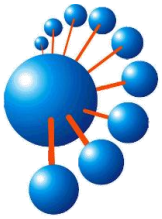
Universidad de Concepción

Facultad de Ingeniería

Departamento de Ingeniería Informática y Ciencias de la Computación



Evaluación y Comparación de Técnicas de Deep Reinforcement Learning en Juegos First Person Shooters

Miguel Ángel Villa Ríos

Memoria presentada para la obtención del título de Ingeniero Civil Informático

Profesor Guia: Julio Godoy del Campo

Septiembre 2025

Concepción, Chile

Índice

1. Introducción

Los videojuegos proporcionan un entorno ideal para la investigación en **inteligencia artificial** (**IA**) debido a los desafíos complejos, controlables y reproducibles que presentan. Estos entornos permiten entrenar agentes inteligentes que, una vez desarrollados, pueden aplicarse a problemas del mundo real, como la programación de vehículos autónomos o el control de drones. Un caso conocido es **GT Sophy** (Wurman et al., 2022), una inteligencia artificial entrenada en el simulador de conducción **Gran Turismo** (Polyphony Digital, 2022) mediante aprendizaje por refuerzo, que ha demostrado su capacidad al competir exitosamente contra vehículos virtuales en escenarios de alta competencia.

En el ámbito de los videojuegos estratégicos, como **Starcraft** (Vinyals et al., 2019), la **IA** enfrenta desafíos adicionales al requerir habilidades avanzadas de toma de decisiones, planificación estratégica y gestión de recursos. Estos juegos han impulsado el desarrollo de nuevos métodos en **IA**, evidenciando la capacidad de los agentes para manejar entornos complejos y dinámicos.

Por otro lado, los juegos de disparos en primera persona (**First Person Shooter, FPS**) destacan como buenas plataformas para evaluar y comparar enfoques de aprendizaje por refuerzo profundo (**Deep Reinforcement Learning, DRL**), que combina técnicas de aprendizaje por refuerzo (**Reinforcement Learning**, **RL**), donde un agente aprende a tomar decisiones a través de la interacción con su entorno, con redes neuronales profundas para manejar tareas complejas en espacios de alta dimensionalidad. En los juegos **FPS**, los agentes deben dominar habilidades como el razonamiento espacial, la toma de decisiones estratégicas en tiempo real y la adaptación a entornos cambiantes, lo que los convierte en un banco de pruebas exigente para algoritmos de **IA**.

Además, los videojuegos ofrecen una ventaja significativa en términos de disponibilidad de datos y velocidad de entrenamiento. En comparación con los entornos físicos, los entornos virtuales permiten generar grandes cantidades de datos de manera rápida y eficiente, reduciendo costos y riesgos asociados al entrenamiento en escenarios reales.

En esta memoria de título se estudiarán, aplicarán, evaluarán y compararán dos técnicas de **DRL** ampliamente utilizadas: los algoritmos **Deep Q-Network** (**DQN**) (Mnih et al., 2015) y **Proximal Policy Optimization** (**PPO**) (Schulman et al., 2017). Ambos métodos se aplicarán al juego **FPS DOOM**, conocido por la complejidad de sus entornos y la inteligencia de sus enemigos. Este videojuego constituye una plataforma adecuada para analizar el desempeño de los algoritmos de **DRL** en escenarios dinámicos y realistas, ofreciendo un marco para el desarrollo de modelos de IA para agentes autónomos en ambientes desafiantes.

2. Objetivos

2.1. Objetivo General

El objetivo general de esta memoria de título es estudiar, seleccionar, aplicar, evaluar y comparar distintas técnicas de aprendizaje por refuerzo profundos en un entorno simulado de juegos de disparo en primera persona.

2.2. Objetivos Específicos

1. Familiarizarse con el entorno de pruebas **ViZDoom** usado para el entrenamiento de técnicas de IA en el videojuego **DOOM**.
2. Seleccionar y aplicar técnicas de algoritmos de **DRL**, como **DQN** y **PPO** para entrenar agentes inteligentes.
3. Definición de métricas a evaluar que sean apropiadas al problema.
4. Realizar experimentos para comparar el rendimiento de las diferentes técnicas de **DRL** con respecto a las métricas definidas.
5. Analizar los resultados de los experimentos y extraer las conclusiones sobre la efectividad y las limitaciones de cada técnica de **DRL**.

3. Marco Teórico

3.1. Aprendizaje por Refuerzo

En el campo de la inteligencia artificial, dentro del aprendizaje automático, el aprendizaje por refuerzo es un enfoque en el cual un agente aprende a interactuar con un entorno mediante un proceso de prueba y error, guiado por un sistema de recompensas. En este contexto, el agente toma decisiones basadas en su estado actual dentro del entorno, con el objetivo de maximizar la recompensa acumulativa a largo plazo. Este aprendizaje se basa en una política, la cual determina las acciones que el agente debe realizar en cada estado.

Como se muestra en la **Figura 1**, en un problema de aprendizaje por refuerzo, el *Ambiente* entrega un *Estado* al *Agente*, quien luego ejecuta una *Acción*, tras lo cual el *Ambiente* devuelve una *Recompensa* correspondiente a dicha acción.

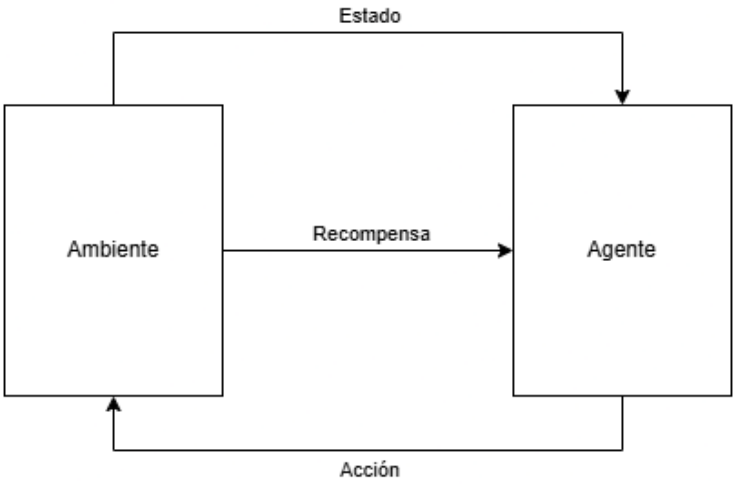


Figura 1: Diagrama Aprendizaje por Refuerzo. Fuente: Autoría Propia.

El aprendizaje por refuerzo se basa en la formulación matemática de los Procesos de Decisión de Markov (**MDP**, por sus siglas en inglés), que modelan entornos donde un agente toma decisiones secuenciales en función de recompensas y transiciones probabilísticas. Un MDP se define mediante el conjunto (S, A, P, R, γ), donde:

- **S** es el conjunto de estados posibles.
- **A** es el conjunto de acciones disponibles.
- **P (s' | s, a)** es la función de transición, que define la probabilidad de moverse de **s** a **s'** tras ejecutar la acción **a**.
- **R (s, a)** es la función de recompensa, que indica el valor obtenido tras tomar la acción **a** en el estado **s**.
- **γ** es el factor de descuento, el cual determina cuánto influyen las recompensas futuras en las decisiones actuales.

El proceso de aprendizaje se basa en la interacción entre el agente (el algoritmo de **RL**) y el entorno (el espacio de problemas con sus reglas y estados posibles). En cada paso, el agente toma una acción **a** en el estado actual **s**, lo que genera una transición a un nuevo estado **s'** y una recompensa **R (s, a)**. A través de prueba y error, el agente aprende qué acciones conducen a mayores recompensas acumuladas en el tiempo.

Una característica fundamental del aprendizaje por refuerzo es la búsqueda de un equilibrio entre la exploración (probar nuevas acciones para descubrir recompensas potenciales) y la explotación (usar el conocimiento actual para maximizar las recompensas). Este proceso es especialmente útil en problemas donde no existe un modelo explícito del entorno y las decisiones deben tomarse en tiempo real.

Los métodos tradicionales de **RL** son eficaces en problemas pequeños, pero se vuelven inviables cuando el espacio de estados y acciones es demasiado grande (como los pixeles de una imagen). En estos casos, almacenar y actualizar todas las combinaciones posibles se vuelve computacionalmente costoso, y los agentes tienen dificultades para generalizar su conocimiento a estados no explorados.

Para abordar estas limitaciones, se introducen técnicas más avanzadas que combinan **RL** con otras técnicas, dando origen al **Aprendizaje por Refuerzo Profundo**.

3.2. Aprendizaje por Refuerzo Profundo

Dado que los métodos tradicionales de aprendizaje por refuerzo no funcionan bien cuando el entorno es muy complejo o tiene demasiada información, el aprendizaje por refuerzo profundo surge como una solución. Este utiliza redes neuronales profundas, que son modelos inspirados en el cerebro humano, formados por muchas “neuronas” conectadas entre sí que procesan datos para reconocer patrones o hacer predicciones. Dentro de estas redes, se usan redes convolucionales, las cuales están diseñadas para trabajar con imágenes, que detecta automáticamente elementos importantes como formas, bordes o colores. Gracias a esto, el agente puede entender mejor la información visual del entorno y tomar decisiones sin tener que memorizar cada situación posible.

