UNIVERSIDAD FRANCISCO DE VITORIA

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA INFORMÁTICA

PROYECTO FINAL DE GRADO

MODALIDAD INGENIERÍA

Adaptación del nivel de dificultad en un videojuego a las características del jugador: una aplicación real de la Inteligencia Artificial.

Carlos Castillo González

Convocatoria de Julio 2023

CALIFICACIÓN DEL PROYECTO FINAL DE GRADO

CUALITATIVA:	
NUMÉRICA:	

Conforme Presidente:	Conforme Secretario:
Fdo.:	Fdo.:

Conforme Vocal:	Conforme Vocal:	Conforme Vocal:
Fdo.:	Fdo.:	Fdo.:

Lugar y fecha: Pozuelo de Alarcón, a ____ de _____ de 202__

Agradecimientos

En primer lugar, quiero agradecer a mis padres que, aunque ya no estén, ellos en vida hicieron el esfuerzo para que actualmente tenga la posibilidad económica de acceder a la educación necesaria y tener la posibilidad de obtener el título de Ingeniero Informático, les estaré eternamente agradecido.

Dar las gracias a mis hermanos y el marido de mi hermana por el apoyo que me ofrecieron cuando tomé la decisión de hacer un cambio de carrera, nunca olvidaré el momento en el que les conté mi decisión y siempre les agradeceré el apoyo que me dieron entonces y el que me han dado a lo largo de estos 5 años que llevo en la Universidad Francisco de Vitoria, y también por los momentos en que me ayudaban con una decisión difícil, como la de cuantas asignaturas cogerme ese año.

Agradecer a mis amigos de toda la vida y mis compañeros que, actualmente se han convertido en amigos que se que tendré para siempre, el apoyo, las charlas nocturnas, audios de WhatsApp de más de dos minutos, los momentos de estudio en grupo y porque no, las salidas y los momentos de relajación, sin ellos no habría tenido la fuerza como para terminar la carrera en el tiempo necesario.

Por último, agradecer a todos los profesores que me han ayudado durante la carrera, con tutorías, resolviéndome dudas y que incluso cuando no entendía algo se tomaban el tiempo necesario hasta que lo entendía.

Si no fuera por todos ellos no sería lo que soy hoy en día, y quería darles las gracias una vez más, aunque siempre les estaré agradecido.

Resumen

En el presente documento se detalla el proceso que se ha llevado a cabo para realizar el Proyecto Final de Grado (PFG), no solo la parte técnica, sino también la parte teórica, así como la investigación previa, los objetivos a cumplir y metodología utilizada. Para la parte técnica se tratan temas como las tecnologías utilizadas, solución técnica y los resultados obtenidos.

Con este proyecto se da una aplicación diferente de la rama subsimbólica de la Inteligencia Artificial (IA) al mundo de los videojuegos, dicha aplicación es la utilización de una red neuronal para ajustar el nivel de dificultad en base a las habilidades del jugador para ello ha sido necesario la creación de un videojuego sencillo y una red neuronal.

Palabras claves

Videojuego, Inteligencia Artificial, Red Neuronal.

Abstract

This document details the process that has been carried out to carry out the Final Degree Project (FDP), not only the technical part, but also the theoretical part, as well as the previous research, the objectives to be met and the methodology used. For the technical part, topics such as the technologies used, technical solution and the results obtained will be discussed.

This project gives a different application of the subsymbolic branch of Artificial Intelligence (AI) to the world of video games, this application is the use of a neural network to adjust the level of difficulty based on the player's skills for it has been necessary to create a simple video game and a neural network.

Keywords

Video game, Artificial Intelligence, Neural Network.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Lista de Acrónimos

Acrónimo	**Significado**
AI	Artificial Intelligence
IDE	Entorno de Desarrollo Integrado
FDP	Final Degree Project
fps	First-Person Shooter, Disparos en Primera Persona
FPS	Fotogramas Por Segundo
GDD	Game Desing Document, Documento de Diseño
IA	Inteligencia Artificial
N/A	No aplica
NPC	Non-Player Character, Personaje No Jugable
ONNX	Open Neural Network Exchange
PFG	Proyecto de Fin de Grado
VS	Visual Studio

1. INTRODUCCIÓN

Cada día la inteligencia artificial está tomando más peso en nuestro día a día con aplicaciones tan interesantes como ChatGPT, un modelo de lenguaje que cada vez más gente conoce y utiliza, o incluso DALL-E, la cual es capaz de generar imágenes nunca vistas con un simple texto o incluso coches que conducen solos. Estas son sólo un ejemplo de las posibilidades que tienen las redes neuronales en nuestro mundo actual.

Dentro del mundo de los videojuegos, aplicar una red neuronal no es tan común; si pensamos en los videojuegos como en un sistema cerrado en donde unas acciones llevan a unos resultados lo normal es utilizar Inteligencias Artificiales que reproduzcan eso, es decir, se utilizan soluciones como los árboles de decisión o máquinas de estados, puesto que es lo más preciso para reproducirlo, estas son las que se utilizan para dar vida a los NPC (Non-Player Character, Personaje No Jugable), y parte de la dificultad del videojuego generalmente viene dada por las capacidades de estos personajes.

Con este proyecto se ha querido dar una aplicación nueva y diferente de las redes neuronales en los videojuegos haciendo que, la dificultad de este se vea modificada por una IA, la cual es capaz de aumentarla, mantenerla o disminuirla en la partida dependiendo de las habilidades del jugador.

2. INVESTIGACIÓN PREVIA

La investigación realizada antes y durante el proyecto se divide en tres bloques distintos, el primero, las diferentes maneras que tienen los juegos para establecer la dificultad, en segundo lugar, los usos de la Inteligencia artificial en los videojuegos, y por último las tecnologías que finalmente se ha utilizado para la solución.

2.1. MANERAS QUE TIENEN LOS JUEGOS PARA ESTABLECER LA DIFICULTAD.

La dificultad en los videojuegos es algo que desde sus inicios es importante tener en cuenta desde el principio del desarrollo, no es algo que se pueda hacer rápido para quitártelo de encima [1], al fin y al cabo, forma parte de un bloque importante en el videojuego ya que, hay que incentivar a que el jugador quiera seguir jugando. Esto viene desde la época de las máquinas recreativas donde, se tenía que planificar muy bien el nivel de exigencia que se pedía en el videojuego para seguir gastando monedas en jugarlo [2].

La dificultad en un videojuego viene dada por la conocida curva de dificultad, esta es importante puesto que, un mal ajuste puede significar el aburrimiento o la frustración del jugador.

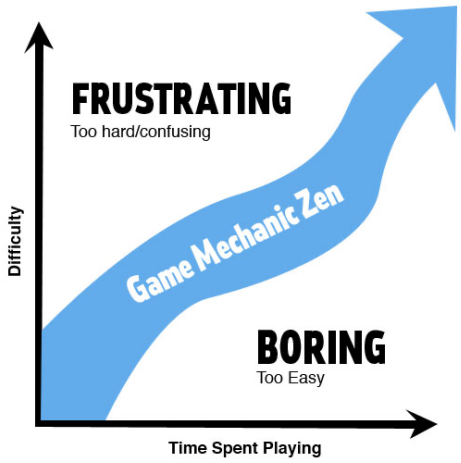

Figura 1 - La Curva de Dificultad en Videojuegos. Fuente: [3]

Actualmente, la mayoría de los videojuegos tienen diferentes maneras en las que, es el propio jugador el que decide si quiere que la partida sea más o menos desafiante.

2.1.1. Selección del nivel de dificultad

Seleccionar el nivel de dificultad desde el modo más fácil, siendo estos: modo historia, muy fácil o aficionado, dependiendo del juego, hasta el más difícil, como los modos realistas donde los enemigos son más inteligentes o se hace más complicado ganarles, es la manera más común dentro del mundo de los videojuegos para que el jugador decida si su experiencia de juego quiere que sea más relajada o difícil.

Con esta manera de seleccionar la dificultad los desarrolladores tampoco pueden modificar mucho la curva de dificultad, aunque esta sí que se vería un poco modificada (como en la Figura 2), puesto que deben tener en cuenta diferentes grados.

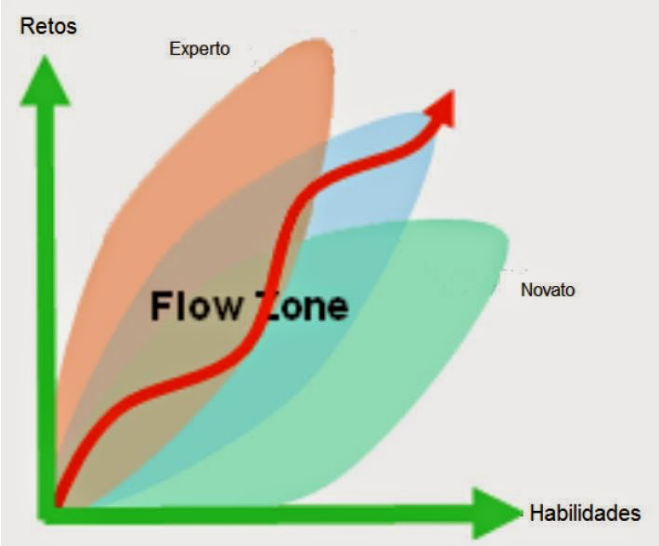

Figura 2: Curva de dificultad Reto/Habilidades. Fuente: [4]

Las técnicas que se utilizan generalmente en juegos de disparos en primera persona (fps, First-Person Shooter), como la saga *Call of Duty*, es reducirle la vida al personaje principal, es decir al jugador o que los enemigos puedan recibir menos daño, esto implica que te tengas que valer más del escenario y que no puedas ir corriendo [5].

Las técnicas usadas, aunque generalmente sean las ya mencionadas, dependen del juego, por ejemplo, en juegos de supervivencia, como el *Dead Space 2*, en el modo normal ofrece una experiencia equilibrada mientras que en el modo hardcore tienes que ser más inteligente y enfocarte más en el juego [6]. Otro ejemplo de este tipo de juegos es *The Last of Us 2*, que los diferentes niveles de dificultad se basan en el número recursos que hay en el escenario, la penalización por errores en el sigilo y la agresividad de los enemigos entre otras cosas, algo interesante que hicieron los desarrolladores en este título es que puedes elegir la dificultad de forma específica, como por ejemplo: enemigos difíciles pero escenario lleno de recursos, es decir puedes no tomar la decisión de forma global si no se quiere [7].

