en el nivel, se genera una lista de enemigos y objetos diferentes a la lista inicial de mayor dificultad.

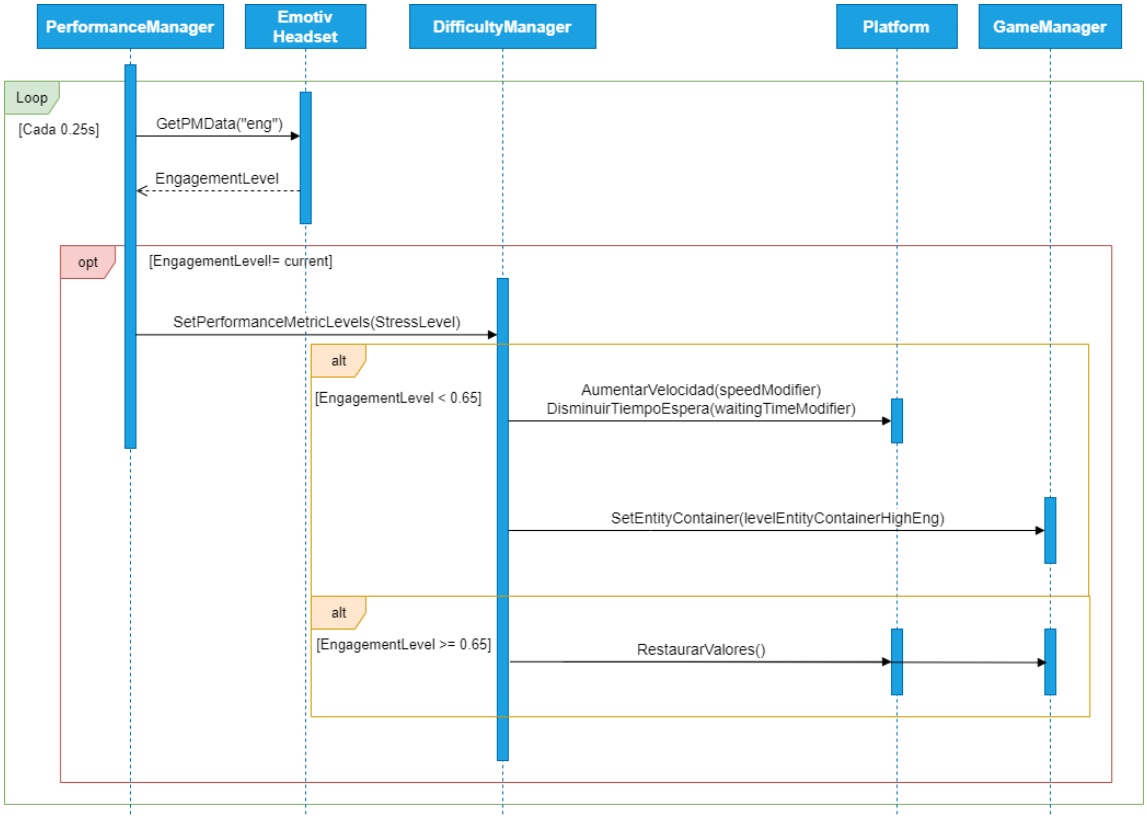

Figura 21. Diagrama de secuencia del indicador *engagement*.

4.3. Diseño del nivel de juego variable

4.3.1. Diseño del nivel

El videojuego cuenta con un único nivel que se ha diseñado haciendo uso de todas las mecánicas y elementos que se han desarrollado. El nivel se ha dividido en 5 zonas principales:

1) La primera zona sirve de introducción al juego y cuenta con las mecánicas principales del juego: enemigos que patrullan el terreno y enemigos que atacan a distancia, plataformas móviles, trampas y potenciadores.

Figura 22. Primera zona del primer nivel.

2) Segunda zona: Tras superar la primera zona del nivel se puede acceder a la segunda, que consiste en un laberinto donde el jugador tiene la cámara fija, con un movimiento de esta solo en horizontal, para evitar que pueda ver más allá de las paredes del mismo. Todo el laberinto se encuentra lleno de trampas fijas y enemigos patrullando.

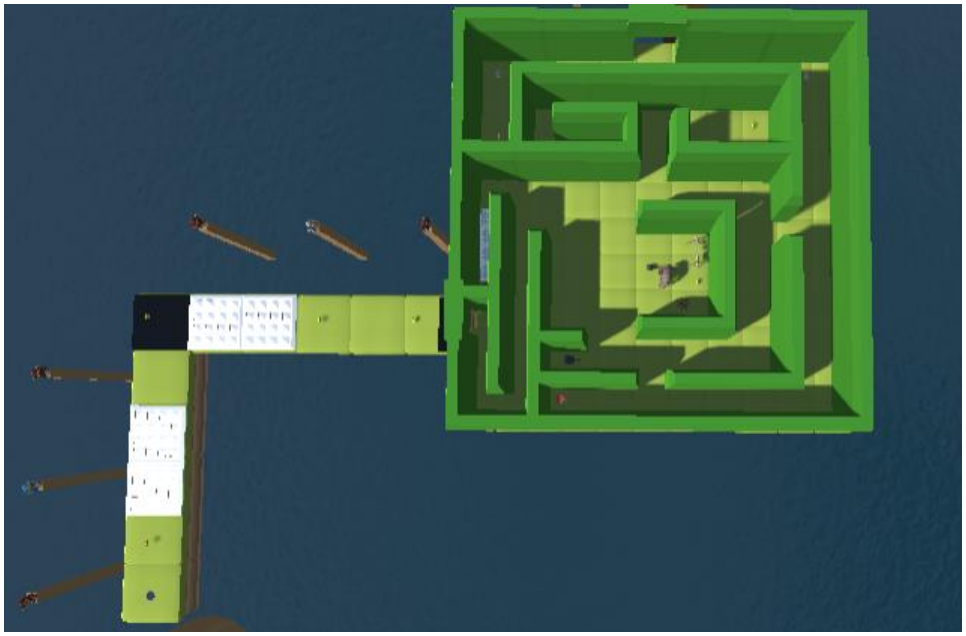

Figura 23. Segunda zona del primer nivel.

3) Tercera zona: La tercera zona está dividida en varias plataformas circulares que realizan un movimiento de rotación mientras que en su superficie se encuentran varios obstáculos y trampas que también tienen la capacidad de desplazarse.

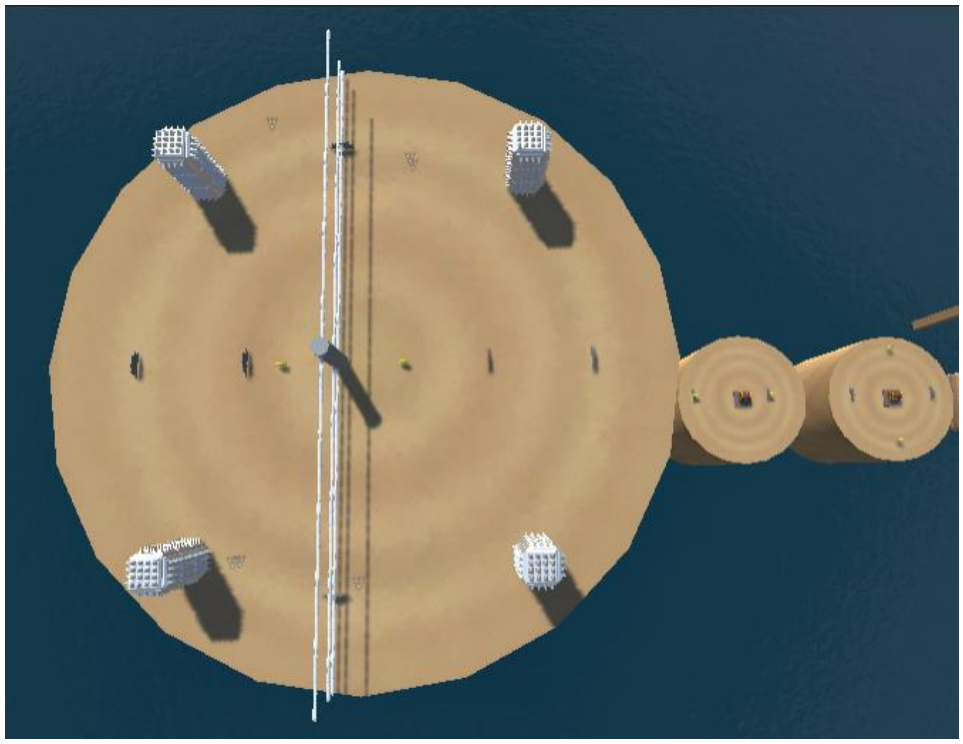

Figura 24. Tercera zona del primer nivel.