Por ejemplo, en **DOOM**, donde el agente ve imágenes del juego en tiempo real, el **DRL** usa estas redes convolucionales para identificar lo más importante de las imágenes, como los enemigos o los recursos que están dispersados por él escenario. Esto le permite aprender a actuar de forma inteligente para eliminar a estos enemigos u obtener los recursos necesarios para sobrevivir y cumplir con sus objetivos.

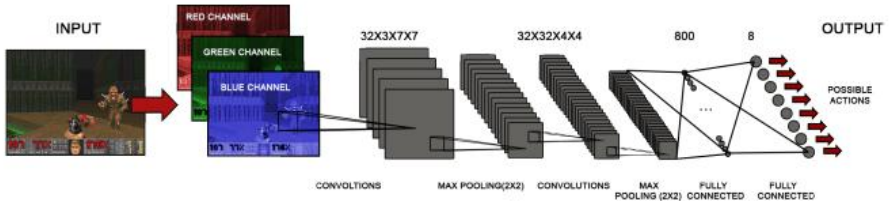


Figura 2: Arquitectura de la red neuronal convolucional utilizada en los experimentos de ViZDoom. Fuente: (Kempka et al., 2016)

3.3. Métodos Utilizados

En esta sección se presentarán los dos métodos utilizados, los cuales son dos técnicas muy utilizadas dentro del aprendizaje por refuerzo profundo: **DQN** y **PPO**. Ambas han demostrado ser efectivas en tareas complejas que requieren toma de decisiones en ambientes dinámicos como los videojuegos. **DQN** y **PPO** siguen diferentes estrategias para que el agente aprenda, lo que permite observar cómo cada enfoque influye en el desempeño del agente dentro del juego **DOOM**. A continuación, se explica cómo funciona cada uno de estos métodos.

3.3.1. Deep Q-Network

El método **DQN** (Mnih et al., 2015) combina el aprendizaje por refuerzo clásico basado en valores con redes neuronales profundas. Se basa en el algoritmo de Q-Learning, que aprende una función de valor **Q (s, a)**, la cual estima la recompensa acumulativa esperada al tomar una acción **a** en un estado **s** y seguir la mejor política posible a partir de ahí, siempre que a sea la acción óptima en ese estado.

En **DQN**, una red neuronal se entrena para estimar el valor **Q (s, a)**, que indica qué tan buena es una acción en un estado dado. Este entrenamiento utiliza una comparación entre las predicciones de la red y un valor de referencia, que incorpora tanto la recompensa inmediata como las futuras, obtenidas desde el siguiente estado. Este enfoque permite que el agente desarrolle una política óptima a largo plazo.

Para que el agente equilibre la **exploración** (probar nuevas acciones para descubrir recompensas potenciales) y la **explotación** (usar su conocimiento actual para maximizar las recompensas), **DQN** emplea la estrategia ***ε-greedy***. En este método:

- El agente selecciona acciones aleatorias con una probabilidad **ε** (exploración).
- Con una probabilidad de 1−**ε**, el agente elige la acción que maximiza su valor estimado **Q (s, a)** (explotación).

Inicialmente, **ε** se establece en un valor alto para fomentar la exploración, pero a medida que el entrenamiento avanza, **ε** disminuye gradualmente hasta un valor mínimo definido, favoreciendo la explotación. Este esquema adaptativo, conocido como **Decay Schedule**, asegura que el agente explore al inicio del entrenamiento y, posteriormente, utilice el conocimiento adquirido para mejorar su desempeño.

La estabilidad del entrenamiento, que se refiere a la capacidad del modelo de aprender de manera consistente sin divergencias o fluctuaciones significativas en su desempeño, es un desafío en **DQN**. Para abordarlo, se implementan dos técnicas clave:

- **Replay Buffer**: Almacena experiencias pasadas del agente (estado, acción, recompensa y siguiente estado) y permite entrenar la red neuronal con lotes de datos aleatorios. Esto reduce la correlación entre muestras consecutivas y mejora la eficiencia del aprendizaje.
- **Target Network**: Es una copia de la red principal que se actualiza periódicamente. Esto proporciona valores de referencia más estables, evitando oscilaciones bruscas durante el proceso de aprendizaje.

3.3.2. Proximal Policy Optimization

En el aprendizaje por refuerzo, los algoritmos suelen dividirse en métodos basados en valores y métodos basados en políticas. Los primeros, como **DQN**, aprenden una función de valor **Q (s, a)** para estimar la recompensa esperada de una acción en un estado dado. En contraste, los métodos basados en políticas optimizan directamente una política **π(a|s)**, que representa la probabilidad de tomar una acción específica dado un estado. Este enfoque es especialmente útil en entornos con espacios de acción grandes o continuos, donde los métodos basados en valores presentan limitaciones.

El algoritmo **PPO** (Schulman et al., 2017), pertenece a la familia de métodos basados en políticas, diseñado para ser simple, efectivo y estable durante el entrenamiento. Una de sus principales aportaciones es una nueva forma de ajustar la política llamada **clipping**, que evita que los cambios sean demasiado bruscos. En lugar de permitir que la política cambie sin control, **PPO** limita cuando puede cambiar la probabilidad de tomar una acción, lo que hace que el aprendizaje sea más suave y confiable.

Además, a diferencia de otros métodos basados en políticas, **PPO** permite reutilizar los datos varias veces entrenando la política durante múltiples pasadas sobre los mismos datos en pequeños lotes, aprovechando mejor cada experiencia del agente y aumentando la eficiencia. Para complementar este proceso, utiliza una medida llamada **ventaja o Advantage** (Schulman et al., 2015), que evalúa qué tan buena fue una acción comparada con lo esperado, proporcionando una señal más precisa para mejorar la política.

Esta combinación de características posiciona a **PPO** como un método robusto y versátil, capaz de adaptarse a diferentes entornos y problemas de aprendizaje por refuerzo. Mientras que **DQN** se destaca en escenarios con espacios de acción discretos y relativamente manejables, **PPO** sobresale cuando se enfrentan espacios de acción más complejos o continuos, donde la optimización directa de la política resulta más efectiva. Además, la estabilidad y eficiencia de **PPO** facilitan un aprendizaje más confiable y acelerado, en contraste con la sensibilidad de los métodos basados en valores a cambios abruptos o datos limitados.

3.4. DOOM y ViZDoom

DOOM (id Software, 1993) es un videojuego de disparos en primera persona desarrollado y publicado por id Software, considerado un hito en la historia del desarrollo de videojuegos por su influencia en el género de los **FPS**. El jugador asume el rol de un marine espacial que debe enfrentar hordas de enemigos, incluyendo humanos no muertos y criaturas demoníacas. El recorrido del juego inicia en las lunas de Marte y culmina en el infierno, donde el jugador debe avanzar por distintos niveles hasta alcanzar la salida o derrotar a jefes finales. A nivel técnico, **DOOM** fue uno de los primeros juegos en simular gráficos tridimensionales utilizando imágenes 2D tipo sprites, una técnica comúnmente referida como gráficos 2.5D.

El impacto cultural y comercial de **DOOM** fue significativo. Para 1999 había vendido más de 3,5 millones de copias, y se estima que alrededor de 20 millones de personas lo jugaron en los dos primeros años tras su lanzamiento. Se le reconoce ampliamente como el “padre” de los shooters en primera persona y como un punto de inflexión en la evolución de los videojuegos. Su éxito dio origen a numerosos imitadores y fomentó el surgimiento de comunidades activas de modding y speedrunning. (Wikipedia contributors, 2025)

Uno de los ports más avanzados de **DOOM** es **ZDoom** (“ZDoom”, s. f.), diseñado para Windows a partir del código fuente abierto del juego original. Esta versión no solo mantiene la fidelidad del juego base, sino que introduce múltiples mejoras en rendimiento, jugabilidad, y, especialmente, en la facilidad para modificar o implementar nuevos escenarios. Estas características convierten a **ZDoom** en una plataforma ideal para proyectos de investigación y desarrollo en inteligencia artificial.

En este contexto, se utilizó **ViZDoom** (Kempka et al., 2016), una biblioteca desarrollada en Python y basada en **ZDoom**, pensada específicamente para la investigación en **Machine Learning** y **Aprendizaje por Refuerzo Profundo**. **ViZDoom** permite configurar entornos de entrenamiento personalizados y ajustar parámetros clave, facilitando el desarrollo y evaluación de agentes inteligentes dentro del universo de **DOOM**.

Entre las funcionalidades más relevantes de **ViZDoom** para esta investigación, se destacan dos:

- **Dificultad del Escenario**: **ViZDoom** permite ajustar la dificultad en una escala del 1 al 5. Este valor determina el daño que los disparos de los enemigos causan al agente. Cada mapa tiene una dificultad predeterminada, y el objetivo es entrenar al agente para que se desempeñe de manera eficiente dentro de esta configuración.
- **Frame Skip**: Esta función permite omitir el renderizado de un número específico de fotogramas durante el entrenamiento, haciendo que el agente repita la acción realizada durante los frames omitidos. Esto reduce la carga computacional y ajusta el enfoque del agente en tareas críticas sin necesidad de procesar cada fotograma.