2.1.2. Sistema de dificultad dinámico

Con este tipo de sistemas predominan los juegos de mundo abierto, y hay diferentes maneras de abordarlo.

En primer lugar, los mundos con un escalado de nivel como los últimos *Assassin's Creed*, en donde el jugador puede dedicarse a explorar el mundo y todas sus posibilidades e ir avanzando por zonas de cada vez más nivel [8]. Esto plantea un problema y es que, el jugador puede recorrerse el mundo entero, hacer todas las misiones y recoger todos los tesoros y que al final las zonas con más nivel se le queden atrás y sean demasiado fáciles, para evitar eso, aunque no en toda su medida, lo que hacen es que las últimas zonas son del mismo nivel o parecido al que el jugador puede llegar a alcanzar.

En los juegos multijugador se hace algo parecido, con la diferencia que las misiones siempre van a ser más o menos de tu nivel y los otros jugadores que te encuentras en tu servidor también. Esto se hace para que, un jugador con un nivel muy alto no se encuentre con otro de nivel muy bajo y pueda quitarle todos los objetos, lo que provocaría una frustración y dejaría de jugar, lo descrito es en juegos de rol, pero en juegos como el *Call Of Duty online* se hace lo mismo para evitar la frustración.

Por otro lado, tenemos los juegos tipo *Dark Souls*, como *Elden Ring*, el cual tiene un sistema dinámico para que el juego siempre sea desafiante [9], es parecido al ejemplo anterior, pero en este el nivel no es visible por el jugador, y la manera que tiene este de ver si una zona es de su nivel o no es yendo a ella.

2.1.3. Sistemas dinámicos interesantes

Por último, los sistemas de los que hablaré a continuación también son considerados dinámicos, pero son diferentes a los vistos anteriormente, por lo que se ha decidido ponerlos en su propia categoría.

En primer lugar, el sistema némesis, que es un sistema patentado por Warner el cual hace que los enemigos tengan diferentes personalidades, miedos y fortalezas, además de que estos si acabas con ellos, se acuerdan del jugador y pueden volver a por ti con más nivel [10]. Esto mantiene al jugador en tensión, debido a que ese evento puede producirse en cualquier momento e incluso venirte más de un enemigo al mismo tiempo.

Finalmente, el sistema del *Alien: Isolation*, el cual se basa en juntar dos Inteligencias artificiales para que trabajen conjuntamente, la primera es la encargada de ver dónde y que está haciendo el jugador en cada momento y la segunda se trata de un árbol con 36 subrutinas las cuales deciden como atacar al jugador [11]. Lo que esto provoca es que el jugador no pueda utilizar siempre la misma estrategia para zafarse del ataque alien, y que no pueda explotar algo específico del juego con lo que le ha ido bien constantemente, puesto que el enemigo te impedirá que vuelvas a repetirlo.

2.2. USOS DE LA INTELIGENCIA ARTIFICIAL EN LOS VIDEOJUEGOS.

Todos los juegos utilizan la inteligencia artificial, aunque en diferentes áreas, estas, dentro del videojuego se usan para darle un movimiento y actos realistas a los NPC o generar de forma procedural escenarios casi infinitos.

La IA que se utiliza para los NPC es de la rama simbólica, más concretamente se utilizan algoritmos de forman máquinas de estados o árboles de decisión, como el caso que se ha visto anteriormente de Alien: Isolation, el cual usa un árbol con 36 subrutinas donde decide como atacar al jugador.

Con el paso de los años estos algoritmos han ido mejorando para darle más realismo a los NPC, desde *Pac-Man*, en donde cada uno de los fantasmas tenía una función diferente como la de ir siempre a la izquierda, hacer movimientos aleatorios o perseguir al personaje, pasando por juegos como el *Metal Gear Solid*, donde los enemigos seguían una rutina predefinida, pero se la saltaban si escuchaban algún sonido o veían al personaje [12], todas estas acciones vienen dadas por funciones que se basan en si ocurre un evento haz una acción. Es raro ver que la IA de los enemigos es una red neuronal, esto podría considerarse una aproximación a la solución propuesta en el proyecto. En este juego, es de defender tu torre, creado por un usuario llamado JLPM [13], el contrincante es una red neuronal la cual ha aprendido a jugar al juego y juega contra ti.

Un uso interesante de Deep Learning, es la generación procedural, el caso más llamativo que tenemos es el *No Man's Sky*, en donde nos encontramos un universo casi infinito generado de forma procedural, tanto las naves, como los planetas, las lunas o los animales dentro de cada planeta es generado de esta manera [14].

Por último, el uso más común de redes neuronales o machine learning en los videojuegos es para enseñarle a la IA a como jugar a un juego, es una práctica bastante extendida, usando aprendizaje por refuerzo o aprendizaje por imitación, aunque también se usan diferentes aprendizajes. Con esta aplicación tenemos muchos casos, uno de los más interesantes, es el de OpenAI, que crearon una inteligencia artificial multiagente que juega al escondite, y es capaz de obtener estrategias, incluso usando las físicas del entorno, para ganar [15].

Estos son solo algunos de los casos y diferentes soluciones que se han dado para el uso de las redes neuronales en los videojuegos.

2.3. TECNOLOGÍAS UTILIZADAS.

El proyecto está pensado para darle un uso diferente de las redes neuronales dentro de los videojuegos. Para poder llevarlo a cabo necesito dos bloques principales, los cuales se entrelazan para que la integración de la IA con el videojuego sea la más simple posible, y es por eso por lo que no se divide en títulos y se tratan conjuntamente en este apartado. El primer bloque es el uso de un motor de videojuegos accesible, y fácil de usar, que además me proporcione las herramientas necesarias para una fácil implementación, y el segundo es

escoger la herramienta que mejor se adapte a mis necesidades a la hora de crear, entrenar e implementar una red neuronal dentro del videojuego.

Para ello se pensaron en varias soluciones, todas ellas con licencias de uso gratuitas, y al final se habla de cuál es la que se escogió y por qué.

2.3.1. Game Maker Studio 2

En primer lugar, se pensó en Game Maker Studio 2 [16], esto fue así puesto que, es un *engine* especializado en el desarrollo de juegos en 2d, es fácil a la hora de la creación del videojuego, puesto que el desarrollo de este se basa en blueprints, que son pequeñas cajas visuales de código que se van conectando para formar funcionalidades completas [17], aunque también tiene la posibilidad de programar más a fondo con C++. Es una tecnología interesante, pero a la hora de programar la red neuronal tendría que haberla hecho, o bien en C++ o con Python e integrarla.

Finalmente, esta opción se descartó por el desconocimiento de la herramienta y la complejidad que tiene a la hora de programar o integrar una red neuronal.

2.3.2. Godot

Esta tecnología se descartó por el desconocimiento que tenemos de la herramienta, puesto que, si tenemos en cuenta que es un motor bastante interesante y usado, por la facilidad de desarrollas videojuegos en él podría ser interesante.

Con esta herramienta se podría haber llegado a la solución tanto del videojuego como de la red neuronal, la programación para el desarrollo dentro de esta herramienta es el uso de blueprints y GDScripts, los cuales el lenguaje usado es muy parecido al de Python y habría sido relativamente sencillo hacer un traspaso de la solución en Python a GDScript.

Godot [18] se descartó por el desconocimiento que se tiene de la herramienta y porque a la hora de mostrar resultados sería mucho más compleja.

2.3.3. Unity 3D

Finalmente, la herramienta escogida es la de Unity3D, y las razones son muy simples, en comparación con los otros dos motores propuestos anteriormente, de este ya se tiene una base sólida y el tiempo de aprendizaje sería mucho, no sería tanto en el *engine*, sino que sería en la parte de desarrollo en 2d.

El lenguaje que se trabaja con esta herramienta el C#, junto con el paradigma de la programación orientada a objetos de las tres propuestas este lenguaje es el que más conozco y el que más he utilizado.

Por último, Unity proporciona un paquete de machine learning propio con el que la creación, entrenamiento e implementación de la IA se hace muy sencilla.

Por estos tres motivos es que finalmente se escogió Unity para realizar el proyecto.

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Objetivo es demostrar que una IA de la rama subsimbólica, es decir una red neuronal es capaz de, dependiendo de las habilidades del jugador adaptar la dificultad a este.

3.2. LISTA DE OBJETIVOS ESPECÍFICOS

Para la realización del proyecto se definieron una serie de objetivos específicos que dividen el objetivo principal en objetivos más pequeños que cumplir.

Tabla 1: Descripción de Objetivos Específicos. Fuente: Elaboración Propia.

Objetivo Específico	Descripción
Obj1	Funcionamiento correcto de las funcionalidades definidas en el documento de diseño (GDD, Game Desing Document) del videojuego.
Obj2	Asegurar que el juego contiene el menor número de errores.
Obj3	Búsqueda de un método de aprendizaje adecuado para la red neuronal.
Obj4	Verificar si el entrenamiento de la red neuronal es correcto, es decir la IA entrena.
Obj5	Funcionamiento correcto de la red neuronal dentro del videojuego.
Obj6	Asegurar un correcto rendimiento del videojuego al completo, es decir, juego e IA funcionando conjuntamente.

3.3. MÉTODOS DE VALIDACIÓN

El método de validación para el objetivo general será probar si el nivel de dificultar se cambia en base a los resultados que el jugador obtiene dentro de la partida.

No se pudo meter a personas a probar el juego porque este no se pudo subir a ninguna plataforma online.

Tabla 2: Métodos de validación. Fuente: Elaboración Propia.

Objetivo Específico	Método de Validación
Obj1	La forma de validar este objetivo será probando el juego y todas sus funcionalidades en conjunto para ver que funcionan correctamente.
Obj2	El método de validación es el testeo exhaustivo de las funcionalidades, escenario y evaluación de físicas para evitar el menor número de errores.
Obj3	Investigar sobre los diferentes tipos de aprendizaje que podría tener la red neuronal, y probar diferentes configuraciones con el seleccionado. Mediante gráficas se verificará cual es la mejor configuración para el entrenamiento.
Obj4	Se verificará mediante gráficas, usando diferentes parámetros, ver con cuales ha entrenado la IA y con cuáles no.
Obj5	Introducir el modelo dentro del juego y probarlo a ver si los resultados que da la IA son correctos.
Obj6	Mediante las estadísticas de Fotogramas Por Segundo (FPS) comprobar si el rendimiento del juego al completo, es decir, el juego terminado con todas las funcionalidades y la red neuronal funcionando conjuntamente.