4) Arena: La penúltima zona consiste en una Arena, donde el jugador debe derrotar a todos los enemigos que se encuentran en ella. Una vez que se han derrotado todos los enemigos, el botón necesario para desbloquear la puerta y avanzar a la siguiente zona aparece en medio de esta zona.

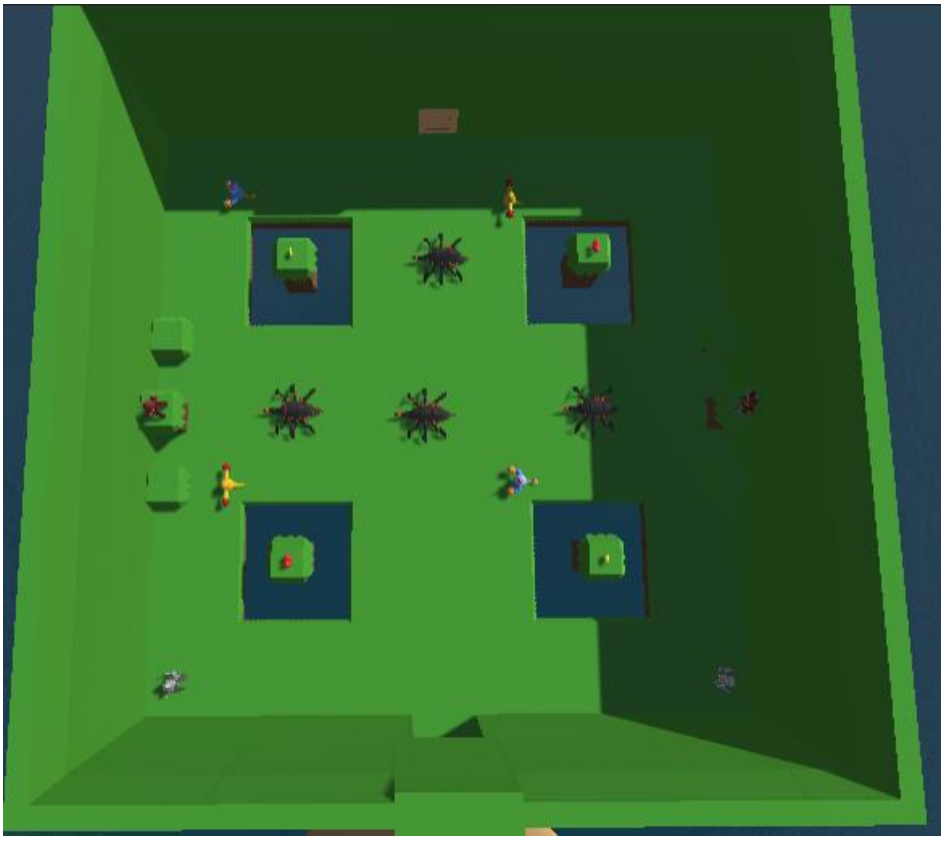

Figura 25. Arena del primer nivel.

5) Zona del jefe final: Una vez que el jugador ha superado las cuatro zonas anteriores, se desbloquea la zona del jefe final. Una vez que se ha derrotado al jefe final aparecerá la estrella final, permitiendo al jugador completar el nivel.

Cada uno de los elementos del juego cuenta con valores que pueden ser modificados, independientes del resto de elementos que se encuentran situados en la escena o del nivel en el que se encuentre. A continuación, se muestran todos los parámetros que pueden ser variados a través del inspector del que dispone Unity, permitiendo una personalización muy extensa de los distintos elementos que forman parte del juego.

a) Jugador

El avatar del jugador contiene los parámetros necesarios para su movimiento por el nivel: velocidad de movimiento, velocidad de rotación, fuerza de salto y rebote al interactuar con un enemigo / potenciador de salto, longitud y fuerza de retroceso en caso de ser dañado y escala de gravedad para controlar la fuerza de salto.

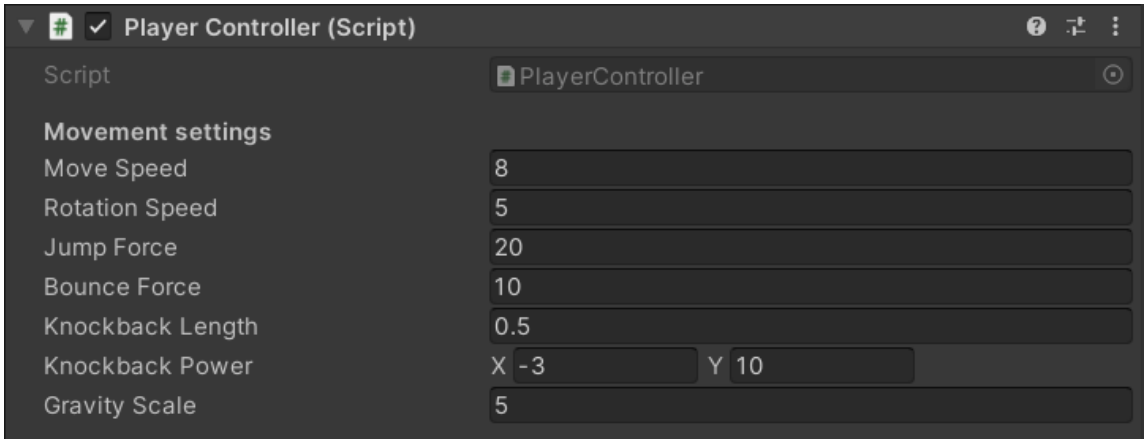

Figura 26. Parámetros del jugador.

b) Plataformas y trampas

Las plataformas y trampas del nivel cuentan con los parámetros para poder desplazarse por la escena. Es posible seleccionar uno de los tres tipos de movimiento diseñados, modificando para cada tipo los parámetros necesarios para llevar a cabo el movimiento.

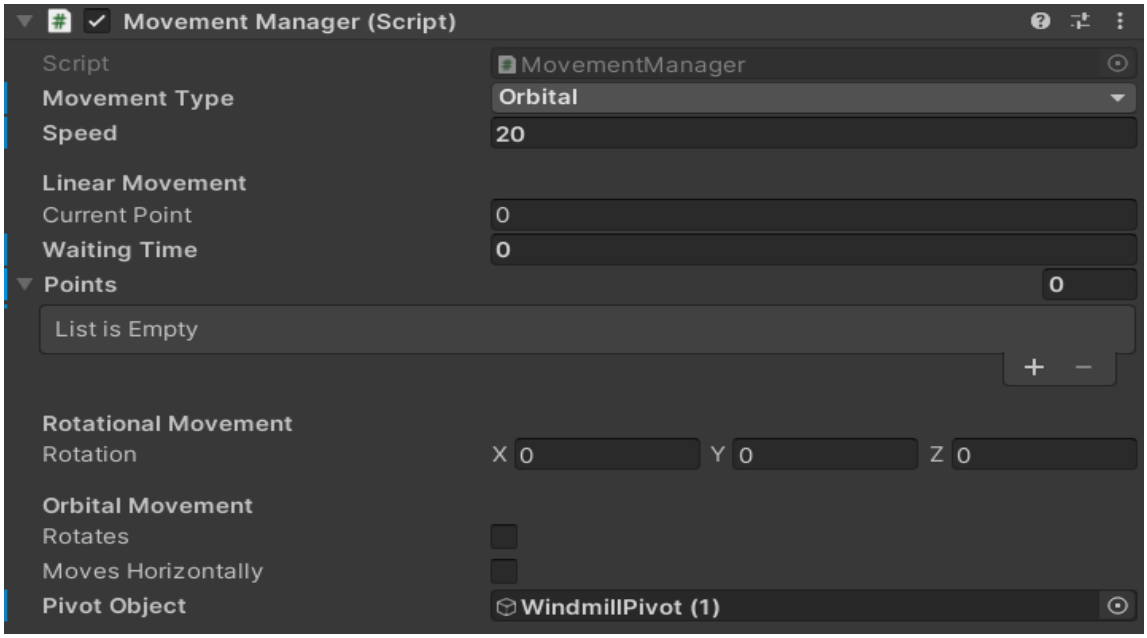

Figura 27. Parámetros de plataformas y trampas.

c) Enemigos

Al existir dos tipos de enemigos en el juego, es necesario contar con dos clases controladoras. Como se puede observar en la Figura 28, el controlador de la izquierda permite modificar los parámetros de los enemigos que pueden atacar a distancia: velocidad de disparo de los proyectiles, rango de ataque hacia el jugador, los proyectiles que pueden disparar o controlar si pueden moverse por el terreno o rotar en su posición para observar el entorno.