Por lo que basándonos en la literatura (Kempka et al., 2016), se decidió utilizar un **Frame Skip** de 4 para mapas donde la reacción rápida del agente es crucial, por ejemplo, al enfrentarse a enemigos, y un **Frame Skip** de 10 para mapas de exploración, donde la velocidad de reacción no es tan relevante.

3.5. Escenarios

Los escenarios utilizados en este trabajo fueron los entregados del entorno **ViZDoom**, el cual ofrece una amplia variedad de mapas predefinidos con distintas características y niveles de dificultad. Para organizar estos mapas de manera más clara, se clasificaron según dos criterios principales: el tipo de tarea que se tiene que realizar y la dificultad asignada por **ViZDoom**. Esto permite entender mejor los desafíos que cada escenario impone al agente que será entrenado.

La clasificación por tarea agrupa los escenarios en 3 tipos. El primer grupo corresponde a **escenarios de disparo**, donde el agente debe únicamente apuntar y eliminar a los enemigos, sin necesidad de que este se mueva. El segundo tipo incluye **escenarios de exploración o supervivencia**, en los que no se dispara, sino que el objetivo consiste en desplazarse por el entorno, ya sea para resistir durante un tiempo determinado o para encontrar un objetivo específico. Finalmente, el tercer grupo combina ambos elementos siendo los **escenarios combinados**: el agente debe moverse por el mapa mientras se enfrenta a enemigos, integrando así habilidades de navegación y combate. Dentro de cada una de estas categorías, los escenarios se ordenan según el nivel de dificultad definido por **ViZDoom**.

3.5.1. Escenarios de Disparo

3.5.1.1. Defend the Center

El escenario es un círculo donde el jugador se ubica en el centro y cinco enemigos que hacen daño cuerpo a cuerpo lo rodean. Los enemigos, que mueren de un solo disparo, se regeneran tras un tiempo. La munición es limitada, lo que obliga al agente a ser estratégico. El episodio termina al morir el agente, y este solo recibe recompensas por eliminar enemigos. La dificultad predeterminada es de 3.

Figura 3: Vista desde la perspectiva del jugador del Escenario Defend the Center. Fuente: Autoría Propia.

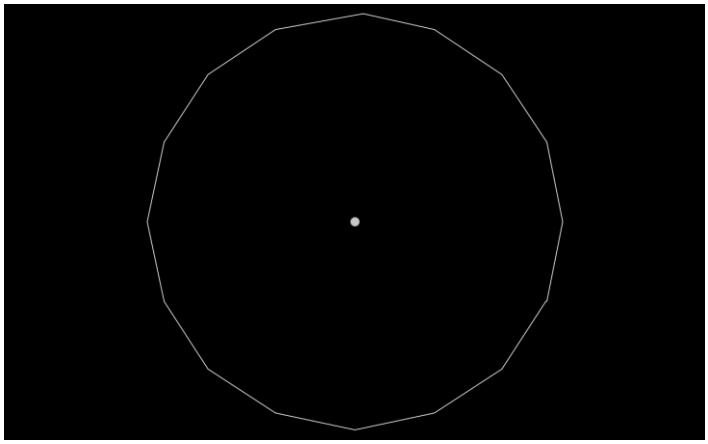


Figura 4: Vista desde arriba del Escenario Defend the Center. Donde el centro es donde aparece el jugador. Fuente: Captura generada por el entorno de ViZDoom.

3.5.1.2. Defend the Line

Este mapa rectangular genera al jugador en un extremo y a los enemigos en el otro. Inicialmente, aparecen tres enemigos cuerpo a cuerpo y tres que disparan, regenerándose tras ser eliminados. Con dificultad predeterminada es de 3, el objetivo es eliminar enemigos y sobrevivir el mayor tiempo posible, priorizando a los enemigos que disparan para prolongar la supervivencia.

Figura 5: Vista desde la perspectiva del jugador del Escenario Defend the Line. Fuente: Autoría Propia.

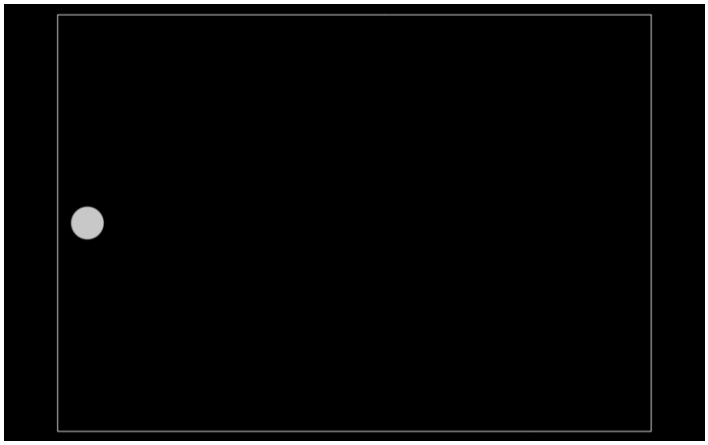


Figura 6: Vista desde arriba del Escenario Defend the Line. Donde el jugador es representado por el circulo blanco. Fuente: Captura generada por el entorno de ViZDoom.

3.5.1.3. Predict Position

El escenario es un rectángulo donde el jugador, ubicado en una pared, enfrenta a un enemigo que se desplaza de derecha a izquierda en la pared opuesta. El agente cuenta con un lanzamisiles y un solo proyectil. El objetivo es sincronizar el disparo con el movimiento del enemigo, considerando el retraso del misil. No tiene dificultad predeterminada asociada y el episodio finaliza cuando el misil impacta o se agota el tiempo.

Figura 7: Vista desde la perspectiva del jugador en el escenario Predict Position. Fuente: Autoría Propia.

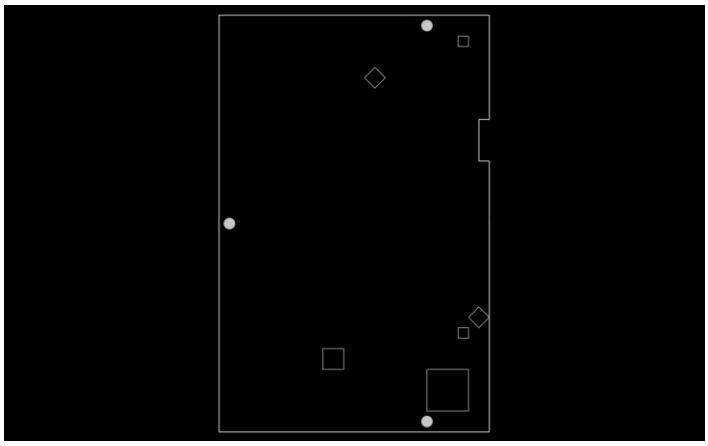


Figura 8: Vista desde arriba del Escenario Predict Position. El punto izquierdo es donde aparece el jugador y los otros puntos donde aparece el enemigo. Fuente: Captura generada por el entorno de ViZDoom.

3.5.2. Escenarios de Exploración y Supervivencia

3.5.2.1. Health Gathering

En este escenario rectangular, el suelo daña constantemente al agente. Al inicio hay botiquines repartidos por el mapa, y luego caen más del cielo. El agente obtiene recompensas por cada tic que permanece vivo y penalizaciones por morir. El objetivo es aprender a prolongar la supervivencia identificando acciones que mantienen la vida. Este mapa no tiene dificultad predeterminada asociada, ya que no hay enemigos.

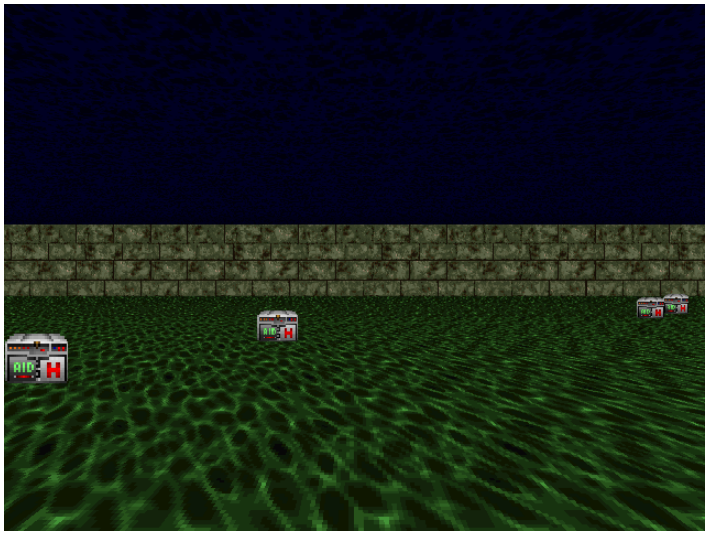


Figura 9: Vista desde la perspectiva del jugador del Escenario Health Gathering. Fuente: Autoría Propia.