4. PLAN DE DESARROLLO DEL PROYECTO

4.1. METODOLOGÍA

Para el desarrollo de este proyecto se escogió una metodología en cascada, esta metodología al ser la más robusta, no solo para entregas, sino también para los requisitos, era la más adecuada para que no se produjeran desviaciones a la hora de desarrollar el videojuego.

Es bien sabido que en los desarrollos de videojuegos lo más normal es que surjan cambios y modificaciones y, por ende, desviaciones, con esta metodología reducía las posibilidades de mejoras innecesarias que no aportaban nada al desarrollo de la inteligencia artificial.

También es la más adecuada en cuanto a que, los requisitos principales tanto para el juego, como para la red neuronal estaban bien definidos desde el principio y no iban a sufrir cambios durante el desarrollo.

En la fase de análisis, se hizo hincapié en la investigación de los diferentes tipos de videojuegos que existían en 2D, y cual se ajustaba más con la solución pensada para la inteligencia artificial.

En la fase de diseño, se hicieron las especificaciones del videojuego, es decir, el tipo de juego, las diferentes dificultades, sistema de puntuación, personaje, escenario y enemigos. Por parte de la IA se definió el tipo de aprendizaje a utilizar, y al ser aprendizaje por refuerzo, las observaciones, premios y acciones a tomar.

Para la fase de desarrollo y verificación, se dividió en dos partes, la primera la del videojuego, en la que se desarrolló en base a *sprites* la parte visual y todas y cada una de las partes funcionales definidas en el diseño, y se verificó probando que no había errores. La segunda parte, fue la de la inteligencia artificial, la cual se programó, se entrenó y verificó que funcionaba correctamente dentro del juego.

Finalmente, para fase de mantenimiento, se hará con actualizaciones del juego y mejoras de la red neuronal.

4.2. TECNOLOGÍAS

4.2.1. Unity3D

Se escogió el motor de videojuegos de Unity3D, porque es una tecnología que conocía de haberla utilizado durante la carrera universitaria en las asignaturas del título propio de *Desarrollo de videojuegos*.

También es un motor de videojuegos altamente extendido y utilizado, con muchas funcionalidades y aplicaciones prácticas, como poder exportarlo a diferentes plataformas y no solo ordenador; tiene funciones para hacer multijugador, importar *assets*, realizar diagnósticos e incluso animaciones [19], además que la licencia es totalmente gratuita.

Es una herramienta potente y sencilla de utilizar para crear juegos en 2D, con un manual y un apoyo extenso de la comunidad de desarrolladores de videojuegos [20].

4.2.2. Visual Studio 2022 y lenguaje C#

Aunque con Unity se puede elegir entre otros entornos de desarrollo integrado (IDE) como Visual Studio Code, decidí escoger la herramienta de Visual Studio 2022 (VS) [21], puesto que también se conoce de la carrera, es sencilla y fácil de usar y la integración con Unity es instantánea y no requiere de mucha preparación como me ocurrió con el Visual Studio Code, de echo VS se descarga, instala e integra automáticamente al descargarse cualquier versión de Unity, y en comparación con la versión de 2019, que era la otra versión conocida, la versión de 2022 tiene mejoras como el de proporcionar mejor código que versiones anteriores.

Además, el lenguaje de C# es el que se utiliza en Unity y este IDE es la mejor opción a la hora de programar con este lenguaje, y paradigma de programación orientada a objetos.

4.2.3. Paquete MLAgents de Unity

Realmente para la red neuronal, al utilizar el motor de Unity, había dos opciones, programarla y entrenarla con Python e integrarla en el videojuego o usar el paquete de MLAgents [22] el cual facilita mucho el desarrollo y entrenamiento de la inteligencia artificial.

Obviamente por tema de facilidad de integración con el motor y desarrollo, aunque por detrás funcione Python esta era la mejor opción.

4.2.4. Tensorboard

Esta tecnología es propia de TensorFlow, y en realidad es opcional usarla puesto que, sin ella el entrenamiento y los resultados no cambian, realmente tensorboard para lo que ha servido en este proyecto es precisamente para visualizar los resultados mediante gráficas, gracias a ella se ha obtenido más datos que poder interpretar, y de una manera mucho más sencilla que por consola [23].

Los datos del entrenamiento si no utilizas esta herramienta, aparecen por la consola del Anaconda Prompt, que es la consola propia de Anaconda, que es la que ha permitido correr el entrenamiento. También los datos obtenidos por consola son mucho más escuetos y más complicado de interpretarlos, es mucho más sencillo de una manera visual, y es por eso por lo que se ha escogido esta tecnología.

4.2.5. Entornos, librerías, paquetes y ecosistemas

Finalmente, entre las tecnologías escogidas, las siguientes eran imprescindibles para el entrenamiento de la red neuronal y la creación del cerebro de esta.

Como entorno, se ha utilizado el entorno virtualizado que ofrece Python3 [24].

La librería ha sido PyTorch que está destinada a aplicaciones del machine learning, reconocimiento de imágenes y aprendizaje profundo [25].

Mlagents Python, es un paquete de la herramienta ML-Agents, que provee de un set de algoritmos de aprendizaje por refuerzo y aprendizaje por imitación [26].

Por último, se ha utilizado el Open Neural Network Exchange (ONNX) que es un formato abierto el cual representar modelos de machine learning, esto es lo que sería el cerebro de la IA que se introdujo en el proyecto [27].

4.3. PLAN DE DESARROLLO DEL PROYECTO

Cada paquete de trabajo viene dado por las diferentes fases descritas anteriormente.

4.3.1. PT 1. Análisis

Tabla 3: Tarea de investigación de diferentes tipos de juegos 2D. Fuente: Elaboración propia.

Código:	PT01-T01	Nombre:	Investigación de los diferentes tipos de videojuegos 2D
Descripción:	Se realizará una investigación de los diferentes tipos de juegos en 2D para conocer diferentes soluciones.		
Entradas:	No aplica (N/A).		
Salidas:	Tipo de videojuego a desarrollar.		
Actividades:	Realizar una investigación detallada de los diferentes tipos de videojuegos en 2D y ver cuál es el más adecuado para su desarrollo.		

Tabla 4: Tarea de investigación de diferentes aprendizajes para la IA. Fuente: Elaboración propia.

Código:	PT01-T02	Nombre:	Investigación de diferentes aprendizajes para la IA
Descripción:	Se realizará una investigación de las diferentes formas de aprender que tiene una IA y cuáles son los más usados en videojuegos.		
Entradas:	N/A.		
Salidas:	Tipo de aprendizaje a utilizar en la IA.		
Actividades:	Realizar un estudio de los diferentes tipos de aprendizaje y con ello escoger el más adecuado para el proyecto.		

4.3.2. PT 2. Diseño

Tabla 5: Especificaciones del personaje. Fuente: Elaboración propia.

Código:	PT02-T01	Nombre:	Especificaciones del personaje.
Descripción:	En base al tipo de juego escogido se definirán las acciones que puede realizar el personaje.		
Entradas:	PT01-T01.		
Salidas:	Lista de acciones que puede realizar el personaje.		
Actividades:	Describir las diferentes acciones que puede realizar el jugado.		

Tabla 6: Especificaciones del escenario. Fuente: Elaboración propia.

Código:	PT02-T02	Nombre:	Especificaciones del escenario.
Descripción:	En base al tipo de juego escogido se definirá como va a ser el escenario y que elementos tendrá.		
Entradas:	PT01-T01.		
Salidas:	Lista con los elementos del escenario y descripción de cómo será visualmente.		
Actividades:	Describir los diferentes elementos que tendrá el escenario y como va a ser este.		

Tabla 7: Especificaciones de los enemigos. Fuente: Elaboración propia

Código:	PT02-T03	Nombre:	Especificaciones de los enemigos.
Descripción:	En base al tipo de juego escogido se define los tipos de enemigos que tendrá y las acciones de estos.		
Entradas:	PT01-T01.		
Salidas:	Lista con los tipos de enemigos y sus acciones		
Actividades:	Describir los enemigos que habrá en el escenario y las acciones que podrán hacer.		

Tabla 8: Especificaciones del sistema de puntuación. Fuente: Elaboración propia.

Código:	PT02-T04	Nombre:	Especificaciones del sistema de puntuación.
Descripción:	Se definen las especificaciones de cómo será el sistema de puntuación.		
Entradas:	PT01-T01.		
Salidas:	Lista con el sistema de puntuación del videojuego.		
Actividades:	Describir el sistema de puntuación, como es y cómo se obtienen los puntos.		

Tabla 9: Especificaciones del sistema de dificultad. Fuente: Elaboración propia.

Código:	PT02-T05	Nombre:	Especificación del sistema de dificultad.
Descripción:	Definir cómo será el sistema de dificultad y los cambios que provocará en el escenario.		
Entradas:	PT01-T01.		
Salidas:	Lista con las diferentes dificultades y los cambios que estas producen.		
Actividades:	Describir y diseñar el sistema de puntuación junto con los cambios.		

Tabla 10: Especificaciones de la IA. Fuente: Elaboración propia.

Código:	PT02-T06	Nombre:	Especificaciones de la IA.
Descripción:	En base al tipo de aprendizaje escogido, se definirá como va a entrenar la IA.		
Entradas:	PT01-T02.		
Salidas:	Lista con las especificaciones de la IA.		
Actividades:	Describir y diseñar el método de aprendizaje de la IA.		

4.3.3. PT 3. Desarrollo y Verificación del videojuego

Tabla 11: Desarrollo del personaje. Fuente: Elaboración propia.

Código:	PT03-T01	Nombre:	Personaje.
Descripción:	Se desarrollarán las diferentes acciones que puede realizar el personaje durante la partida.		
Entradas:	PT01-T01 y PT02-T01.		
Salidas:	Código asociado al personaje.		
Actividades:	- Desarrollar el movimiento del personaje. - Desarrollar el sistema de disparo. - Desarrollar el sistema de vidas.		

Tabla 12: Desarrollo del personaje. Fuente: Elaboración propia.

Código:	PT03-T02	Nombre:	Escenario.
Descripción:	Se creará el escenario y se desarrollarán las diferentes funcionalidades asociadas a este.		
Entradas:	PT01-T01 y PT02-T02.		
Salidas:	Código asociado al escenario y escenario.		
Actividades:	- Creación del escenario. - Desarrollo del sistema de victoria/derrota. - Desarrollar el sistema de añadido de vidas al personaje.		

Tabla 13: Desarrollo de enemigos. Fuente: Elaboración propia.