El controlador de la derecha permite modificar los parámetros relacionados con los enemigos comunes que atacan cuerpo a cuerpo. Mediante este controlador se puede modificar el tiempo de espera del enemigo al llegar a la posición indicada en la lista de puntos de patrulla, las configuraciones de rango y enfriamiento de ataque y la capacidad de defenderse tras realizar un ataque o volar para poder sortear obstáculos con mayor facilidad.

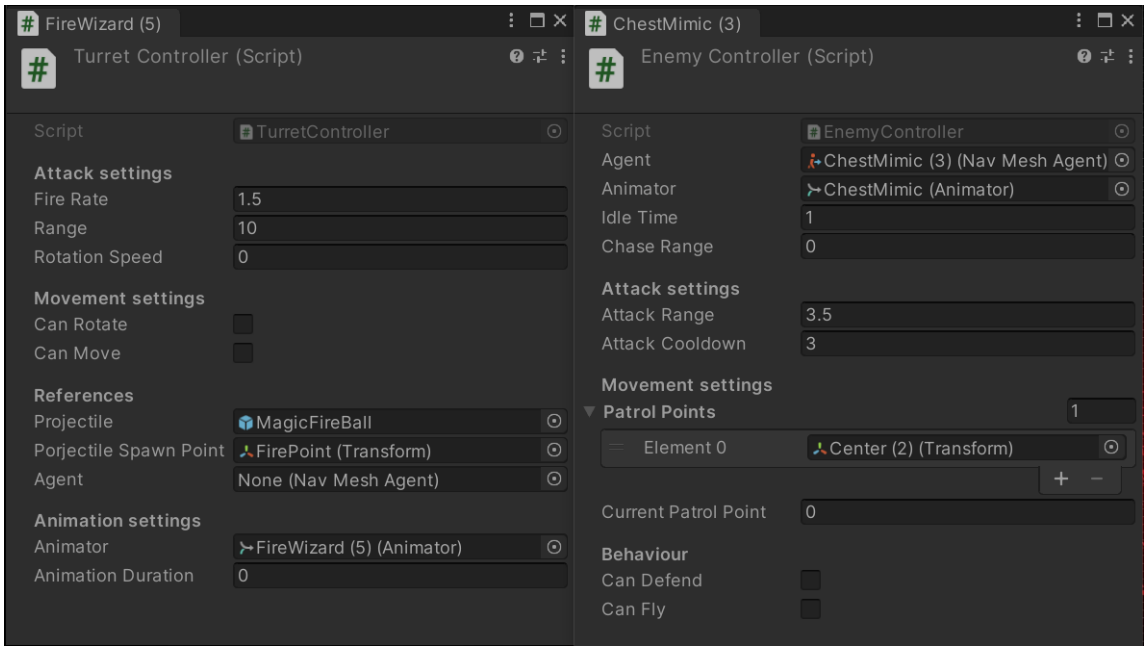

Figura 28. Parámetros de enemigos.

Además, ambos tipos de enemigos cuentan con parámetros con la capacidad de modificar los valores relacionados con la pequeña Inteligencia Artificial que contienen y el daño que pueden realizar al jugador, como se puede ver en la Figura 29.

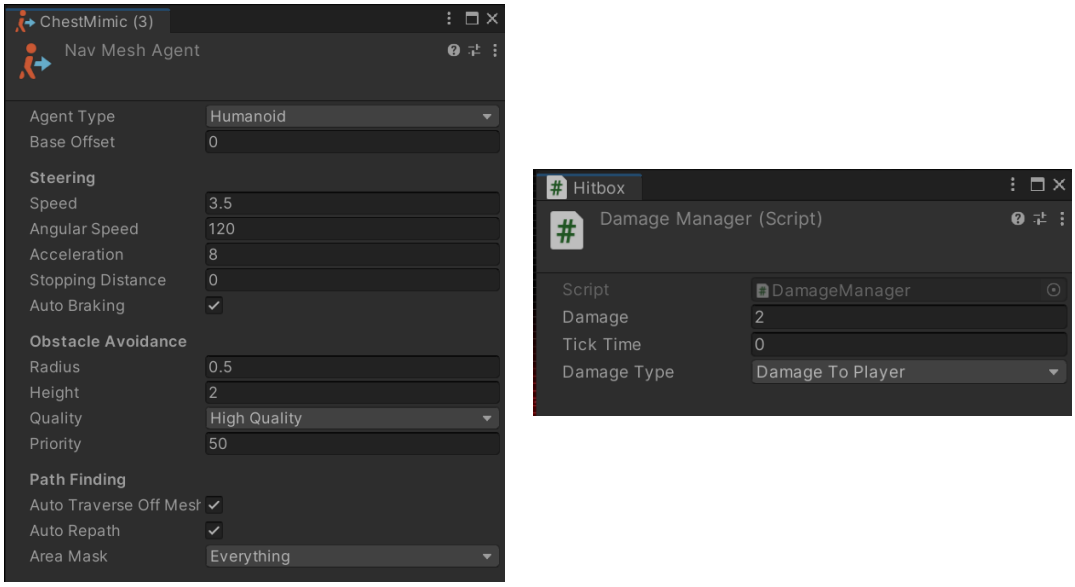

Figura 29. Parámetros de IA y daño de los enemigos.

d) Proyectiles

Los proyectiles cuentan con los parámetros necesario para modificar la velocidad de movimiento, el daño que realizan en caso de impactar con el jugador y el tiempo de vida. Los proyectiles pueden tener dos tipos de movimiento, en línea recta y siguiendo al jugador una vez son lanzados, y tienen la capacidad de generar una explosión al impactar con un elemento del nivel, generando un daño en un área alrededor del impacto.

Además, ciertos proyectiles tienen la capacidad de aplicar una penalización en caso de impactar con el jugador.

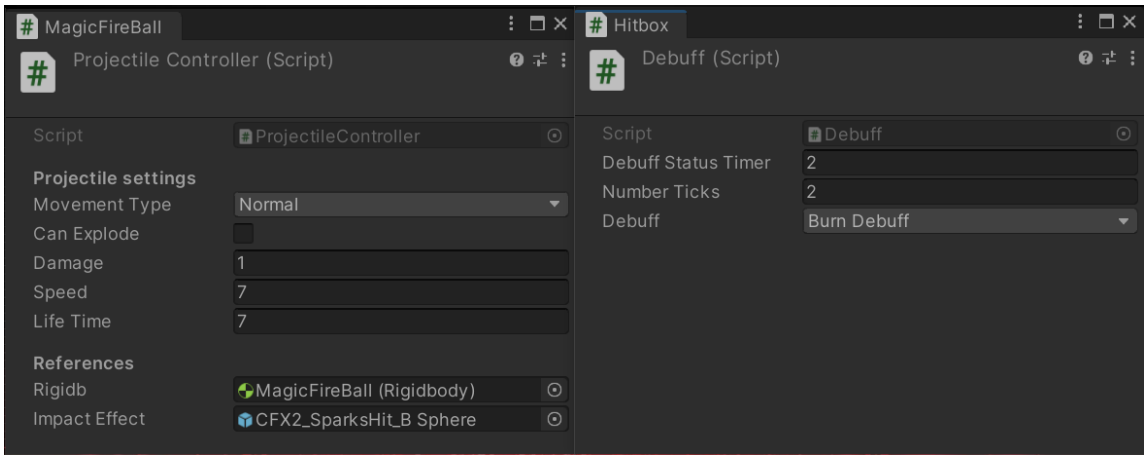

Figura 30. Parámetros de los proyectiles.