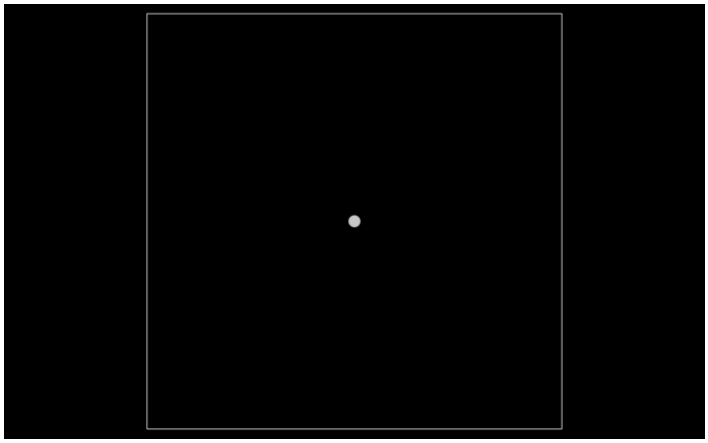


Figura 10: Vista desde arriba del Escenario Health Gathering. El centro es donde aparece el jugador. Fuente: Captura generada por el entorno de ViZDoom.

3.5.2.2. My Way Home

Este mapa consiste en habitaciones conectadas por pasillos. El jugador se genera en una ubicación aleatoria y debe encontrar una armadura ubicada en una sala específica. El escenario no tiene dificultad asociada y el objetivo es enseñar al agente a navegar por un entorno desconocido y descubrir su objetivo. El episodio termina cuando el agente alcanza la armadura o se agota el tiempo.

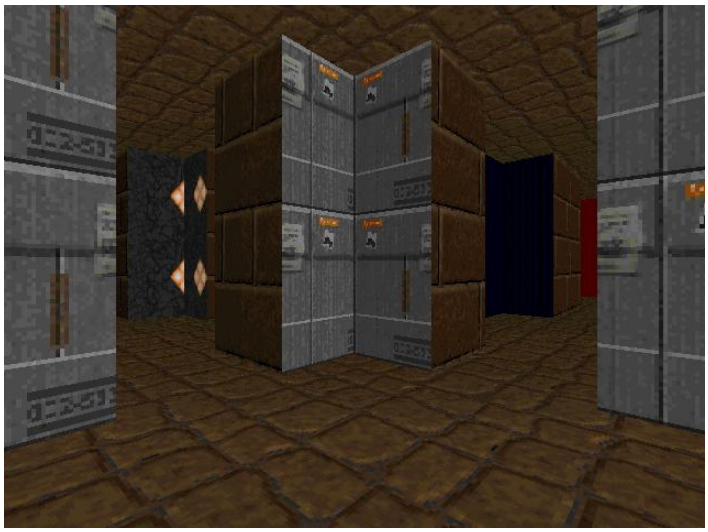


Figura 11: Vista desde la perspectiva del jugador del Escenario My Way Home. Fuente: Autoría Propia.

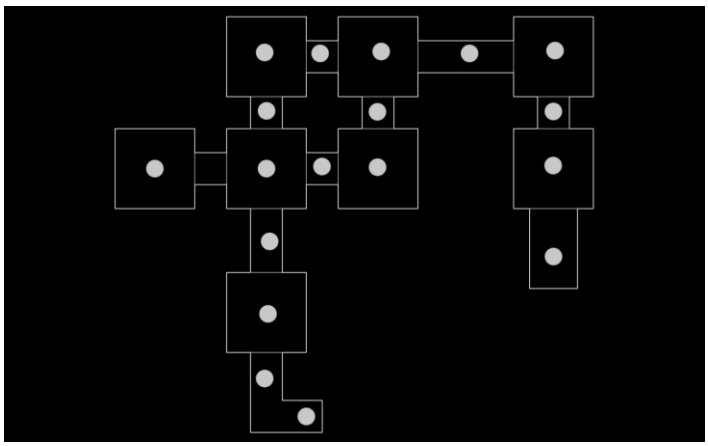


Figura 12: Vista desde arriba del Escenario My Way Home. Cada punto es donde puede aparecer el jugador. Fuente: Captura generada por el entorno de ViZDoom.

3.5.2.3. Take Cover

Este mapa rectangular coloca al jugador junto a una pared mientras los enemigos que disparan aparecen en la pared opuesta, generándose más con el tiempo. Con una dificultad predeterminada de 4, el objetivo es esquivar disparos para sobrevivir el mayor tiempo posible. El agente obtiene recompensas por cada tic que permanece vivo.

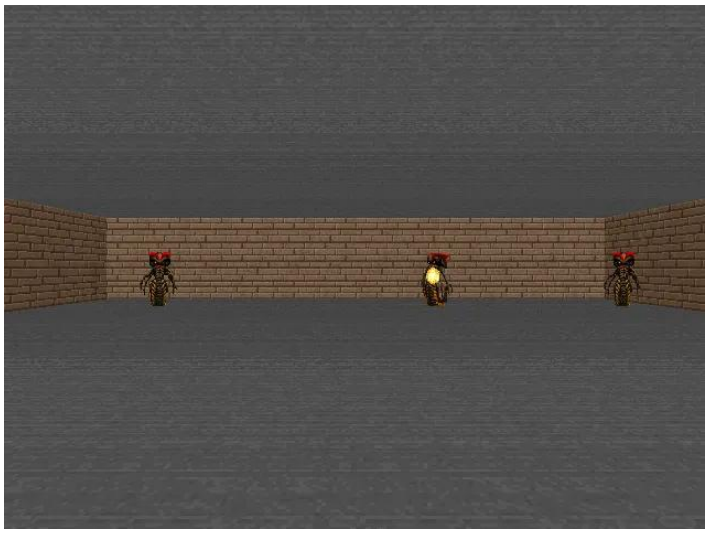


Figura 13: Vista desde la perspectiva del jugador en el escenario Take Cover. Fuente: Autoría Propia.

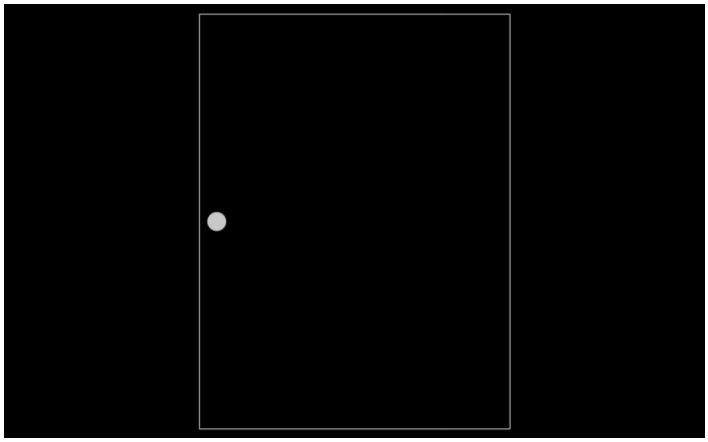


Figura 14: Vista desde arriba del Escenario Take Cover. El circulo es donde aparece el jugador. Fuente: Captura generada por el entorno de ViZDoom.

3.5.3. Escenarios Combinados

3.5.3.1. Deadly Corridor

Este mapa consiste en un pasillo estrecho con enemigos a ambos lados que disparan al jugador. Al final del pasillo se encuentra una armadura. La recompensa es proporcional al avance del agente hacia la armadura, con una bonificación al obtenerla. No se otorgan recompensas por eliminar enemigos, por lo que el objetivo es enseñar al agente a navegar hacia su meta mientras sobrevive. Con una dificultad predeterminada de 5, ignorar a los enemigos provoca una rápida eliminación.

Figura 15: Vista desde la perspectiva del jugador del Escenario Deadly Corridor. Fuente: Autoría Propia.

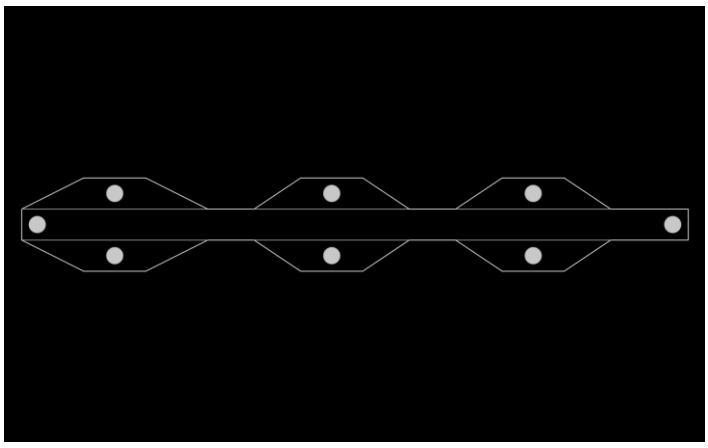


Figura 16: Vista desde arriba del Escenario Deadly Corridor. Al extremo izquierdo el jugador y al extremo derecho la armadura. Fuente: Captura generada por el entorno de ViZDoom.

3.5.3.2. *Deathmatch*

Este escenario tiene forma cuadrada con salas llenas de recursos como armas, municiones y botiquines. Enemigos aleatorios aparecen cada pocos segundos e intentan eliminar al agente. La recompensa varía según la dificultad del enemigo eliminado, siendo mayor para enemigos más fuertes. Con una dificultad predeterminada de 3, el objetivo es eliminar la mayor cantidad de enemigos antes de ser eliminado o agotar el tiempo.