Código:	PT03-T03	Nombre:	Enemigos.
Descripción:	Se desarrollarán las diferentes acciones que pueden realizar los enemigos junto con sus estadísticas.		
Entradas:	PT01-T01 y PT02-T03.		
Salidas:	Código asociado a los enemigos		
Actividades:	- Desarrollar el tipo de enemigo. - Desarrollar su propio sistema de movimiento. - Desarrollar su propio sistema de disparo. - Desarrollar su propio sistema de vidas.		

Tabla 14: Desarrollo del sistema de la puntuación. Fuente: Elaboración propia.

Código:	PT03-T04	Nombre:	Puntuación.
Descripción:	Se desarrollarán las diferentes funcionalidades asociadas al sistema de puntuación.		
Entradas:	PT01-T01 y PT02-T04.		
Salidas:	Código asociado a la puntuación.		
Actividades:	- Desarrollo de la puntuación en base a enemigos eliminados - Desarrollo de la puntuación en base a los premios obtenidos - Desarrollo de la puntuación en base a la victoria o derrota del jugador.		

Tabla 15: Desarrollo del sistema de dificultad. Fuente: Elaboración propia.

Código:	PT03-T05	Nombre:	Dificultad.
Descripción:	Desarrollo del sistema de dificultad.		
Entradas:	PT01-T01 y PT02-T05.		
Salidas:	Código del apartado de dificultad.		
Actividades:	- Desarrollar las diferentes dificultades. - Desarrollar los cambios producidos en el escenario en base a la dificultad.		

Tabla 16: Prueba del juego de la inteligencia artificial. Fuente: Elaboración propia.

Código:	PT03-T06	Nombre:	Prueba del juego sin la IA.
Descripción:	Se probará si el juego funciona correctamente y se probará también posibles fallos por físicas.		
Entradas:	Juego funcional.		
Salidas:	Todas las funcionalidades descritas funcionan correctamente y no tiene errores.		
Actividades:	Probar el juego tratando de romperlo tanto en las físicas como en las funcionalidades.		

4.3.4. PT 4. Desarrollo y Verificación de la IA

Tabla 17: Programación de la IA. Fuente: Elaboración propia.

Código:	PT04-T01	Nombre:	Programación de la IA.
Descripción:	Se desarrollará la inteligencia artificial.		
Entradas:	PT01-T02 y PT02-T06.		
Salidas:	Código de la IA.		
Actividades:	- Sistema de observaciones. - Sistema de recompensas a la IA. - Sistema de acciones de la IA.		

Tabla 18: Entrenamiento de la IA. Fuente: Elaboración propia.

Código:	PT04-T02	Nombre:	Entrenamiento de la IA.
Descripción:	Se entrenará a la IA con diferentes configuraciones.		
Entradas:	PT01-T02 y PT02-T06.		
Salidas:	Gráficas y modelo onnx.		
Actividades:	- Crear el entorno virtualizado. - Descargar las librerías y paquetes necesarios. - Comprobar con Tensorboard los resultados.		

Tabla 19: Prueba de la IA dentro del juego. Fuente: Elaboración propia.

Código:	PT04-T03	Nombre:	Prueba de la IA dentro del juego.
Descripción:	Se probará todo en conjunto, pero la prueba se centrará en la Inteligencia Artificial para comprobar que ha adquirido los patrones correctos en el entrenamiento.		
Entradas:	Juego e IA funcionales.		
Salidas:	La IA saca los resultados correctamente.		
Actividades:	Probar el juego junto con la IA para ver que esta saca correctamente las acciones en base a las observaciones.		

4.4. PLAN DE TRABAJO

En el diagrama de Gantt se estableció las entregas y los plazos para el desarrollo de cada una de las tareas.

Además, este diagrama está pensado para tener cierto margen sobre todo a la hora de la creación y entrenamiento de la Inteligencia artificial.

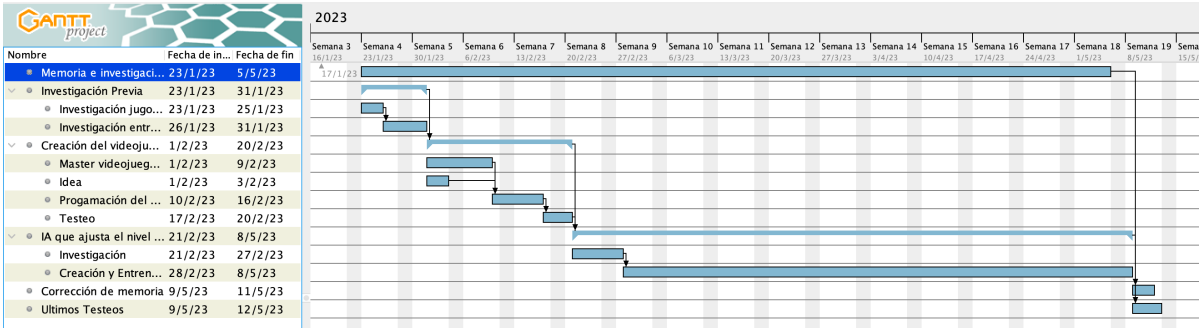

Figura 3: Diagrama de Gantt. Fuente: Elaboración Propia.

4.5. RECURSOS.

Los recursos humanos necesarios para la realización del proyecto son los siguientes:

- Jefe de Proyecto a 18€/hora de trabajo
- Diseñador de niveles a 10€ la hora de trabajo
- Diseñador Gráfico a 10€ la hora de trabajo
- Diseñador de la red neuronal a 12€ la hora de trabajo
- Documentador a 8€ la hora
- Equipo de desarrollo, a 10€ la hora de cada uno de los componentes
- Equipo de pruebas, a 7€ la hora de cada uno de los componentes

Aunque se haya tenido en cuenta diferentes roles para el proyecto, este es un proyecto realizado por una sola persona por lo que todos los roles y recursos fueron cubiertos por esta.

Cierto es que si realmente hubiera habido un equipo amplio de trabajo habría facilitado y agilizado el progreso, sobre todo a la hora de desarrollar diferentes funcionalidades, las cuales una vez tenido un consenso podrían haberse hecho de forma paralela.

En el siguiente apartado se explica con más detalle los costes, tanto de personal como de material.

4.6. COSTES

Es necesario distinguir de manera clara entre los costos relacionados con el material y el personal necesarios para llevar a cabo el proyecto. Además, debemos considerar un costo que no implica un gasto económico directo, pero que desempeña un papel crucial: el tiempo. Un retraso en el proyecto puede generar costos adicionales e incluso llevar a la pérdida de este, por lo tanto, es fundamental cumplir con el plazo establecido y lo acordado.

En cuanto a los materiales necesitamos lo siguiente:

Tabla 20: Costes Materiales. Fuente: Elaboración Propia.

Nombre del recurso	Precio	Unidades	Total
Oficina	30€/m2	25m2	3.750€
Mobiliario de oficina	1.000€	1	1.000€
Equipos informáticos	2.000€	3	6.000€
Licencias de sistema operativo	200€	3	600€
Licencias de software	200€	3	600€
Servidor de base de datos	60€/mes (pagado anualmente)	1	360€
		Total	12.310€

La estimación para estos costes es de 5 meses, desde febrero hasta junio. En relación con el coste humano del proyecto, vamos a detallar en una tabla los costes correspondientes, teniendo en cuenta los recursos mencionados anteriormente. Para esto, calcularemos el coste por hora de cada miembro involucrado en el proyecto, considerando que trabajarán x horas al día durante los n días laborables de la semana (ajustar según cada caso).

Dado que se espera que el proyecto se desarrolle hasta mayo, tendremos un total de 4 meses efectivos de trabajo para los miembros. Sin embargo, incluiremos los costes hasta junio, al igual que hicimos con los recursos materiales. Por lo tanto, calcularemos un total de 80 horas al mes, durante un período de cinco meses, lo que suma un total de 300 horas.

Tabla 21: Costes Humanos. Fuente: Elaboración Propia.

Nombre del recurso	Precio (€/hora)	Horas	Total (€)
Jefe de proyecto	18	400	7.200
Diseñador de niveles	10	100	1.000
Diseñador Gráfico	10	150	1.500
Diseñador de la red neuronal	12	200	2.400
Documentador	8	120	960
Equipo de desarrollo	10	300	3000
Equipo de pruebas	7	100	700
		Total	16.760

El coste definitivo del proyecto, sumando capital humano y capital material hace un total de 29.070 euros para su elaboración. Además, debemos incluir un margen de 2.000 euros para posible formación del equipo o algún imprevisto en cuanto al hardware o respecto a los servidores. El coste definitivo con este margen quedaría en 31.070 euros sin IVA incluido.

Tabla 22: Costes Definitivos. Fuente: Elaboración Propia.

Resumen General	Total (€)
Materiales	12.310
Recursos humanos	16.760
Imprevistos	2000
Total	31.070
IVA (21%)	6.524,7
Total (IVA incluido)	37.594,7

4.7. CONDICIONANTES Y LIMITACIONES

Durante el proyecto han surgido diferentes limitaciones, generalmente todos están relacionados con la IA y su entrenamiento, la más importante ha sido la falta de datos para poder entrenar a la red neuronal, para obtener una gran cantidad de datos tuve que generarlos aleatoriamente de manera que esto ha provocado que la IA no funcione a la perfección.

Otra de las limitaciones, ha sido que mi solución y me proyecto al ser novedoso no había nada en lo que me pudiera apoyar, más que aprender de otros proyectos que usaban el paquete de MLAgents de Unity, y de la documentación, la cual se basa en un repositorio de GitHub y un tutorial del área de aprendizaje de Unity.

La complicación del uso del paquete de MLAgents y el desconocimiento de este y de otros tipos de aprendizaje como el de aprendizaje por imitación han limitado el entrenamiento de la inteligencia artificial.

Por último, tener que trabajar y al mismo tiempo ir a la universidad y estudiar, ha supuesto tener que invertir un menor número de horas, para que el proyecto pudiera tener, en el caso del videojuego, diferentes escenarios o enemigos y en el caso de la IA un entrenamiento mejor.

5. DESARROLLO DE LA SOLUCIÓN TÉCNICA

5.1. PT 1. ANÁLISIS

En este primer paquete de trabajo se realizaron las tareas de investigación de los diferentes juegos y los diferentes aprendizajes de la IA.

En la primera, la de investigar los diferentes tipos de juegos que existen en formato 2D, se hizo una búsqueda de los tipos de juegos en 2D que existen y de sus mecánicas, estos son, por ejemplo: plataformas, acción y aventura, de rol, de estrategia, juegos de puzles y muchos más.