4.3.2. Adaptabilidad del nivel

En este apartado se muestra cómo se produce la adaptabilidad del nivel según la modificación de los valores de los estados emocionales del jugador.

En la figura 31, se muestra la ejecución de una partida normal del juego. Como se puede ver, el indicador de *engagement* del jugador muestra un valor bastante alto (círculo azul celeste de valor 94 en la esquina inferior derecha de la interfaz gráfica del juego). A la derecha de la pantalla de puede observar los parámetros de una de las plataformas de esta zona del nivel: tipo de movimiento, velocidad y tiempo de espera.

Figura 31. Estado del juego con un alto nivel de *engagement*.

Sin embargo, como se muestra en la Figura 32, una vez que disminuye el *engagement* del jugador por debajo del umbral predefinido, la velocidad de las plataformas y trampas del nivel se ve aumentada un 50% y el tiempo de espera de las plataformas se reduce a la mitad. Además, los enemigos situados en el terreno han sido sustituidos por otros que representan una mayor dificultad. Los esqueletos se han cambiado por enemigos con la capacidad de flotar, pudiendo sortear obstáculos que los enemigos terrestres no pueden y se ha añadido un nuevo enemigo a distancia entre los 2 árboles situados en la parte izquierda de la zona nivel.

Figura 32. Estado del juego con un bajo nivel de engagement.

Como se ha comentado anteriormente, para realizar la adaptabilidad del juego en función del *stress* del jugador, se han modificado los parámetros relacionados con los elementos activos del juego. Como se puede observar en la Figura 30, en el nivel de dificultad mayor, cuando el estrés ha superado el umbral de 0.85 sobre 1, los valores de los parámetros son los que tienen asignados por defecto al comenzar el nivel.

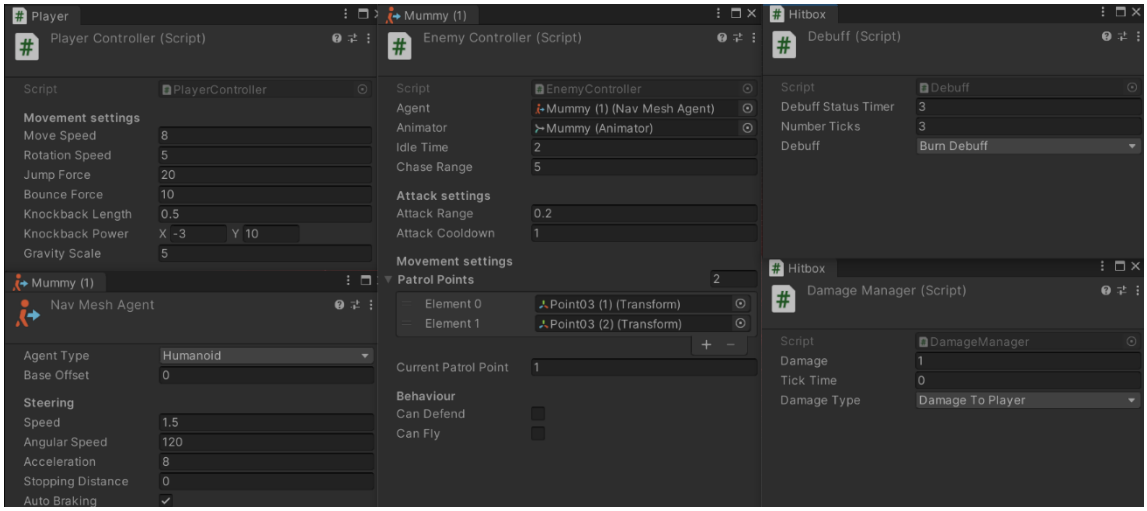

Figura 33. Parámetros del juego en el mayor nivel de dificultad.

Cuando el nivel de *stress* ha bajado por debajo del umbral mínimo, 0.7 sobre 1, el juego alcanza su mayor nivel de dificultad. En la Figura 31, se puede observar cómo se han modificado todas las características de los elementos: la velocidad del jugador y el tiempo en estado *Idle* de los enemigos han disminuido mientras que la velocidad de movimiento de los enemigos, su daño y el número de veces que se aplica una penalización al jugador es aumentado. Los parámetros relativos a la salud actual de los enemigos y al tiempo de invulnerabilidad del jugador no se pueden mostrar en el inspector de Unity debido a que son datos privados del juego, aunque al igual que el resto de parámetros mostrados, también son modificados.

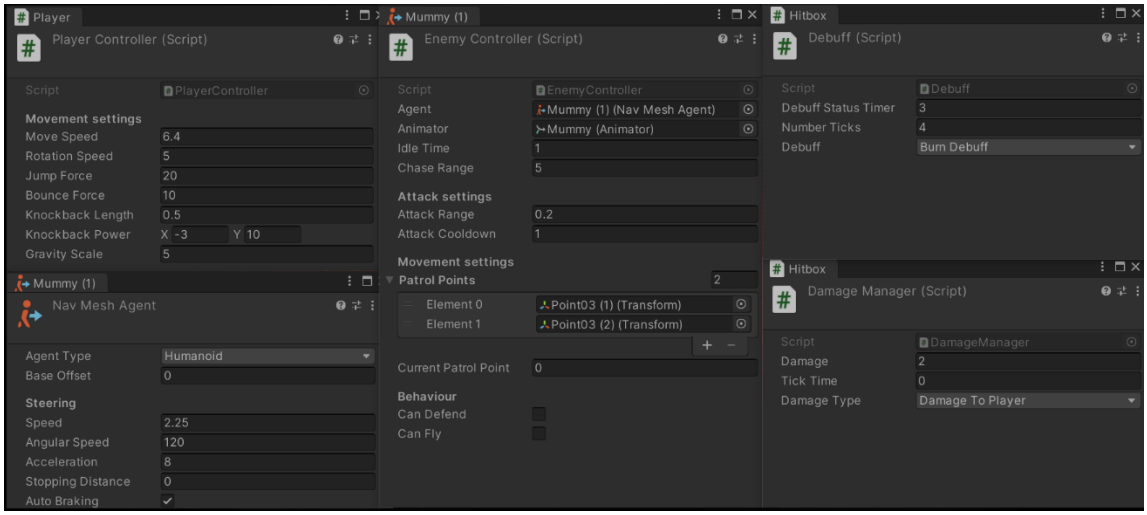

Figura 34. Parámetros del juego en el menor nivel de dificultad.

A modo de resumen se facilita las siguientes tablas para tener una perspectiva general de las propiedades de adaptabilidad del nivel a los estados de *stress* y de *engagement*. Los parámetros privados de los elementos se marcan de color rojo.

Stress	
Descripción	**Parámetro**
Velocidad de movimiento del jugador	Move Speed
Tiempo de invulnerabilidad del jugador	Invincible Length
Velocidad de movimiento del enemigo	Speed
Tiempo en estado *Idle* del enemigo	Idle Time
Daño del enemigo	Damage
Vida actual del enemigo	Health
Número de veces que se aplica una penalización	Number Ticks

Tabla 2. Propiedades de adaptabilidad del estado de *stress*.

Engagement	
Descripción	**Parámetro**
Velocidad de movimiento de plataformas y trampas	Speed
Tiempo de espera de plataformas y trampas	Waiting Time
Contenedor de todas las entidades del juego: enemigos, Arena, *ítems* y puntos de patrulla.	Entity Container

Tabla 3. Propiedades de adaptabilidad del estado engagement.

5. Validación del sistema

Durante el desarrollo se han realizado varias pruebas para comprobar que el funcionamiento del juego corresponde con el objetivo que inicialmente se planteó. Para realizar el experimento fue necesario hacer uso del dispositivo Emotiv EPOC X y conectarlo al juego en Unity mediante el plugin específicamente desarrollado.

Para realizar las pruebas fue necesario contar con una persona externa al desarrollo, por lo que estas pruebas se realizaron a través de una persona de 63 años sin ningún conocimiento en videojuegos desarrollados para plataformas de escritorio y con un conocimiento limitado sobre aspectos tecnológicos y manejo del teclado.

1) La primera prueba se realizó a mitad del desarrollo del proyecto, cuando el videojuego contaba con las funcionalidades básicas y necesarias para poder jugar. Para realizar este experimento se diseñó un nivel extremadamente difícil, con apenas probabilidad para poder llegar a completarlo.