Figura 17: Vista desde la perspectiva del jugador del Escenario Deathmatch. Fuente: Autoría Propia.

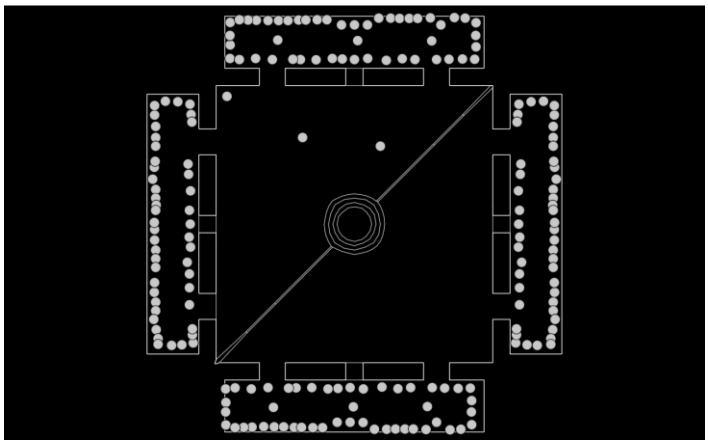


Figura 18: Vista desde arriba del Escenario Deathmatch. Los puntos son los objetos que se pueden recolectar. Fuente: Captura generada por el entorno de ViZDoom.

3.6. Parameter Tuning y Reward Shaping

En el desarrollo y optimización de algoritmos de aprendizaje por refuerzo profundo en entornos complejos como **ViZDoom**, es crucial implementar estrategias que mejoren la eficiencia del entrenamiento y la capacidad de generalización del agente. Entre estas estrategias, destacan el **Parameter Tuning** y el **Reward Shaping** (Ibrahim et al., 2024), las cuales fueron utilizadas para mejorar el desempeño de los algoritmos **DQN** y **PPO**.

El **Parameter Tuning** es un proceso usado en el aprendizaje por refuerzo, ya que el desempeño del agente depende en gran medida de la configuración de estos parámetros. En este estudio, se aplicó esto a los algoritmos **DQN** y **PPO** con el objetivo de mejorar su estabilidad y eficiencia durante el entrenamiento.

Para llevar a cabo esta optimización, se utilizó **Optuna**, una biblioteca especializada en la búsqueda automatizada de hiperparámetros. **Optuna** emplea técnicas avanzadas, como la **búsqueda bayesiana**, que utiliza el historial de resultados previos para predecir qué combinaciones de parámetros podrían ser más prometedoras, y la **selección adaptativa de pruebas**, que prioriza aquellas configuraciones que tienen mayor probabilidad de mejorar el rendimiento. Estas estrategias permiten explorar el espacio de parámetros de manera más eficiente que los métodos tradicionales de búsqueda en cuadrícula o aleatoria. Este enfoque facilitó la identificación de configuraciones óptimas que maximizaban el desempeño del agente sin necesidad de realizar pruebas manuales extensas, contribuyendo así a una mejora sustancial en la convergencia y estabilidad del entrenamiento.

Por otro lado, el **Reward Shaping** es una estrategia utilizada en el aprendizaje por refuerzo para modificar la función de recompensa con el fin de guiar al agente hacia los comportamientos deseables de manera más eficiente. En entornos complejos, como **ViZDoom**, donde el aprendizaje puede verse obstaculizado por una exploración ineficiente, el diseño adecuado de recompensas es fundamental para acelerar el proceso de entrenamiento y mejorar el rendimiento del agente.

El **Reward Shaping** implica la introducción de incentivos adicionales que refuerzan acciones beneficiosas y penalizan comportamientos no deseados. A través de esta técnica, se logra que el agente aprenda políticas más efectivas al proporcionar señales de recompensa más informativas que la recompensa original del entorno. Esto facilita la exploración eficiente del espacio de estados y permite una convergencia más rápida hacia estrategias optimas.

4. Desarrollo

4.1. Familiarización con el Entorno e Implementación Inicial de Algoritmos

El desarrollo del proyecto inició con una etapa de familiarización con el entorno **ViZDoom**, en la cual se llevó a cabo un análisis y categorización de los mapas disponibles, clasificándolos en tres tipos principales: mapas de disparo, enfocados en el combate directo; mapas de exploración o supervivencia, centrados en la navegación y recolección de recursos; y mapas combinados, que integraban elementos de ambas dinámicas. Durante esta fase, también se exploraron posibles extensiones al entorno, como la incorporación de nuevos mapas que facilitaran la implementación de técnicas como **Continual Learning** (Tomilin et al., 2023) y otros métodos de entrenamiento más, tales como **A3C** y redes neuronales recurrentes (**RNN**). Sin embargo, se decidió no avanzar con estas opciones, ya que agregar más técnicas y complejidad al entrenamiento podría haber dificultado tanto el proceso de entrenamiento como la comparación entre los distintos métodos, desviando así el foco del análisis. Por ello, se optó por mantener una estructura más simple, centrada en comparar dos métodos: uno más clásico (**DQN**) y otro más reciente (**PPO**). De esta manera, fue posible observar con mayor claridad cómo se comporta cada técnica en los distintos escenarios del entorno de juego.

Una vez comprendido el entorno, se procedió con la implementación inicial de algoritmos de aprendizaje por refuerzo, comenzando con Deep Q-Networks (**DQN**) utilizando la librería **PyTorch** y siguiendo la guía del artículo original. Los entrenamientos iniciales se realizaron en el mapa **Defend the Center**, replicando los parámetros reportados en la literatura. No obstante, los resultados obtenidos fueron poco satisfactorios. A pesar de aplicar ajustes, como la extensión del tiempo de entrenamiento y la modificación de la tasa de aprendizaje, las mejoras observadas fueron mínimas. Esto podría deberse, por un lado, a la falta de experiencia al implementar un algoritmo desde cero, ya que, incluso siguiendo el artículo original, detalles como la arquitectura de la red, el uso del replay buffer o la actualización de la red objetivo pueden afectar significativamente el rendimiento si no se configuran adecuadamente. Por otro lado, al tratarse de los primeros experimentos del proyecto, aún no se tenían bien definidos los hiperparámetros óptimos para el entorno, lo cual también pudo influir negativamente en los resultados.

Ante esta situación, se decidió probar la implementación de **DQN** proporcionada por la librería **SB3**, con el fin de verificar si el bajo rendimiento se debía a errores en la implementación propia o a otros factores externos. Esta prueba permitió obtener los primeros resultados positivos de manera más eficiente, lo que indicó que el problema estaba efectivamente en la implementación inicial. Debido a esto, y considerando los buenos resultados obtenidos con **SB3**, se optó por continuar utilizando esta implementación como solución final.

4.2. Ajustes de Parámetros y Cambios en las recompensas

4.2.1. Ajuste de Hiperparámetros

Para el ajuste de hiperparámetros, se empleó la biblioteca **Optuna**, utilizada para la optimización automatizada de parámetros. Este procedimiento implicó definir los hiperparámetros del algoritmo y realizar una serie de pruebas iterativas hasta identificar aquellos valores que maximizaban las recompensas obtenidas durante el entrenamiento. Los experimentos se llevaron a cabo en los mapas **Defend the Line** y **Defend the Center**, utilizando los algoritmos **DQN** y **PPO**. Aunque la metodología proporcionó información valiosa, las mejoras en el desempeño del agente no fueron significativas, lo que llevó a priorizar la exploración de otras estrategias.

Un factor limitante identificado durante el uso de **Optuna** fue su alta demanda de recursos computacionales y tiempo de procesamiento. Esta situación se agravó en escenarios que requerían periodos de entrenamiento extendidos, como aquellos enfocados en la exploración. Para mitigar este problema, se optó por implementar parámetros más simples con ajustes incrementales, lo que permitió mantener un equilibrio entre eficiencia y rendimiento.

Adicionalmente, se realizaron ajustes en la tasa de exploración del método **ε-greedy** del algoritmo **DQN**. Una limitación identificada en la versión original de **SB3** era la reducción lineal rápida del parámetro **ε**, que alcanzaba valores mínimos en los primeros pasos del entrenamiento, afectando negativamente el desempeño en escenarios que demandaban una exploración prolongada. En respuesta, se implementó una función modificable que permitía controlar el decrecimiento de **ε** de manera más adaptable, utilizando esquemas de reducción lineal lenta o exponencial según los requerimientos del mapa. Este cambio resultó en una exploración más equilibrada y contribuyó de manera significativa a mejorar las recompensas obtenidas por el agente.

4.2.2. Cambios en las Recompensas

Para mejorar el aprendizaje y el comportamiento del agente, se aplicaron estrategias de **Reward Shaping**, una técnica que consiste en modificar la función de recompensa del entorno para guiar de mejor manera al agente en el entrenamiento. En lugar de depender únicamente de las recompensas por defecto del entorno, se definen recompensas intermedias asociadas a acciones específicas que reflejan avances parciales hacia el objetivo general.