De entre todos los tipos de juegos se decidió por optar por aquellos que eran más sencillos de implementar diferentes niveles, en si en todos se acaba pudiendo porque depende del juego que quieras hacer, pero esto ayudó a reducir la lista y a quedarnos con juegos de plataformas, acción y aventura y estrategia, puesto que los demás implementar una IA que aumentara o disminuyera la dificultad iba a ser demasiado complejo.

Finalmente, el resultado de esta se optó por un tipo de juego de acción plataformas, del estilo de la saga *Metal Slug* [28], este nos permitía hacer un sistema de puntuación y dificultad que fueran sencillos de implementar y que fueran acorde con la solución de la red neuronal.

Por otro lado, había que investigar qué tipos de aprendizaje admitía el paquete de MLAgents, de entre los cuales están, aprendizaje por refuerzo, aprendizaje por imitación y aprendizaje supervisado.

Se optó por el aprendizaje por refuerzo, porque la herramienta está diseñada principalmente para este tipo de aprendizaje, además con este tipo de aprendizaje se nos permitía controlar las acciones que toma el agente, las observaciones que hace y las recompensas que este recibe.

Una vez se tenía pensado el tipo de juego y el tipo de aprendizaje que iba a tener la red neuronal se pasó a diseñas los aspectos de cada una de estas.

5.2. PT 2. DISEÑO

En el paquete de diseño se diseñó de manera exacta todos los elementos a necesitar para el videojuego y todas las funcionalidades, se optó por no hacer un juego muy complejo, puesto que no era necesario y el desarrollo de este no podía llevar más tiempo de lo necesario.

En un primer momento, se definió todas las especificaciones principales del videojuego: personaje, escenario, enemigos, sistemas de dificultad y puntuación.

Para el personaje se tomó en cuenta las acciones que hacen estos en los videojuegos, como andar, saltar, correr, disparar, puesto que es estilo *Metal Slug*, se descartó el correr porque se vio innecesario este tipo de acción, además se definió los controles del personaje. También se definió la vida que iba a tener y el daño que iba a recibir.

Los enemigos, en un principio iban a ser dos, uno que aguantara más, pero hiciera menos daño y otro que aguantara menos, pero hiciera más daño, finalmente esto se descartó y optó por un solo tipo de enemigo, esto fue así porque el sistema de dificultad modifica la vida de los enemigos y la aumenta o disminuye en base a esta, eso implicaba más tiempo de desarrollo del videojuego.

Para el escenario se hizo un esbozo de los elementos que iba a tener a parte de los enemigos y el personaje principal, estos serían: vidas, premios y la puerta de la victoria. También se pensó en cómo hacerlo gráficamente y se ideó dividirlo por zonas para dar sensación de un mundo profundo.

Por último, relacionado con el videojuego se definió el sistema de puntuación, y se escogió un sistema de puntuación por estrellas, ya que, así era más sencillo de decidir cuando un jugador era muy habilidoso para ese nivel, estaba en su nivel o era demasiado complicado. El sistema se estrellas se basó en conseguirlas en base a los premios obtenidos, enemigos eliminados o si te has pasado el nivel o no.

Finalmente se diseñó las recompensas, acciones y observaciones que iba a tener la red neuronal, eso es así, puesto que el paquete de MLAgents es como funciona. Las observaciones se harían a la dificultad y al número de estrellas que obtiene el jugador durante la partida. Las recompensas se definieron en base a si la acción que escogía la IA, es decir, poner el nivel de dificultad en fácil, medio o difícil, era la decisión correcta o no.

Con todo esto definido, es decir, todas las tareas que se tratan en este paquete de trabajo terminadas y con el resultado del listado de cada una de ellas, se pudo dar paso al comienzo del desarrollo del proyecto, comenzando con el videojuego, y una vez este esté terminado y comprobado que todas las funcionalidades funcionan correctamente se dio paso al desarrollo, entrenamiento y prueba de la red neuronal.

5.3. PT 3. DESARROLLO Y VERIFICACIÓN DEL VIDEOJUEGO

El desarrollo del videojuego, aunque en un principio no iba a ser así y se iba a hacer con las figuras base que tiene Unity, se hizo en base a *sprites*, estos se obtuvieron de un paquete

gratuito y libre de uso llamado *Jojo Jambo Jungle Sprite Pack* [29]. Esta forma de trabajar era nueva, pero fue más sencillo para el desarrollo de las físicas.

Se programaron las diferentes funcionalidades ya definidas anteriormente. Estas partes de código tienen una diferencia de dificultad bastante abismal, siendo las más sencillas, clases con apenas diez líneas y otras con 70 líneas de código.

Lo más complicado de esta parte fue la conexión entre diferentes interacciones que tenían por ejemplo los enemigos con los premios, los cuales hacían de barrera y los enemigos no podían ver al personaje.

En un primer momento los enemigos iban a ser torretas, pero al final de decidió por hacer que se movieran a cierta distancia del personaje, y que sólo dispararan si estos lo veían, para hacer la visión de los enemigos se utilizó un rayo que sale del enemigo, y que tiene cierta distancia y es la colisión con el personaje la que activa al enemigo para perseguir al jugador.

Otro de los desarrollos importantes fue la parte visual del escenario, la cual se utilizó una herramienta de dibujo de mosaicos propia del motor del videojuego, esto es algo que no habíamos hecho nunca y era una práctica totalmente novedosa.

Finalmente, una vez todas las funcionalidades habían sido probadas poco a poco durante el desarrollo, se hizo la prueba definitiva de que todas funcionaban bien en conjunto. Después de algunos errores encontrados, como la puntuación o la vida del personaje y de haberlos subsanado, se pasó a la prueba de romper el juego, hacer cosas inesperadas para ver donde falla. Los errores encontrados en esa parte fueron por parte de las físicas, ya que hay veces que si el elemento va muy rápido el fotograma del motor no es capaz de captar esa colisión y se la salta. Para arreglar esto, se pusieron más plataformas en el escenario y evitar caídas a muy alta velocidad y además un botón para reiniciar el nivel en case de que esto ocurra.

Este botón no afecta a la puntuación ni la dificultad, y por ende no afecta a la red neuronal.

Los resultados de las tareas de este paquete de trabajo fueron los códigos que se crearon en relación a cada una de las tareas definidas, es decir, a cada uno de los desarrollos que tuvieron relación con el videojuego.

5.4. PT 4. DESARROLLO Y VERIFICACIÓN DE LA IA

El último paquete del desarrollo, una vez el juego funcionaba correctamente, fue el del desarrollo, entrenamiento y prueba de la IA.

Este paquete fue el más complicado y extenso no tanto de desarrollar sino también de entrenar, al fin y al cabo, se necesitan muchas horas de prueba y error hasta dar con el modelo idóneo para que funcione.

En un primer momento se hizo el script el cual iba a usar la IA para entrenar, en ese se programaron las observaciones que tendría, las acciones que realizaría y las recompensas obtenidas si ha tomado correctamente la decisión.

También por la falta de datos se tubo de desarrollar un script que generara de forma automática y aleatoria los diferentes estados en los que se encontraría el jugador en la partida, es decir, si ha ganado o no y el número de estrellas que obtiene. Este script, trabajando en conjunto con el propio del agente de inteligencia artificial, es como se entrenó la red neuronal.

Para esta parte también fue necesario crear un entorno virtualizado de Python, y descargar e instalar en el entorno las librerías necesarias para que entrenara, generara datos, se visualizaran las gráficas, se conectara con Unity y generara el modelo.

Para una vez funcionando todo para el entrenamiento y la generación de diferentes modelos, para posteriormente escoger el óptimo, se tuvo que probar diferentes configuraciones en las recompensas que recibía la IA. Esto fue un proceso largo puesto que cada entrenamiento duraba entre 45 minutos y 1 hora, lo bueno es que con Tensorboard fuimos capaces de ver el proceso de entrenamiento de una manera sencilla y al instante.

Finalmente, con los diferentes modelos generados, tuvimos que probarlos en el juego para ver cuál era el mejor, es decir, cual había obtenido las reglas necesarias para hacer las acciones correctas durante el juego.

Los resultados de las tareas de este paquete de trabajo fueron los códigos que se crearon en relación con cada una de las tareas definidas, es decir, a cada uno de los desarrollos que tuvieron relación con la IA. Además, como resultado de las tareas PT04-T02 y PT04-T03 se obtuvieron gráficas y datos que se comentará en el apartado de resultados.

6. RESULTADOS

6.1. RESULTADOS DEL OBJETIVO PRINCIPAL

El objetivo principal de este proyecto se basaba en hacer una red neuronal capaza de adaptar la dificultad de un juego a las habilidades del jugador.

Los resultados obtenidos fueron, que la IA es capaz de aumentar o disminuir la dificultad en base a los resultados, es decir si estas en una dificultad fácil y consigues tres estrellas la red neuronal cambiará la dificultad a una mayor, a difícil, y si en difícil pierdes te devolverá a fácil, en el caso de que en difícil obtengas menos de 3 estrellas te mantiene esa dificultad.

Después del desarrollo y los resultados obtenidos, finalmente se puede decir que el objetivo se ha cumplido satisfactoriamente, y que se ha generado un modelo de machine learning el cual es capaz de aumentar o disminuir la dificultad durante la partida en base a las habilidades del jugador.

6.2. RESULTADOS DE OBJETIVOS ESPECÍFICOS

Los resultados específicos se dividen en tren bloques que se pueden ver a continuación, dentro de cada uno de ellos estarán los objetivos específicos.

6.2.1. Funcionamiento correcto del videojuego

Dentro de los objetivos específicos que hacen referencia al funcionamiento correcto de las funcionalidades del videojuego, los cuales serían el Obj1 y Obj2, se puede decir que se han cumplido con éxito, el juego se probó con todas las funcionalidades al mismo tiempo y este funciona correctamente y, teniendo en cuenta problemas irresolubles con las físicas encontrados a la hora de hacer un testeo exhaustivo, se ha aportado una solución.

El juego cumple con las especificaciones de este y en conjunto no hay errores a la vista entre sus funcionalidades a la hora de jugarlo.

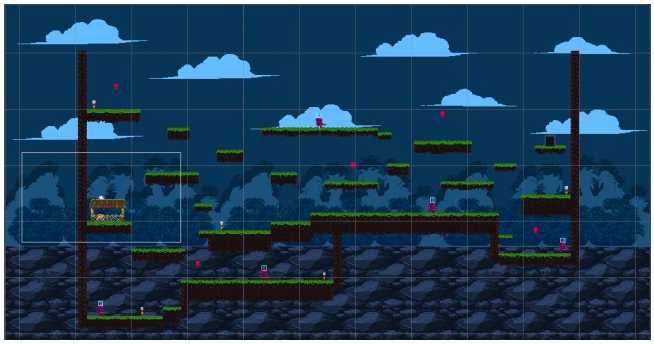

Figura 4: Escenario del videojuego. Fuente: Elaboración Propia.