   El objetivo de este experimento fue analizar cómo respondían las emociones del jugador captadas por la diadema EEG en situaciones de estrés extremo, como un nivel prácticamente imposible, así como en situaciones donde el sujeto se encontraba totalmente relajado.

2) La segunda prueba se realizó una vez fue terminado el juego, con la misma persona. En este experimento se analizó cómo los cambios en los valores de los estados emocionales influían en las características del juego y que percepción de jugabilidad tenía el jugador cuando se producían estos. Para realizar este experimento se hizo uso de un dispositivo virtual.

5.1. Primer experimento

Como se ha comentado anteriormente, en el primer experimento se estudió cómo el estado emocional del jugador se veía afectado en situaciones de estrés nulo y extremo. Para obtener estos valores de emociones se utilizó la aplicación EmotivBCI que venía incluido en Emotiv Launcher. Esta aplicación ofrece distintas gráficas sobre todo los datos que mide el dispositivo, como las expresiones faciales, los estados cognitivos y emocionales o los sensores de movimiento. En ambas situaciones de estrés, se realizaron sesiones de 3 minutos.

Este tiempo de sesión se obtuvo tras la realización de varias pruebas preliminares, observando durante cuánto tiempo se muestran los datos en las gráficas incluidas en la aplicación EmotivBCI, debido a que no se contaba con la licencia de pago necesaria para obtener todos los datos en crudo de la sesión. Una vez que la sesión superaba los tres minutos, los datos más antiguos dejaban de mostrarse en las gráficas incluidas en la aplicación para dejar paso a los más actuales.

Para realizar el experimento, fue necesario un nivel del juego diseñado específicamente para generar en el jugador un alto nivel de estrés. Para alcanzar este objetivo el nivel contaba con un gran número de obstáculos y enemigos dispersos por toda la escena, desde el momento en el que el jugador entraba al nivel. Este método provocaba que el jugador estuviera en constante movimiento impidiendo que tuviera un respiro para que la relajación disminuyera y aumentara el nivel de estrés.

En primer lugar, antes de que el jugador pudiera jugar al nivel, se analizaron los valores de los estados emocionales cuando la persona se encontraba en completa relajación mientras estaba reposando, sentado en una silla, sin realizar ninguna acción y con los ojos cerrados durante 3 minutos.

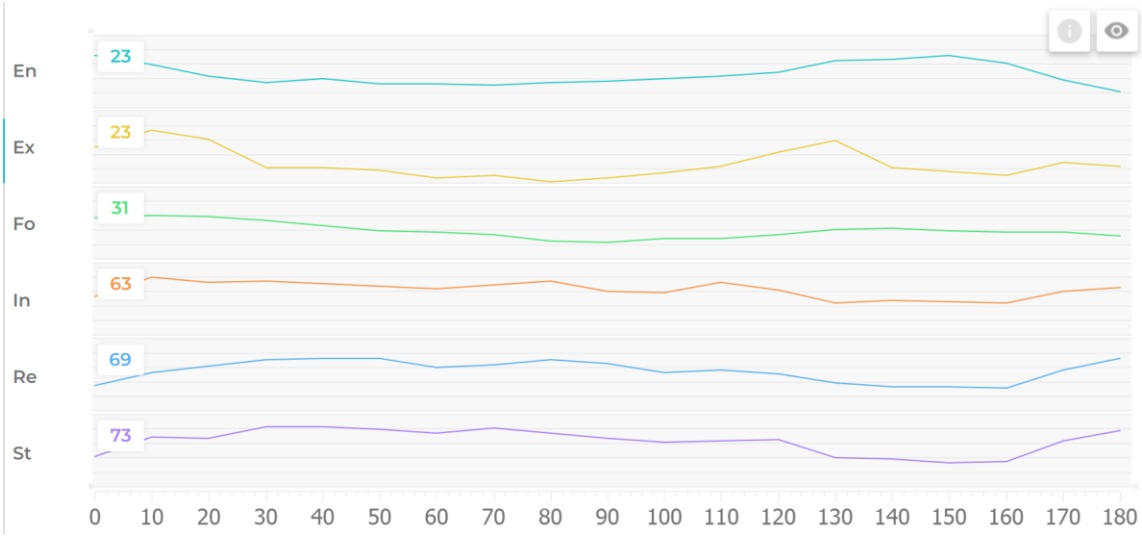

Figura 35. Valores de las emociones en relajación.

` Como se puede observar en la gráfica (Figura 35), durante todo el tiempo que el sujeto estuvo relajado las emociones de interés (In) y relajación (Re) son los que estuvieron en un rango más alto de valores, entre 60 y 75, indicando que la persona se encontraba relajada y experimentaba un medio / alto grado de atracción hacia la actividad propuesta, la relajación con los ojos cerrados.

El resto de emociones, sin embargo, tenían valores más bajos que estas dos emociones debido a que el sujeto no estaba realizando ninguna acción. Como se puede ver en la gráfica, al final del experimento el valor de estrés (St) alcanzó el mayor valor y este pico de los últimos 20 segundos se produjo debido a los movimientos faciales de la persona que alteraron la capacidad para detectar esta emoción.

En el segundo experimento, se analizaron los valores de las emociones mientras el jugador jugaba al nivel, que se obtuvieron en una sesión de la misma duración que el experimento anterior, 3 minutos.

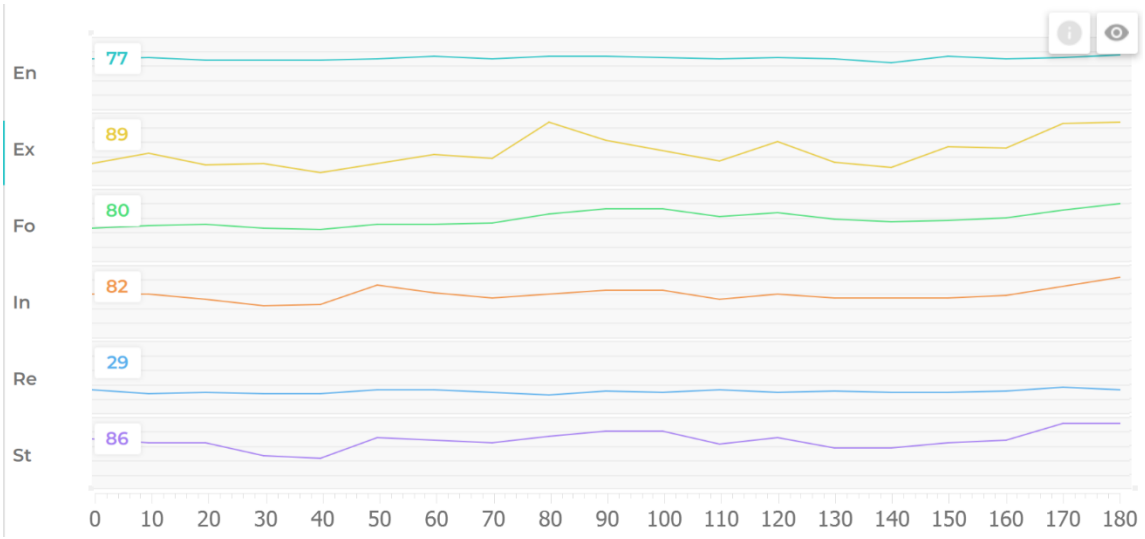

Figura 36. Valores de las emociones jugando.

Como se puede observar en la gráfica, al contrario que en la sesión de relajación, los indicadores de las emociones tenían valores mucho más altos. Mientras iba avanzando el tiempo de juego y el jugador se sentía más frustrado al no poder completar la tarea principal que se le había impuesto, llegar al final del nivel, los valores iban aumentando hasta situarse en un rango de valores entre 75 y 99, siendo este último valor el máximo alcanzado por un indicador en específico, el estrés.

5.2. Segundo experimento

Este experimento se realizó una vez que el juego fue finalizado. El objetivo de este experimento era analizar que percepción tenía el jugador cuando la jugabilidad del juego se adaptaba a los estados emocionales de *stress* y *engagement* de este.

Para realizar este experimento no fue necesario disponer de la diadema EEG física, debido a que Emotiv Launcher dispone de la posibilidad de crear un dispositivo virtual capaz de simular los indicadores de los estados emocionales del jugador, pudiendo seleccionar cada uno de los valores que ofrece manualmente.