La elección de aplicar **Reward Shaping** surgió de observar que en los escenarios combinados (que combinan combate y exploración), las recompensas originales eran insuficientes para guiar eficazmente al agente. Por ejemplo, en el escenario **Deadly Corridor**, la recompensa por defecto se da al acercarse a la armadura, lo que generaba que el agente avanzara ignorando a los enemigos, lo que resultaba en su muerte.

De manera más general, se propuso una función de recompensa personalizada para cada escenario que consideraban recompensas positivas por darle a los enemigos con los disparos o recoger objetos. Como también penalizaciones por recibir daño o gastar munición. Esto se eligió porque permite dar retroalimentación más frecuente al agente, lo que acelera su aprendizaje y mejora la estabilidad durante el entrenamiento.

Escenario	**Recompensa Original**	**Recompensa Agregada**
Deadly Corridor	Recompensa en función de la distancia a la armadura	+100 por bala impactada -3 por daño recibido (Total: -300) -1 por bala gastada
Deathmatch	Recompensa según el enemigo eliminado	+0.1 por recoger objeto +0.1 por bala impactada +1 por enemigo eliminado

Tabla 1: Escenarios con Reward Shaping con su recompensa original y su recompensa agregada

4.3. Experimentos Por Escenario

Cada experimento fue realizado usando los métodos de **DQN** y **PPO**, en donde se usaron hiperparámetros distintos para cada método. Para realizar una comparación adecuada los agentes se entrenaron por la misma cantidad de **Time Steps** los cuales se cambiaban entre 300.000 y 1.000.000 según el escenario y las necesidades del entrenamiento.

Los escenarios fueron entrenados en orden de acuerdo con su tipo de mapa y dificultad, clasificándolos en **Escenarios de Disparo** (donde el objetivo es eliminar a los enemigos), **Exploración** (donde el objetivo es el moverse por el mapa) y **Supervivencia** (donde el objetivo es sobrevivir el mayor tiempo posible), siendo los **Escenarios Combinados** los que combinan todos los tipos anteriores. La idea de esta clasificación era el dejar los escenarios Combinados para el final ya que eran más complejos e iniciar con los escenarios que cubrían uno o dos de los aspectos para iniciar con los experimentos.

A continuación, se presentarán los experimentos realizados a los agentes en cada escenario y se explicarán las decisiones tomadas para entrenar a los agentes y como estas afectaron el entrenamiento, esto con sus debidos gráficos de recompensa por episodio.

4.3.1. Escenarios de Disparo

4.3.1.1. Defend the Center

El mapa **Defend the Center** fue utilizado como escenario base para la mayoría de las pruebas y ajustes iniciales del proyecto. Debido a su relativa simplicidad en comparación con otros mapas, sirvió como un entorno controlado para evaluar el desempeño de diferentes configuraciones y metodologías de entrenamiento.

En las primeras pruebas, se implementó una versión propia de **DQN** basada en el artículo original (Kempka et al., 2016), utilizando la librería **PyTorch** y manteniendo los mismos parámetros descritos en dicho estudio. Sin embargo, los resultados fueron deficientes, ya que el agente no lograba aprender estrategias de supervivencia efectivas dentro del entorno. Ante esta situación, se intentaron ajustes como extender el tiempo de entrenamiento y modificar el **Learning Rate**, pero estas modificaciones no generaron mejoras sustanciales en el rendimiento del agente. (**Figura 19 y 20**).

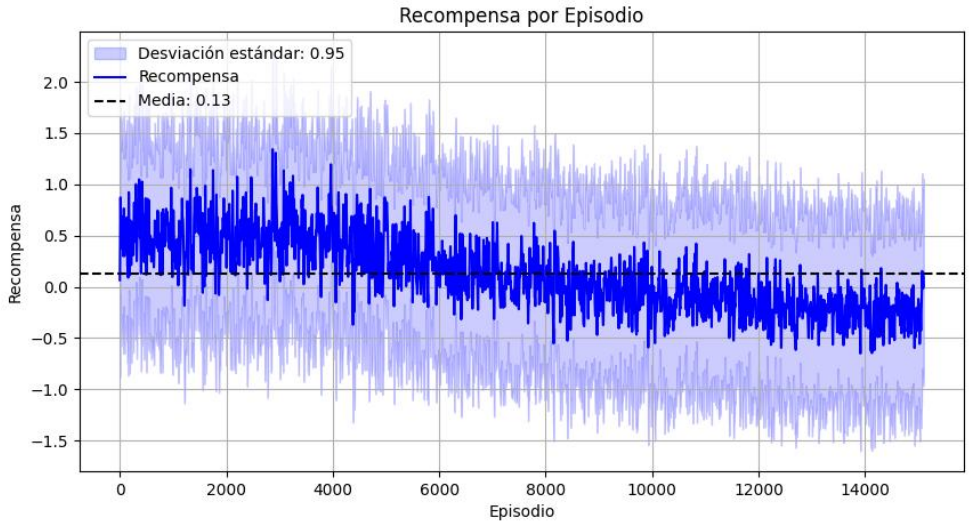


Figura 19: Gráfico Episodio Recompensa del entrenamiento en el escenario Defend the Center con DQN. Fuente: Autoría Propia.

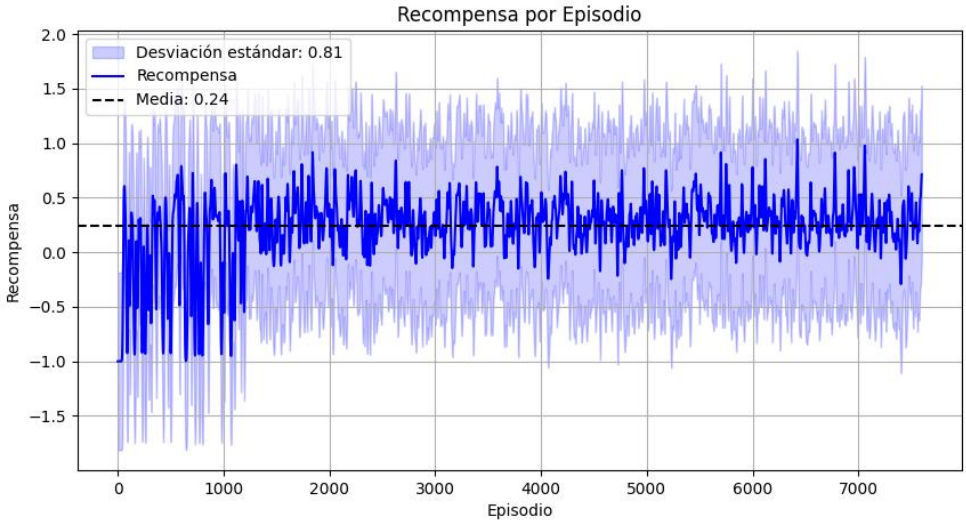


Figura 20: Gráfico Episodio Recompensa del entrenamiento en el escenario Defend the Center con DQN. Fuente: Autoría Propia.

Como se describió en la sección 4.1, los primeros entrenamientos en el mapa **Defend the Center** fueron realizados con una implementación de **DQN** en **PyTorch**, la cual no entregó resultados satisfactorios. Posteriormente, se optó por utilizar las versiones de **DQN** y **PPO** incluidas en la biblioteca **Stable-Baselines3** (**SB3**), con el objetivo de mejorar el rendimiento del agente. Sin embargo, en esta etapa inicial, los resultados también fueron limitados, ya que se utilizaban los hiperparámetros por defecto de **SB3**, sin ajustes específicos para el escenario, lo que afectó el desempeño del agente.

Para mejorar el desempeño, se implementaron los ajustes descritos previamente en la sección 4.2. A través de distintos experimentos, se lograron resultados considerablemente mejores, con puntuaciones promedio superiores a los 10 puntos en ambos algoritmos. Fue en este momento donde se aplicaron los cambios en la estrategia de exploración de **DQN**, lo que permitió una fase inicial de entrenamiento más efectiva. Además, se continuó ajustando otros hiperparámetros tanto en DQN como en PPO, con el fin de explorar combinaciones que pudieran contribuir a optimizar aún más el rendimiento del agente. (**Figura 21 y 22**)

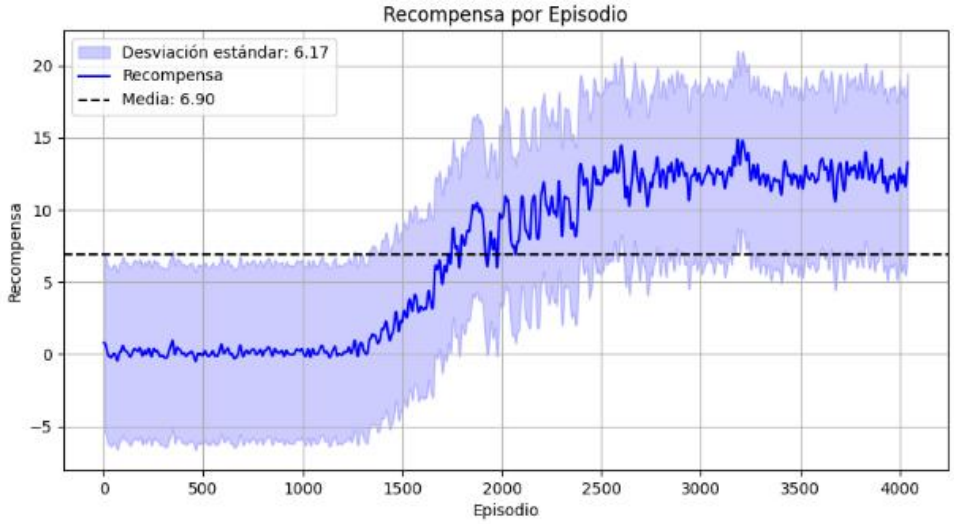


Figura 21: Gráfico Episodio Recompensa del entrenamiento con tuning en el escenario Defend the Center con DQN. Fuente: Autoría Propia.