6.2.2. Funcionamiento correcto de la IA

Los objetivos específicos que se hacen referencia a este bloque son el Obj3, Obj4 y Obj5, estos tienen en cuenta haber escogido una buena manera de que la red neuronal aprenda y utilizar los mejores parámetros para que el entrenamiento de esta sea óptimo.

Tras la investigación de los diferentes modelos de aprendizaje que ofrece el paquete MLAgents de Unity, se pudo determinar que el idóneo era el aprendizaje por refuerzo, ya que, la herramienta está diseñada para que ese sea su tipo de aprendizaje.

Se utilizaron diferentes parámetros y configuraciones para ver cuáles eran las mejores y con cuales la IA entrenaba correctamente.

A continuación, se pueden ver gráficas que demuestran que la IA ha entrenado y lo ha hecho correctamente en base a los diferentes parámetros y configuraciones.

En estas se pueden ver las diferentes configuraciones tratadas y que modelos han conseguido entrenar y cuales no, por ejemplo, la línea que aparece en color naranja, la cual no aumenta en esta primera figura (Figura 5) y no disminuye en las siguientes (Figura 6 y Figura 7), quiere decir que el modelo no está consiguiendo entrenar, por eso se descartó al instante.

Por otro lado, las demás líneas sí que aumentan, por lo que son modelos que sí que han entrenado. El mejor modelo de todos es donde la línea sale en verde, aunque en la figura 5 no sea el que se acerca más a 1, en la figura 6 sí que es la que más disminuye, lo que indica que no obtiene valores negativos del entrenamiento.

Este modelo es el que finalmente se ha utilizado para introducirlo en el videojuego, se probó y dentro de este es el que mejores resultados daba, es decir, es el que mejor tomaba decisiones de si hay que aumentar, mantener o disminuir en base a los resultados en el juego.

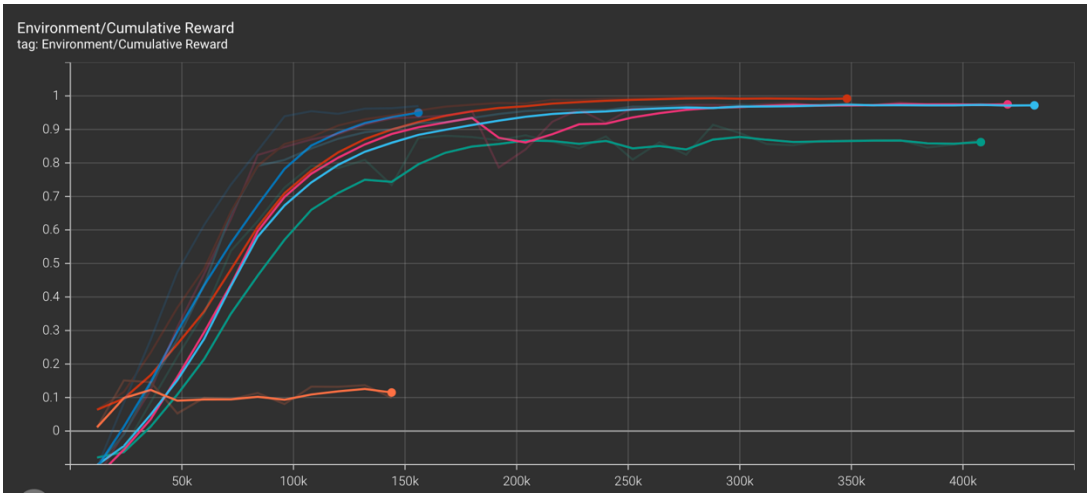

Figura 5: Recompensa Acumulativa. Fuente: Elaboración Propia.

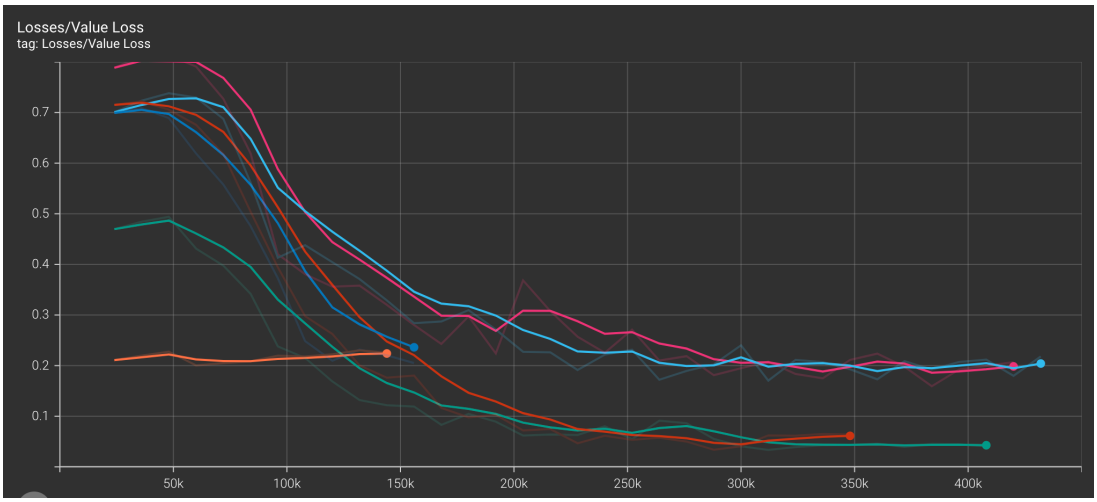

Figura 6: Pérdida de Valor. Fuente: Elaboración Propia.

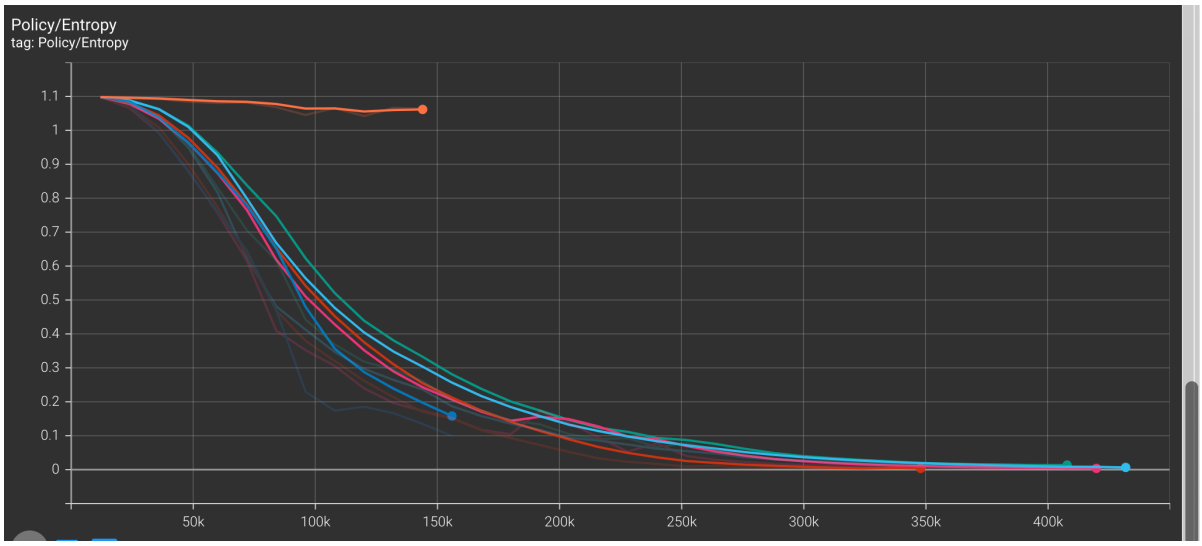

Figura 7: Entropía. Fuente: Elaboración Propia.

6.2.3. Rendimiento correcto del juego junto con la IA

Finalmente, el Obj6, también se puede decir que se ha cumplido con éxito, uno de los miedos que se tuvo a la hora de realizar este proyecto, o cualquier videojuego es que el rendimiento de este no vaya bien y los fotogramas por segundo sean menores a 30, aunque hoy en día se nota cuando un juego va a menos de 60 FPS.

Podemos decir que el rendimiento del videojuego es totalmente aceptable puesto que incluso dentro del motor de Unity, el cual consume más fotogramas, se puede ver que va a unos 70 FPS.

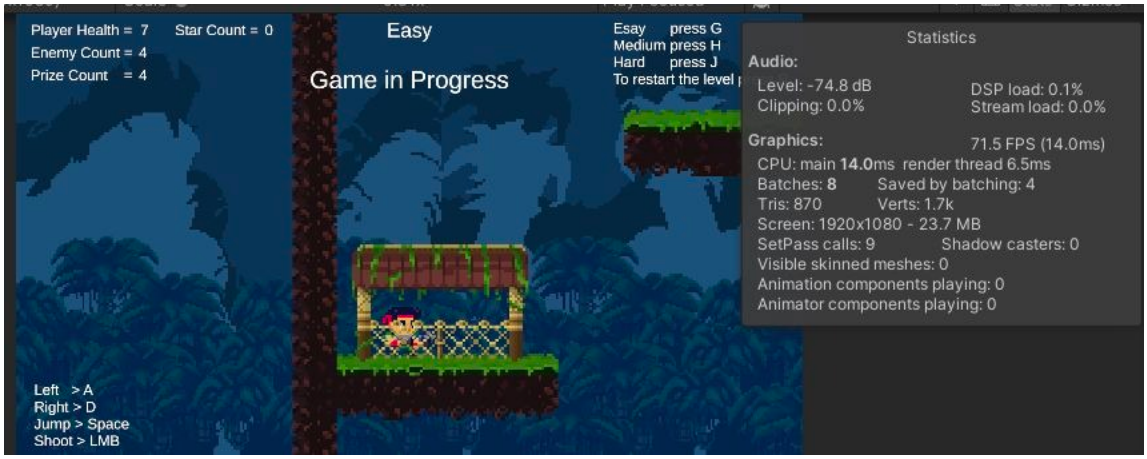

Figura 8: Rendimiento del Videojuego. Fuente: Elaboración Propia.

7. IMPLICACIONES ÉTICAS E IMPACTO SOCIAL

7.1. INTRODUCCIÓN

En este apartado se tratarán las implicaciones éticas y el impacto social del proyecto, así como para quien está dirigido y las responsabilidades que se tiene dentro del campo de los videojuegos y la inteligencia artificial.