5.2.1. Cuestionario de usabilidad QUIS

Una vez que el sujeto finalizó la sesión de juego, que duró aproximadamente 15 minutos, se le realizaron dos encuestas para que valorase la experiencia de usuario (UX) del juego y la adaptabilidad del mismo a los estados emocionales.

El primer cuestionario utilizado para evaluar la usabilidad del juego fue el Cuestionario de satisfacción con la interacción del usuario (QUIS) de 1988 [12]. El cuestionario QUIS cuenta con diferentes apartados independientes de cada uno donde cada respuesta se encuentra en una escala de valores de 0 a 9, con la posibilidad de no responder a la pregunta en caso de duda y tiene como objetivo guiar al evaluador en el diseño de sistemas interactivos y valorar las áreas en el que este sistema puede ser mejorado.

OVERALL REACTION TO THE SOFTWARE													
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		**NA**
	terrible								X			wonderful	
	difficult					X						easy	
	frustrating							X				satisfying	
	Inadequate power											adequate power	X
	dull									X		stimulating	
	rigid				X							flexible	
SCREEN													
Reading characters on the screen	hard									X		easy	
Highlighting simplifies task	not at all								X			very much	
Organization of information	confusing								X			very clean	
Sequence of screen	confusing							X				very clean	
TERMINOLOGY AND SYSTEM INFORMATION													
Use of terms throughout system	inconsistent											consistent	X
Terminology related to task	never							X				always	
Position of messages on screen	inconsistent								X			consistent	
Prompts for input	confusing											clear	X
Computer informs about its progress	never											always	X
Error messages	unhelpful											helpful	X

LEARNING													
Learning to operate system	difficult				X							easy	
Exploring new features by trial and error	difficult											easy	X
Remembering names and use of commands	difficult			X								easy	
Performing tasks is straightforward	never							X				always	
Help messages on the screen	unhelpful											helpful	X
Supplemental reference materials	confusing					X						clear	
SYSTEM CAPABILITIES													
SYSTEM speed	too slow										X	fast enough	
SYSTEM reliability	unreliable											reliable	X
SYSTEM tends to be	noisy							X				quiet	
Correcting your mistakes	difficult					X						easy	
Designed for all levels of user	never									X		always	

Tabla 4. Cuestionario de satisfacción con la interacción del usuario (QUIS).

Tras realizar la encuesta al sujeto, se puede observar cómo entre los aspectos más positivos del juego se encuentra la interfaz gráfica de usuario; la información que se presenta al jugador mientras juega y su relevancia. Debido a que el sujeto no tenía experiencia con juegos de este tipo y con el manejo del teclado, le fue más difícil aprender a manejar al personaje principal del juego.

5.1.2. Cuestionario de experiencia de usuario SUPR-Q

El segundo cuestionario utilizado para evaluar el juego fue el Cuestionario estandarizado de rango percentil de la experiencia del usuario (SUPR-Q) de 2015 [13]. El cuestionario SUPR-Q cuenta con ocho preguntas divididas en cuatro apartados que mide la percepción del usuario respecto a la usabilidad, confianza y credibilidad, apariencia visual y lealtad de una página web. Todas las preguntas se valoran con un valor entre 1 y 5, siendo 5 el máximo excepto la última, que cuenta con una valoración entre 1 y 10.

Debido a que el cuestionario ha sido diseñado para una página web, se han redireccionado las preguntas para valorar estas cuatro categorías según la jugabilidad e interfaz gráfica del juego. Una vez que se han obtenido todos los valores de la encuesta, se calcula la media de los valores de la encuesta, así como el percentil en el que se encuentra el juego respecto a una base de datos con las valoraciones de distintos proyectos que han realizado el mismo test.

CATEGORY	QUESTION	SCORE
Usability	Emot3D is easy to use	4
	It is easy to navigate within Emot3D	5
Trust & Credibility	The information on Emot3D is credible	5
	The information on Emot3D is trustworthy	5
Appearance	I find Emot3D to be attractive	5
	Emot3D has a clean and simple presentation	5
Loyalty	I will likely return to Emot3D in the future	4
	How likely are you to recommend Emot3D to a friend or colleague?	9

Tabla 5. Cuestionario estandarizado de rango percentil de la experiencia del usuario (SUPR-Q).

$$Raw\ SUPR-Q\quad Score = \frac{4+5+5+5+5+5+4+\left({}^{1}/_{2}*9\right)}{8} = 4{,}6875$$

A pesar de que no se puede calcular el percentil en el que se encuentra la usabilidad del juego debido a que es necesario el pago de una licencia para el acceso a la base de datos y realizar las comparaciones, además de solo poder realizar la encuesta a una sola persona por problemas de logísticas para realizar las pruebas, se ha obtenido un resultado que indica que el videojuego tiene un alto grado de usabilidad para la persona que ha realizado las pruebas.

Este resultado, aun siendo preliminar y con la necesidad de realizar más pruebas, es positivo y da pistas de por dónde continuar mejorando el juego y ofrece indicios de seguir trabajando en la buena dirección.

6. Problemas y soluciones

A continuación, se detallan los problemas que han surgido durante el desarrollo del proyecto y a que solución se ha llegado para poder resolverlos.

- **Emotiv no ofrecía soporte a su SDK**

  Al comienzo del desarrollo, se planteó realizar la integración de la diadema EEG con el motor de videojuegos de Unity mediante el SDK que ofrecía Emotiv. Debido a que el SDK se desarrolló hace más de 7 años, Emotiv dejó de darle soporte para centrarse en el desarrollo de una API nueva, denominada Cortex [14], por lo que la aplicación necesaria ya no se encontraba disponible para su descarga.

  Solución: Fue necesario buscar un nuevo modo para realizar esta integración con el motor de videojuegos. Como la página web de Emotiv no mostraba de forma clara qué usar para llevar a cabo el objetivo, se perdió un poco de tiempo buscando hasta poder encontrar el plugin que se utilizó para este proyecto.

- **El plugin contenía errores de funcionamiento**

  La versión del plugin que se descargó al comienzo del desarrollo del proyecto, que en su momento era la más actualizada, contenía un error en el código que manejaba el controlador de eventos para poder realizar la conexión entre el motor de videojuegos y el plugin. Debido a que este error no estaba documentado, se inició el desarrollo usando esta versión, por lo que se perdieron unas dos semanas más de las previstas intentado hacer funcionar el plugin.

  Solución: Fue necesario esperar hasta que fuera lanzada una nueva versión corrigiendo este error para realizar la integración, por lo que se retrasó el desarrollo del juego.

- **La documentación del plugin es escasa**

  La documentación que se ofrecía en GitHub, la página donde se podía descargar el plugin, solo describía de forma muy general el funcionamiento total del mismo plugin. Además, el código solo ofrecía una pequeña descripción de cada uno de los métodos de cada clase del plugin.

  Aunque la documentación de la API de Emotiv ofrecía mucha más información, explicaba su funcionamiento en función de JSON y WebSockets y no en C#, el lenguaje de programación utilizado para el código, por lo que la amplia mayoría de las llamadas no se correspondían con el método del código que se debía utilizar.

Solución: Hubo que probar cada una de las funciones del plugin por separado para aprender su funcionamiento y que valores se obtenían a partir de cada una.

- **Escasa formación en Unity**

  A pesar de que ya he tenido algo de experiencia en el desarrollo de un juego, el motor de videojuegos utilizado y su funcionamiento, así como el género del juego eran completamente distintos, por lo que mi formación en Unity era bastante escasa.

  Solución: Fue necesario usar una parte del tiempo planificado para el desarrollo del trabajo fin de grado para realizar una formación en Unity mediante tutoriales que se encontraban disponibles en Internet.

- **Los sensores necesitan ser hidratados**

  Debido a que la solución necesaria para hidratar los sensores para que se pudiera realizar una captación fiable de los impulsos electromagnéticos obtenidos a través de la diadema EEG se agotaba a un ritmo acelerado, durante la mitad de desarrollo, para realizar las pruebas planificadas, se intentó hidratar solo los 8 sensores que se colocaban en la parte frontal de la cabeza.

  Sin embargo, para que la diadema EEG pudiera captar estos valores era necesario que todos los sensores estuviesen hidratados.