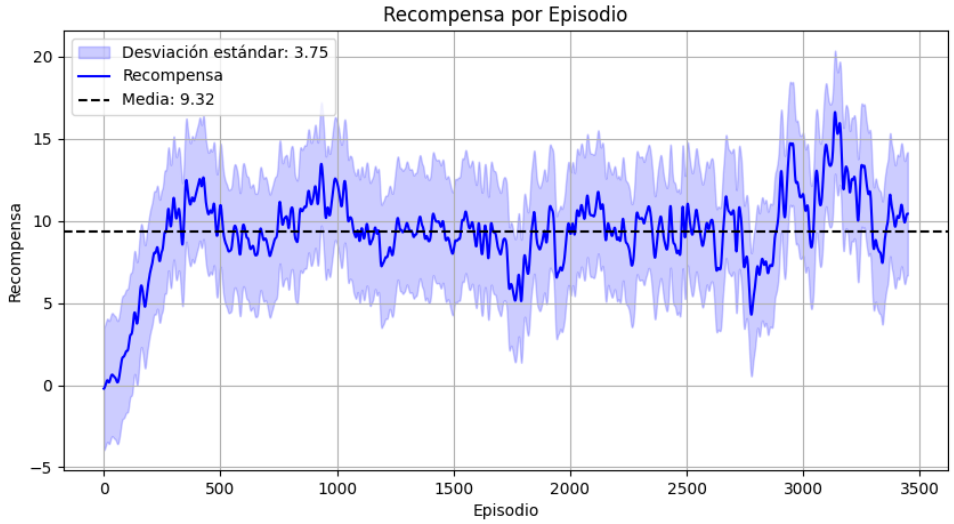


Figura 22: Gráfico Episodio Recompensa del entrenamiento con tuning en el escenario Defend the Center con PPO. Fuente: Autoría Propia.

Los cambios en los hiperparámetros de **DQN** no generaron mejoras significativas más allá del ajuste en **ε-greedy**, por lo que se mantuvo esta configuración sin alteraciones adicionales. (**Figura 23**). Por otro lado, en **PPO**, los ajustes implementados sí lograron una mejora sustancial en los resultados, pasando de una puntuación promedio de 10-15 puntos a aproximadamente 18 puntos. (**Figura 24**). Esto demostró que, con una adecuada optimización de parámetros, **PPO** podía alcanzar un desempeño superior en este escenario en comparación con **DQN**.

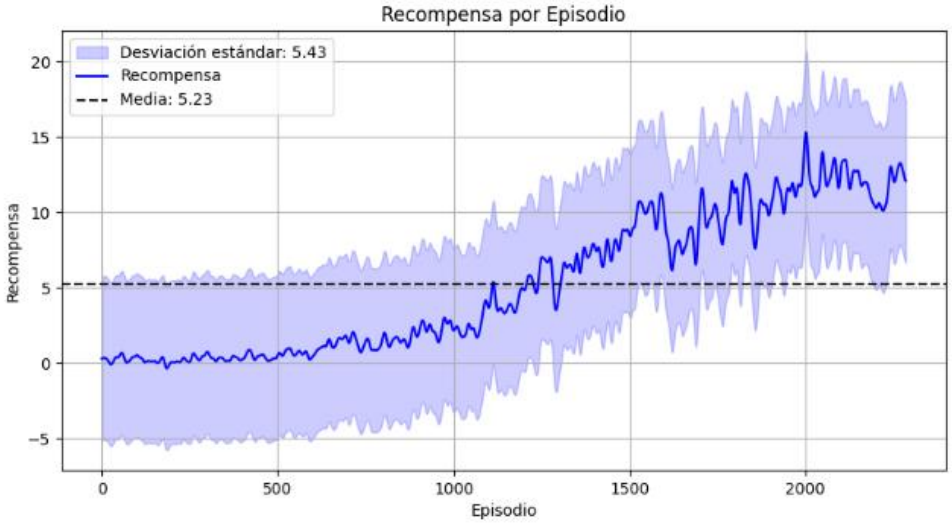


Figura 23: Gráfico Episodio Recompensa del entrenamiento final en el escenario Defend the Center con DQN. Fuente: Autoría Propia.

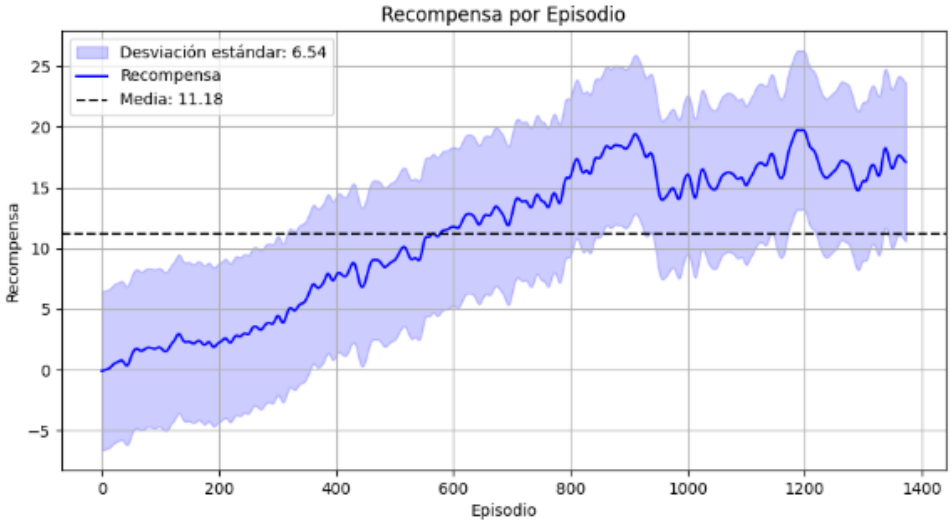


Figura 24: Gráfico Episodio Recompensa del entrenamiento final en el escenario Defend the Center con PPO. Fuente: Autoría Propia.

4.3.1.2. Defend the Line

El segundo escenario en el que se realizaron experimentos fue **Defend the Line**, el cual, debido a su similitud con **Defend the Center**, no requirió modificaciones significativas en los hiperparámetros. Dado que ambos mapas comparten mecánicas y estructuras similares, las configuraciones y estrategias que demostraron ser efectivas en el primer escenario fueron aplicadas directamente en este mapa.

Una vez que se completó el proceso de tuning de hiperparámetros en **Defend the Center** y se verificó que los resultados eran óptimos, se procedió a experimentar con **Defend the Line**. Se inició aplicando los hiperparámetros optimizados obtenidos en el mapa anterior y, tras ejecutar los entrenamientos iniciales, se observaron resultados sobresalientes, con puntuaciones promedio de 25 puntos. Durante las pruebas, se evidenció que el agente cumplía de manera eficiente sus objetivos dentro del entorno. (**Figura 25 y 26**).

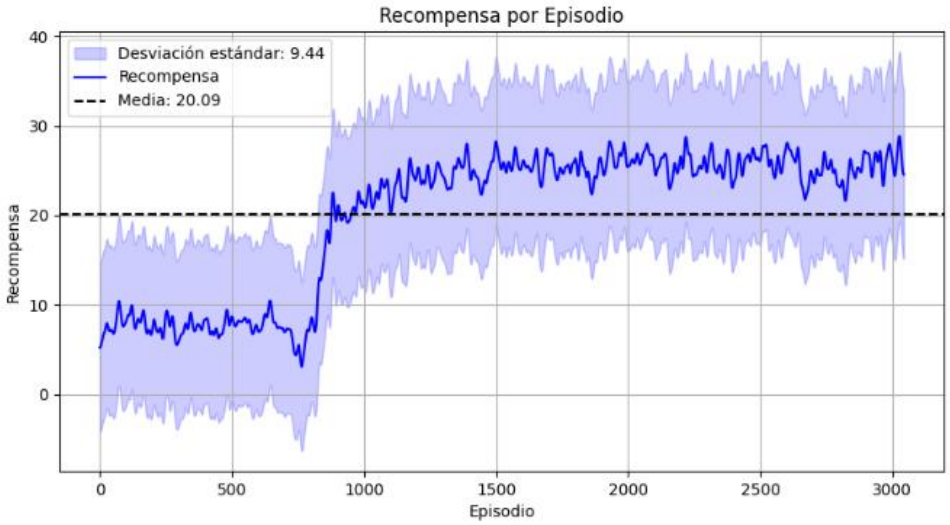


Figura 25: Gráfico Episodio Recompensa del entrenamiento con tuning en el escenario Defend the Line con DQN. Fuente: Autoría Propia.