Actualmente vivimos en una sociedad donde la inteligencia artificial y los videojuegos están cada vez más presentes en nuestra vida, hoy en día quien no ha oído hablar de ChatGPT o los coches con conducción autónoma y quien no ha tenido, por lo menos, algún juego en su teléfono móvil.

Hay juegos que no están pensados para un público que lo que quieres es una experiencia equilibrada, sino que buscan desafiar a los jugadores al máximo, casi llegando al punto de frustración de estos.

Por último, las IAs, como ya se ha comentado, cada vez más presentes en nuestro día a día ofrecen soluciones a problemas que a simple vista parecen demasiado complejos, y ofrecen a las personas facilidades en ciertos aspectos diarios.

7.2. DESARROLLO

La aportación que da este proyecto a la sociedad al introducir una red neuronal para controlar la dificultad es brindar a los jugadores una experiencia equilibrada en base a sus habilidades.

Esta solución ofrece una mayor accesibilidad a todo tipo de personas a introducirse, de una manera amigable, en el mundo de los videojuegos, puesto que a la inteligencia artificial no le importará si eres más o menos habilidoso en el juego, esta adaptará el nivel de dificultad para que se ajuste al jugador.

También dentro del campo de la IA ofrece una aplicación real de estas, y esto puede llevarse a proporcionar otros tipos de soluciones reales de la inteligencia artificial.

Pero como en todo proyecto, y sobre todo los relacionados con la tecnología, esto es un arma de doble filo, puesto que el acomodamiento y sentirse a gusto en un videojuego puede llevar

a problemas serios de adicción con estos, y puede llevar a un deseo descontrolado de jugar al videojuego por un gran número de horas [30].

En nuestras manos como ingenieros está en crear un sistema que detenga el juego si llevas demasiadas horas seguidas jugando.

Otro de los problemas éticos que hay en torno a los videojuegos, son las compras, o micro transacciones dentro de estos, en nuestra solución no se va a buscar métodos de pago dentro del juego, obviamente el juego no va a ser gratuito, sino no se podría mantener este y no se podrían hacer actualizaciones del mismo.

Por la parte de las inteligencias artificiales, tenemos con esta solución se tuvo en cuenta que la red neuronal no pudiera hacer modificaciones dentro del juego, más que el cambio de dificultad, esto viene dado a que si no controlamos esto la IA podría desarrollar métodos para hacer trampa y que el jugador se sienta frustrado.

7.3. CONCLUSIONES

Aunque dentro del mundo de los videojuegos existan adicciones, también hay cosas positivas, como el entretenimiento que ofrecen.

Aun habiendo varios impactos negativos referente a los videojuegos, la solución que se da con este proyecto y las medidas que tendríamos para paliarlos, podemos concluir que es un proyecto éticamente viable y que puede ofrecer un gran valor a la sociedad, sobre todo en el ámbito de la accesibilidad a los videojuegos y un valor positivo al mundo de la aplicación de inteligencias artificiales en problemas reales.

8. CONCLUSIONES

La realización del proyecto, en base a lo que se propuso y a lo que se ha conseguido no ha sido tarea fácil, ha requerido de muchas horas de estudio y entendimiento de herramientas totalmente nuevas con las que no se estaba familiarizado, aunque finalmente se ha cumplido, aunque claro está que, si se realizara junto con expertos en el sector del videojuego y la inteligencia artificial, se podrían haber conseguido cosas mucho más potentes.

Aún con todo esto, el proyecto viene de una idea innovadora, la cual no se ha puesto antes en práctica, no es solo un videojuego y ya está, es una nueva aplicación del machine learning dentro del mundo de los videojuegos, y las posibilidades que estas pueden llegar a ofrecer en un futuro al sector.

Para concluir, desde un punto de vista más personal, el hecho de que este proyecto me haya hecho enfrentarme a un problema real, y tener que buscar la solución por mi cuenta ha sido desafiante y enriquecedor. También haberme acercado más los grandes mundos de los videojuegos e inteligencia artificial con herramientas y técnicas, que ni si quiera conocía, me ha parecido fascinante.

8.1. EVOLUCIÓN DEL PROYECTO

Por las dos partes, tanto por el videojuego, como por la red neuronal, puede llegar a escalar tanto como se quiera.

El videojuego se ha implementado un único escenario con un único tipo de enemigos, se podrían crear un número grande de escenarios, de tipos de enemigos, de mundos e incluso se podrían ampliar los niveles de dificultad añadiendo que incluso los enemigos estén controlados por una red neuronal, que estos parezcan más inteligentes.

Podrían hacerse diferentes armas, armaduras y aumentos para el personaje principal. Básicamente los planes de futuro del videojuego son enormes, puesto que aún se puede desarrollar mucho más.

Por el lado de la IA, esta puede mejorar con más datos, más partidas, diferentes entrenamientos ayudándose conjuntamente, e incluso implementar, como ya he dicho antes diferentes redes neuronales a los enemigos o usarlas para crear diferentes escenarios.

Por último, como plan de futuro sería subir el juego a diferentes plataformas y promocionarlo para que la gente lo pruebe, pero claro antes de esto el juego debería de estar mucho más desarrollado junto con una inteligencia artificial que funcionase a la perfección.

9. OTROS MÉRITOS DEL PROYECTO

El mérito principal del proyecto ha sido la posibilidad de utilizar el paquete de MLAgents de Unity, este es un paquete que se utiliza mucho para hacer que las IAs aprendan a jugar a juegos.

Pero otros méritos adicionales con los que no contaba, ha sido el aprender a desarrollar juegos en 2D con sprites, animaciones y mapa de mosaicos, siendo estas de libre licencia para poder usarse, y que en un principio no contaba con un aspecto tan bonito del juego.

Se han utilizado librerías de software libre como ONNX o PyTorch para el entrenamiento de la red neuronal.

Finalmente, el proyecto me brinda la posibilidad de seguir trabajando en el juego y en la inteligencia artificial para poder seguir aprendiendo y llegar a sacarlo en diferentes plataformas.

10. BIBLIOGRAFÍA

[1] I. Fdez, «Xataka,» Xataca, 23 Septiembre 2018. [En línea]. Available: https://www.xataka.com/videojuegos/ponmelo-dificil-como-se-estructura-la-dificultad-en-un-videojuego. [Último acceso: Marzo 2023].

[2] F. J. Brenlla, «Meristation,» Meristation, 9 Noviembre 2020. [En línea]. Available: https://as.com/meristation/2020/11/09/reportajes/1604907622_249520.html. [Último acceso: Marzo 2023].

[3] J. García, «IGN España,» IGN, 21 Abril 2015. [En línea]. Available: https://es.ign.com/reportaje/92847/feature/la-curva-de-dificultad-en-videojuegos. [Último acceso: Marzo 2023].

[4] J. Mairena, «Videojuegos Accesibles,» 21 Septiembre 2014. [En línea]. Available: http://www.videojuegosaccesibles.es/2014/09/ajuste-dinamico-de-la-dificultad-en.html. [Último acceso: Marzo 2023].

[5] D. Rodíguez, «Hobbyconsolas,» Hobbyconsolas, 14 Diciembre 2019. [En línea]. Available: https://www.hobbyconsolas.com/listas/juegos-son-imposibles-completar-modo-dificil-529339. [Último acceso: Marzo 2023].

[6] D. Rodríguez, «Hobbyconsolas,» Hobbyconsolas, 14 Diciembre 2019. [En línea]. Available: https://www.hobbyconsolas.com/listas/juegos-son-imposibles-completar-modo-dificil-529339. [Último acceso: Marzo 2023].

[7] J. Cano, «Vandal,» Vandal, 1 Junio 2020. [En línea]. Available: https://vandal.elespanol.com/noticia/1350734888/the-last-of-us-parte-ii-asi-son-sus-niveles-de-dificultad-completamente-personalizables/. [Último acceso: Marzo 2023].

[8] F. J. Brenlla, «La dificultad en los videojuegos: Evolución, variación e importancia,» MeriStation, 9 Noviembre 2020. [En línea]. Available: https://as.com/meristation/2020/11/09/reportajes/1604907622_249520.html. [Último acceso: Marzo 2023].

[9] «Hay Un Sistema De Dificultad Dinamico en Elden Ring,» Nucleo Visual, [En línea]. Available: https://nucleovisual.com/hay-un-sistema-de-dificultad-dinamico-en-elden-ring/. [Último acceso: Marzo 2023].

[10] J. Tones, «Xataka,» Xataka, 9 Febrero 2021. [En línea]. Available: https://www.xataka.com/videojuegos/que-implica-que-nadie-pueda-usar-sistema-juego-como-sombras-mordor-asi-se-registra-mecanica-videojuego. [Último acceso: Marzo 2023].

[11] M. Gómez, «Este enemigo es tan inteligente y duro de roer, que sus autores ni se molestaron en crear animaciones o sonidos para su muerte,» 3D Juegos, 19 Jusio 2022. [En línea]. Available: https://www.3djuegospc.com/terror/este-enemigo-inteligente-duro-roer-que-sus-autores-se-molestaron-crear-animaciones-sonidos-para-su-muerte. [Último acceso: Marzo 2023].

[12] T. School, «La inteligencia artificial en videojuegos,» 28 Diciembre 2020. [En línea]. Available: https://www.tokioschool.com/noticias/inteligencia-artificial-videojuegos/. [Último acceso: Marzo 2023].

[13] JLPM, «Itch.io,» 3 Enero 2021. [En línea]. Available: https://jlpm.itch.io/neural-network-tower-defense. [Último acceso: Marzo 2023].

[14] M. González, «Cómo se aplica la Inteligencia Artificial en los videojuegos,» UAM, [En línea]. Available: https://www.iic.uam.es/noticias/como-aplica-inteligencia-artificial-en-videojuegos/. [Último acceso: Marzo 2023].

[15] A. Pilipiszyn, «Emergent tool use from multi-agent interaction,» OpenAI, 17 Septiembre 2019. [En línea]. Available: https://openai.com/research/emergent-tool-use. [Último acceso: Marzo 2023].

[16] «gamemaker,» 2017. [En línea]. Available: https://gamemaker.io/es. [Último acceso: Marzo 2023].

[17] T. School, «Blueprints Unreal: crea videojuegos de forma visual y fácil,» Tokio School, 1 Agosto 2020. [En línea]. Available: https://www.tokioschool.com/noticias/blueprints-unreal/#:~:text=Los%20Blueprints%20son%20assets%20dentro,conceptos%20disponibles%20para%20los%20programadores.. [Último acceso: Marzo 2023].