  Solución: Se redujo el número de pruebas que se realizaron a lo largo del desarrollo del proyecto para evitar que se agotara la solución.

- **Se agotó la solución necesaria para hidratar los sensores**

  A finales del desarrollo, debido al problema anterior, se agotó la solución salina necesaria para hidratar los sensores y no se podían realizar nuevas pruebas.

  Solución: Se tuvo que adquirir una solución de lentillas en una farmacia, ya que como indicaba la documentación del dispositivo se podía usar como reemplazo a la solución original. Tras comprobar el correcto funcionamiento de esta nueva solución, se continuó con las pruebas planificadas.

- **El cuero cabelludo interfería con la detección de valores**

  Para obtener el mejor resultado de los indicadores era necesario alcanzar una calidad de contacto del 98%. Para alcanzar este valor, los sensores de la diadema necesitaban hacer un contacto directo con el cuero cabelludo de la persona que lo lleva puesto.

Tras realizar las pruebas conmigo mismo, al tener un cabello más grueso, los sensores no detectaban de una forma correcta este contacto, por lo que los resultados no eran fiables y concluyentes.

Solución: Fue necesario buscar a una persona con una menor cantidad de volumen de cabello para realizar las pruebas, por lo que limitó el mismo número de pruebas que se podían realizar.

- **La aplicación de Emotiv se actualizó**

  A finales de mayo, la aplicación de Emotiv se actualizó automáticamente a una versión más reciente. Esta actualización provocó que el modo en el que esta aplicación interactuaba con el plugin cambiase, modificando los valores que le transmitía.

  Solución: Hubo que ajustar el código del videojuego que interactuaba con el plugin de Emotiv.

- **Poco tiempo de desarrollo para el proyecto**

  Debido a que se comenzó con el desarrollo del proyecto a comienzos de abril, con su posterior entrega en julio, no se ha podido seguir la planificación que se planteó al comienzo del desarrollo del mismo.

  Solución: Fue necesario recortar algunas de las funcionalidades que se tenían previstas para el juego en un primer momento para poder entregar una versión funcional del mismo. Entre las funcionalidades que no se desarrollaron, se encontraba utilizar el giroscopio del dispositivo para que el jugador pudiera controlar la cámara mediante los movimientos de su cabeza y usar un entrenamiento para que el jugador pudiera utilizar sus pensamientos para el movimiento del personaje que controlaba.

7. Manual de usuario

Se ha diseñado un breve manual para guiar al usuario del videojuego sobre la configuración para realizar su ejecución y los controles e iconos que forman parte del juego.

7.1. Configuración inicial

En primer lugar, es necesario tener instalado la aplicación de Emotiv Launcher en el mismo ordenador donde se va a ejecutar el juego y haberse registrado con una cuenta de Emotiv para poder obtener los datos necesarios para conectar el dispositivo.

Una vez que se han obtenido los datos, es necesario completar estos datos en el archivo AppConfig.cs: ClientId, ClientSecret, AppVersion y TmpAppDataDir. Los primeros dos valores se obtienen una vez que se ha creado la cuenta de Emotiv y se ha registrado una aplicación.

Finalmente, antes de iniciar el juego, es necesario que el dispositivo que se vaya a utilizar para medir los estados emocionales, ya sea físico o virtual creado mediante la aplicación de Emotiv, se encuentre encendido. Una vez que todos estos requerimientos se encuentren satisfechos, se puede proceder al inicio del juego y a la conexión con la diadema EEG utilizada, mostrando en primer lugar el proceso de validación de esta conexión, dando paso al menú principal[3].

7.2. Controles

El juego cuenta con un control básico del sistema. Los controles del juego se detallan a continuación en la siguiente tabla:

[3] Debido al funcionamiento de la aplicación de Emotiv, es posible que se valide la conexión con el dispositivo, pero no se suscriba a ningún flujo de datos, provocando que los indicadores devuelvan un valor negativo y no se actualicen. Este problema se puede solucionar seleccionando el botón *QUIT* del menú principal y volviendo a ejecutar el juego.

Tecla	Acción
W	Avanzar
S	Mover hacia atrás
A	Moverse a la izquierda
D	Moverse a la derecha
Espacio	Saltar
Esc	Abrir menú de opciones
Ratón	Mover la cámara

Tabla 6. Controles del juego.

7.3. Interfaz gráfica

La interfaz gráfica del juego se muestra una vez que el jugador se encuentra en un nivel del juego. Los diferentes elementos que se observan en la interfaz se detallan en la siguiente tabla:

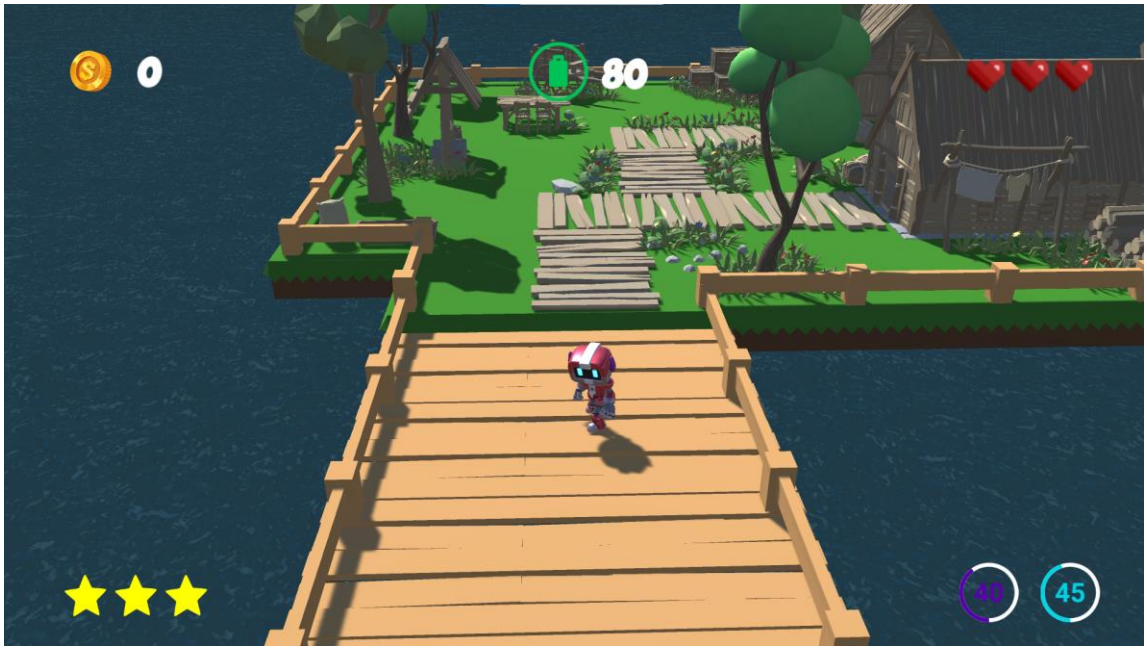

Figura 37. Interfaz gráfica.

Icono	Descripción
	Indicador de salud del jugador, puede mostrar un corazón completo, medio corazón o no mostrarlo
	Indicador de batería del dispositivo conectado. Muestra un valor entre 0 y 100
	Monedes que ha recogido el jugador en un nivel. Su valor mínimo es 0
	Dificultad del juego, variando en función del indicador de estrés. Una estrella indica la dificultad más baja y tres estrellas la más alta
	Indicador de *stress* del jugador. Muestra un valor entre 0 y 100.
	Indicador de *engagement* del jugador. Muestra un valor entre 0 y 100.
	El jugador puede realizar un doble salto. Este indicador se desactiva al comienzo del juego o cuando el jugador interactúa con un checkpoint y se activa cuando el jugador ha muerto cinco veces.

Tabla 7. Iconos de la interfaz gráfica.

8. Conclusiones y vías futuras

En este proyecto se ha diseñado e implementado un nivel de un videojuego que considera una alta posibilidad de adaptación a cambios de dificultad según los estados emocionales del jugador. Durante episodios estresantes, la dificultad del juego se ve reducida para disminuir ese estado mientras que esta se ve incrementada si la atención del jugador disminuye. Estos cambios se ven reflejados en los elementos que se encuentran en el videojuego y permite una alta configuración para modificar la dinámica del juego mientras el usuario se encuentra jugando.