[18] «Godot Engine,» Godot, 2007. [En línea]. Available: https://godotengine.org. [Último acceso: Marzo 2023].

[19] A. C. González, «Profesional Review,» 20 Febrero 2020. [En línea]. Available: https://www.profesionalreview.com/2022/02/20/unity-3d-que-es-y-para-que-se-utiliza/. [Último acceso: Abril 2020].

[20] Unity, «Unity Documentation,» Unity, Marzo 2023. [En línea]. Available: https://docs.unity3d.com/Manual/ScriptingSection.html. [Último acceso: Abril 2023].

[21] Microsoft, «Microsoft,» Microsoft, 2023. [En línea]. Available: https://visualstudio.microsoft.com/es/. [Último acceso: Abril 2023].

[22] Unity, «Unity Machine Learning Agents,» Unity , [En línea]. Available: https://unity.com/products/machine-learning-agents. [Último acceso: Abril 2023].

[23] TensorFlow, «TensorFlow,» TensorFlow, 6 Enero 2022. [En línea]. Available: https://www.tensorflow.org/tensorboard/get_started?hl=es-419. [Último acceso: Abril 2023].

[24] Python, Pyton, 23 Marzo 2023. [En línea]. Available: https://docs.python.org/es/3/tutorial/venv.html. [Último acceso: Abril 2023].

[25] «PyTorch,» [En línea]. Available: https://pytorch.org. [Último acceso: Abril 2023].

[26] Python, «mlagents,» Python, 21 Noviembre 2022. [En línea]. Available: https://pypi.org/project/mlagents/. [Último acceso: Abril 2023].

[27] «ONNX,» [En línea]. Available: https://onnx.ai. [Último acceso: Abril 2023].

[28] «Vandal,» Vandal, [En línea]. Available: https://vandal.elespanol.com/sagas/metal-slug. [Último acceso: Abril 2023].

[29] D. Gameboy, «Jambo Jungle Free Sprites Assets Pack by didigameboy,» itch.oi, 26 Julio 2022. [En línea]. Available: https://didigameboy.itch.io/jambo-jungle-free-sprites-asset-pack. [Último acceso: Abril 2023].

[30] C. L. P. Moreno, «Topdoctors,» 5 Abril 2023. [En línea]. Available: https://www.topdoctors.es/diccionario-medico/adiccion-a-los-videojuegos#. [Último acceso: Mayo 2023].

ANEXO A: DOCUMENTO DE DISEÑO

Con el fin de tener la definición completa del videojuego se ha hecho un documento de diseño, que es una síntesis de lo que se va a hacer.

CONCEPTO

- **Estudio/Diseñadores:** Carlos Castillo González.
- **Género**: Acción/Plataformas 2D.
- **Plataforma:** Ordenador.
- **Categoría:** saga *Metal Slug*.
- **Mecánica:** El jugador podrá moverse hacia los lados, saltar y disparar. El jugador deberá usar estas acciones para eliminar los enemigos si fuera necesario, obtener los premios y llegar al final de la pantalla.
- **Público:** El juego va dirigido a todo tipo de persona, está pensado para que la IA adapte la dificultad en base a las habilidades, por lo que siendo mejor o peor podrá ser jugado.

MECÁNICA DEL JUEGO:

- **Cámara:** El jugador verá el juego en 2D.
- Describir el tipo de cámara que se utilizará. Es decir, qué perspectiva tiene el jugador ante lo que está viendo en el juego, si es 3D o 2D, vista isométrica, en primera persona, etc.
- Periféricos: Necesitara de un teclado y un ratón para jugarlo.
- **Controles**: Para moverse hacia los lados se utilizarán las teclas: A y D. “A” para moverse a la izquierda y “D” para moverse a la derecha. Para saltar utilizar el espacio (barra espaciadora) y para disparar el botón izquierdo del ratón.
- **Puntación**: La puntuación se define por las estrellas que se consiguen en el nivel, estas se obtienen de tres maneras: llegando al final del nivel, eliminando a todos los enemigos u obteniendo todos los premios.

INTERFACES: la única interfaz que tiene es la propia pantalla de juego.

PERSONAJES: Único personaje principal controlado por el jugador.

ENEMIGOS: Único tipo de enemigo que, en cuanto ve al jugador dispara y lo persigue, y hasta que deje de verle.

ARMAS: Única arma que tendrán tanto enemigos como personaje principal de disparo semiautomático.

ITEMS: Objeto que aumenta la vida en uno al jugador, premios, y la salida.

IMÁGENES DE CONCEPTO:

- **Personaje:**

Figura 9: Personaje Principal. Fuente: Elaboración Propia.

- **Enemigos:**

Figura 10: Enemigo. Fuente: Elaboración Propia.

- **Premios:**

Figura 11: Premio. Fuente: Elaboración Propia.

- **Vida:**

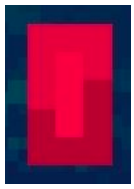

Figura 12: Vida. Fuente: Elaboración Propia.

- **Casa de inicio:**

Figura 13: Casa de inicio. Fuente: Elaboración Propia.

- **Puerta de salida:**

Figura 14: Puerta de salida. Fuente: Elaboración Propia.

ANEXO B: COMUNICACIONES IA/VIDEOJUEGO

En la siguiente figura (Figura 15) se ve el funcionamiento de la red neuronal.

Esta observará en todo momento la dificultad de la partida actual, y las estrellas que obtiene el jugador.

Cuando el estado del juego es 0 o 1, es decir el jugador ha ganado o perdido, la red neuronal tomará las acciones de aumentar, disminuir o mantener la dificultad.

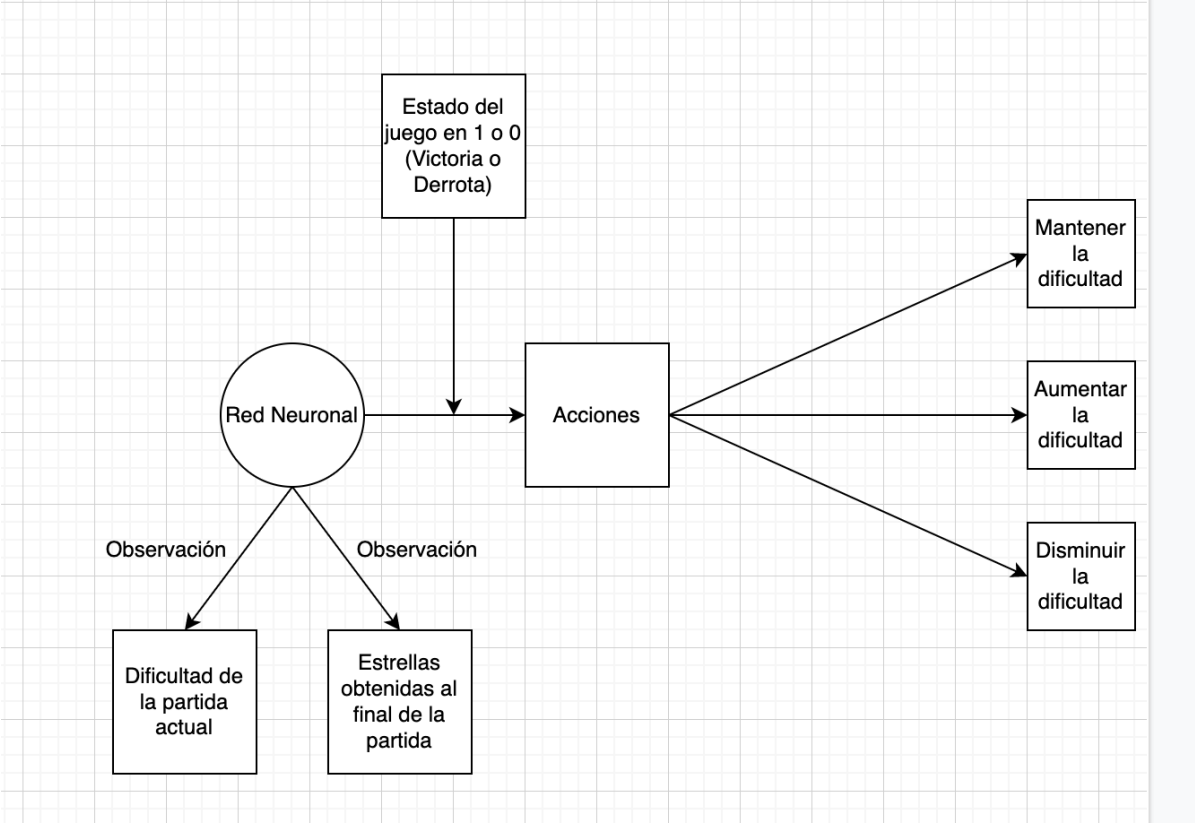

Figura 15: Funcionamiento de la red neuronal. Fuente: Elaboración Propia.

ANEXO C: MANUAL DE JUEGO

1. Resumen del juego:

El juego trata de tener que llegar al final como se pueda.

Dentro de la partida, podrás obtener premios y eliminar enemigos.

En el caso de que obtengas todos los premios (Figura 11) obtendrás una estrella, lo mismo si eliminas todos los enemigos (Figura 10) de la partida. Siempre que llegues al final (Figura 14) obtendrás una estrella y siempre que mueras, pierdas, perderás todas las estrellas.

Podrás aumentar tu vida en 1 durante la partida obteniendo dichos objetos (Figura 12).

Cuando la partida termine, es decir llegues al final o pierdas, la red neuronal decidirá a que nivel de dificultad tienes que jugar la siguiente partida.

2. Controles y mecánicas del juego:

El jugador se moverá hacia la izquierda y derecha con las teclas “A” y “D”. Podrá saltar con la barra espaciadora y disparará con el botón izquierdo del ratón.

Los enemigos seguirán y dispararán al jugador cuando le vean y hasta que dejen de verle.

Cada bala que impacte en el jugador por parte de los enemigos se le restará una vida, hasta que llegue a 0 que pierde la partida, si esta impacta en los enemigos por parte del jugador, se les restará una vida hasta que llegue a 0 y estos mueran.

Cuando el jugador choque con una vida o un premio este los recogerá, con las vidas se añadirán 1 por colisión y cuando choche con la puerta final este ganará la partida.

ANEXO D: CONTENIDO DEL CD

- Documento con la memoria, el mismo que se presenta en formato digital.
- El ejecutable del juego. Que se encuentra en PFG-GAME > Builds > PFG-Game.exe.