El desarrollo de este proyecto no solo me ha ayudado a crecer en un ámbito académico, sino que también me ha aportado una experiencia en tecnologías que anteriormente no había tenido la posibilidad de utilizar. Es cierto que algunos aspectos de este desarrollo se podrían haber hecho mejor, aunque también han surgido complicaciones, como la falta de soporte del software y su escasa documentación o falta de tiempo debido a la carga de trabajo generada durante el curso lectivo. A pesar de estas complicaciones, creo que se ha alcanzado un resultado satisfactorio y motivación para continuar trabajando en esta línea.

Con respecto al futuro, considero que se pueden ampliar las funcionalidades del proyecto desarrollado. Debido a los problemas que han surgido durante el mismo y una falta de experiencia con las tecnologías que se han hecho uso durante el desarrollo, no se ha podido implementar todo lo que se había pensado antes de comenzar este proyecto. Estas mejoras no sólo se podrían lograr a través de la jugabilidad que ofrece el juego, sino haciendo uso de todas las capacidades que ofrecen las diademas EEG que se han utilizado o de dispositivos que cuentan con una funcionalidad similar. Se podrían haber aprovechado todos los estados emocionales de los que dispone el dispositivo o utilizar las funciones de detección de expresiones faciales y pensamientos para realizar el movimiento del jugador. Además, sería interesante poder contar con una licencia donde se obtengan los datos en crudo de la EEG y explorar más fielmente los parámetros de estrés o *engagement*, creando modelos inteligentes (por ejemplo, basados en aprendizaje automático) que aprendan de los estados estresantes o de *engagement* de un jugador; ya que, ahora mismo sólo podemos confiar en los valores de éstos ofrecidos por Emotiv.

Ha sido interesante el uso de este tipo de dispositivos, que no solo tienen la capacidad de ofrecer una nueva forma de experimentar un videojuego, sino que también puede ser utilizado en otros ámbitos debido a que puede ofrecer una accesibilidad a personas que lo necesitan, un tema sobre el que me gustaría profundizar ya que cada vez tiene una mayor relevancia en la sociedad actual.

Como conclusión, considero que este proyecto ha logrado cumplir los objetivos que se plantearon al inicio del desarrollo y se ha ofrecido una nueva experiencia de juego en el saturado mundo de los videojuegos, debido a que en la actualidad es muy complicado que una persona tenga la capacidad económica de apropiarse de uno estos dispositivos y por consiguiente las empresas actuales no valoran el desarrollo de videojuegos que hacen uso de estas tecnologías.

9. Bibliografía

[1] Asociación Española de Videojuegos (20 de abril de 2022). *El videojuego facturó 1.795 millones de euros en 2021, con una base superior a los 18 millones de usuarios en España.* http://www.aevi.org.es/videojuego-facturo-1-795-millones-euros-2021-una-base-superior-los-18-millones-usuarios-espana/

[2] EMOTIV EPOC X 14 Channel Mobile Brainwear.
https://www.emotiv.com/product/emotiv-epoc-x-14-channel-mobile-brainwear/

[3] Kosiński, J., Szklanny, K., Wieczorkowska, A. y Wichrowski, M. (2018). An Analysis of Game-Related Emotions Using EMOTIV EPOC. *2018 Federated Conference on Computer Science and Information Systems (FedCSIS).* pp. 913-917.

[4] Harrison, T. (2013). *The Emotiv mind: Investigating the accuracy of the Emotiv EPOC in identifying emotions and its use in an Intelligent Tutoring System*. [Tesis de Licenciatura, Universidad de Canterbury]. http://hdl.handle.net/10092/14867

[5] Unity
https://unity.com/es

[6] Unity Asset Store
https://assetstore.unity.com/

[7] Emotiv EPOC X
https://www.emotiv.com/epoc-x/

[8] Emotiv Unity Plugin
https://github.com/Emotiv/unity-plugin

[9] Emotiv Launcher
https://www.emotiv.com/emotiv-launcher/

[10] Cinemachine
https://unity.com/es/unity/features/editor/art-and-design/cinemachine

[11] NavMesh Documentation
https://docs.unity3d.com/ScriptReference/AI.NavMesh.html

[12] Chin, J.P., Diehl, V.A., Norman, K.L. (1988). Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface. *CHI'88: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. pp. 213-218.

[13] Sauro, J. (2015). SUPR-Q: A Comprehensive Measure of the Quality of the Website User Experience. *Journal of User Experience.* pp. 68-96.

[14] Emotiv Cortex documentation
https://emotiv.gitbook.io/cortex-api/

9.1. Unity Assets

A continuación, se detalla una lista de todos los *assets* que se han descargado y utilizado para llevar a cabo el desarrollo del juego:

150+ LowPoly Nature Models
https://quaternius.itch.io/150-lowpoly-nature-models

2D ITEMS SET - HANDPAINTED
https://assetstore.unity.com/packages/2d/gui/icons/2d-items-set-handpainted-210729

52 Special Effects Pack
https://assetstore.unity.com/packages/vfx/particles/spells/52-special-effects-pack-10419

Battle Wizard Poly Art
https://assetstore.unity.com/packages/3d/characters/humanoids/fantasy/battle-wizard-poly-art-128097

Cartoon FX Free
https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-free-109565

Cartoon FX Free Pack
https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-free-pack-169179

Creature: Cave Troll
https://assetstore.unity.com/packages/3d/characters/creatures/creature-cave-troll-115707

Fantasy Mushroom Mon
https://assetstore.unity.com/packages/3d/characters/creatures/fantasy-mushroom-mon-115406

Free Monster Bat
https://assetstore.unity.com/packages/3d/characters/free-monster-bat-158125

Free Mummy Monster
https://assetstore.unity.com/packages/3d/characters/free-mummy-monster-134212

Jammo Character | Mix and Jam
https://assetstore.unity.com/packages/3d/characters/jammo-character-mix-and-jam-158456

LowPoly Animated Monsters
https://quaternius.itch.io/lowpoly-animated-monsters

LowPoly Modular Dungeon Pack
https://quaternius.itch.io/lowpoly-modular-dungeon-pack

Meshtint Free Polygonal Metalon
https://assetstore.unity.com/packages/3d/characters/creatures/meshtint-free-polygonal-metalon-151383

Mini Legion Roc Golem PBR HP Polyart
https://assetstore.unity.com/packages/3d/characters/humanoids/fantasy/mini-legion-rock-golem-pbr-hp-polyart-94707

Modular Terrain Pack
https://fertile-soil-productions.itch.io/modular-terrain-pack

Nature Kit
https://www.kenney.nl/assets/nature-kit

NavMesh
https://github.com/Unity-Technologies/NavMeshComponents

Party Monster Duo Polyart PBR
https://assetstore.unity.com/packages/3d/characters/creatures/party-monster-duo-polyart-pbr-195698

POLYDesert
https://assetstore.unity.com/packages/3d/environments/landscapes/polydesert-107196

Procedural fire
https://assetstore.unity.com/packages/vfx/particles/fire-explosions/procedural-fire-141496

Retro Medieval Kit
https://www.kenney.nl/assets/retro-medieval-kit

Robot Sphere
https://assetstore.unity.com/packages/3d/characters/robots/robot-sphere-136226

RPG Monster Duo PBR Polyart
https://assetstore.unity.com/packages/3d/characters/creatures/rpg-monster-duo-pbr-polyart-157762

RPG Monster Partners PBR Polyart
https://assetstore.unity.com/packages/3d/characters/creatures/rpg-monster-partners-pbr-polyart-168251

RPG Poly Pack – Lite
https://assetstore.unity.com/packages/3d/environments/landscapes/rpg-poly-pack-lite-148410

Sentry Robot
https://assetstore.unity.com/packages/3d/characters/robots/sentry-robot-222899

Simple UI Elements
https://assetstore.unity.com/packages/2d/gui/icons/simple-ui-elements-53276

Space Kit
https://www.kenney.nl/assets/space-kit

Standard Assets (for Unity 2018.4)
https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-for-unity-2018-4-32351

Stone Monster
https://assetstore.unity.com/packages/3d/characters/stone-monster-101433

Ultimate Platformer Pack
https://quaternius.itch.io/ultimate-platformer-pack

Volumetric Lines
https://assetstore.unity.com/packages/tools/particles-effects/volumetric-lines-29160