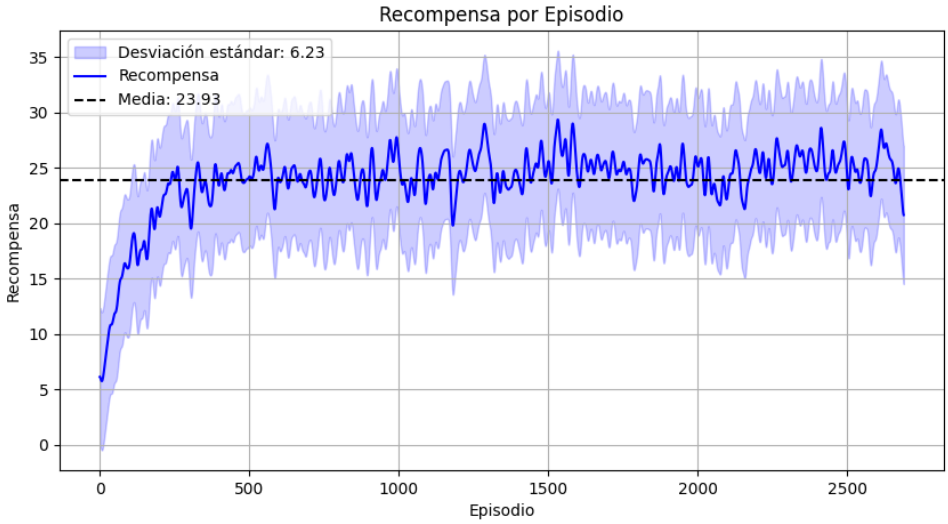


Figura 26: Gráfico Episodio Recompensa del entrenamiento con tuning en el escenario Defend the Line con PPO. Fuente: Autoría Propia.

Posteriormente, se implementaron ajustes adicionales en la función **ε-greedy**, aplicando los cambios previamente validados en **Defend the Center**. Además, se realizaron nuevas pruebas con diferentes combinaciones de hiperparámetros, evaluando si estos podían generar mejoras adicionales en el desempeño del agente. No obstante, los resultados obtenidos fueron muy similares a los alcanzados con la configuración anterior, por lo que se decidió mantener los hiperparámetros optimizados previamente como la configuración final para este escenario.

Dado que **Defend the Line** es un mapa relativamente simple y con dinámicas similares a las de **Defend the Center**, no se consideró necesario realizar entrenamientos adicionales o introducir nuevas estrategias. Una vez confirmada la efectividad de los parámetros seleccionados, se concluyó la experimentación en este escenario, permitiendo enfocar esfuerzos en mapas más complejos dentro del proyecto. (**Figura 27 y 28**)

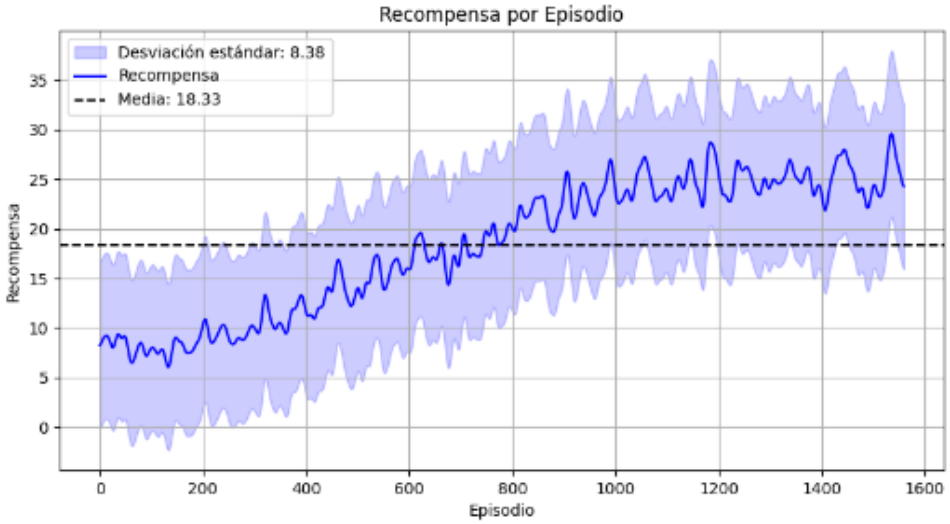


Figura 27: Gráfico Episodio Recompensa del entrenamiento final en el escenario Defend the Line con DQN. Fuente: Autoría Propia.

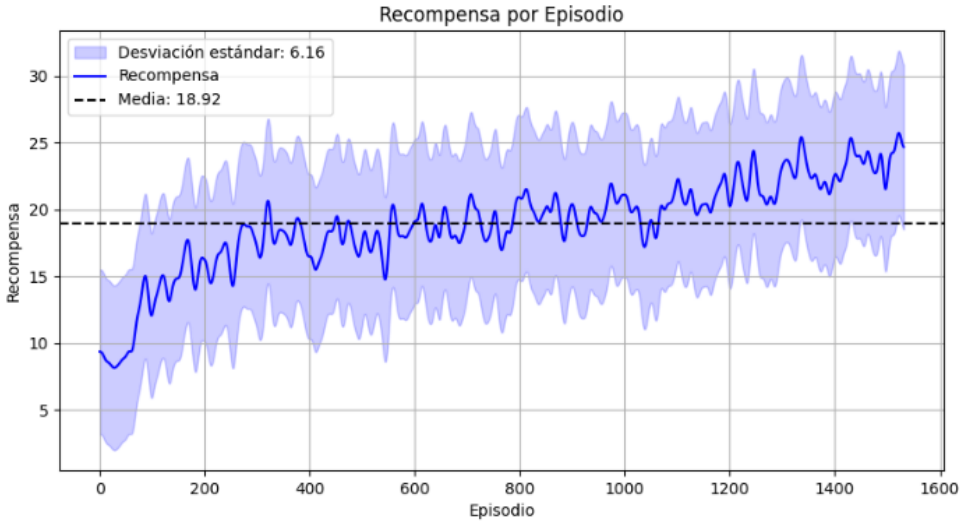


Figura 28: Gráfico Episodio Recompensa del entrenamiento final en el escenario Defend the Line con PPO. Fuente: Autoría Propia.

4.3.1.3. Predict Position

A diferencia de los mapas de disparo convencionales, en este escenario el jugador no usa una pistola con disparo instantáneo, sino un lanzamisiles, lo que introduce una nueva dificultad: el proyectil tarda en llegar al objetivo. Esto hace que el agente deba anticipar el movimiento del enemigo para acertar sus disparos, lo que complica significativamente el aprendizaje.

Para abordar este desafío, se realizaron experimentos iniciales utilizando hiperparámetros básicos sin modificaciones significativas. En el caso de **DQN**, se probaron distintos valores de **ε**, ajustando la velocidad de decaimiento para incentivar más exploración o más explotación. Sin embargo, el agente tuvo dificultades para aprender una estrategia efectiva. Aunque en algunos casos lograba seguir al enemigo con la mira, sus disparos seguían siendo aleatorios, lo que resultaba en una baja tasa de aciertos. (**Figuras 29, 30 y 31**).

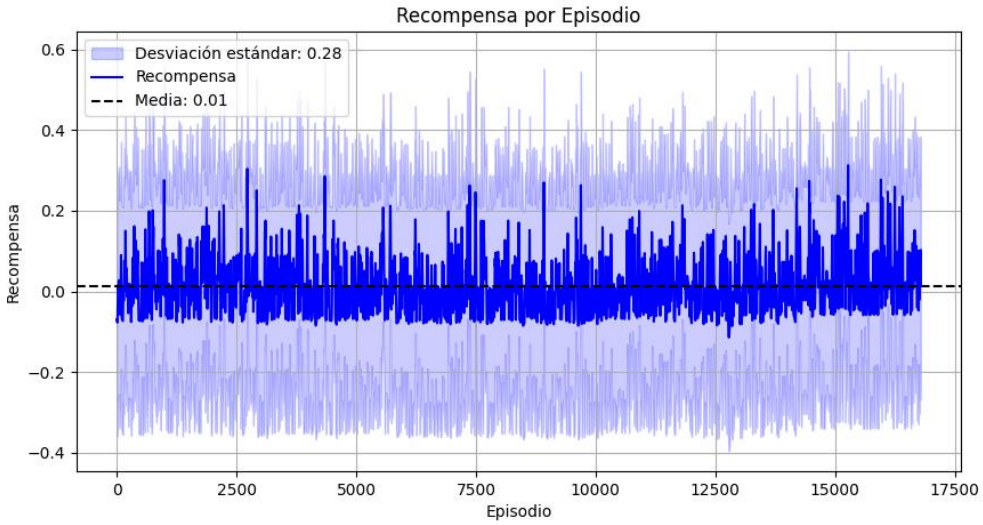


Figura 29: Gráfico Episodio Recompensa del entrenamiento n° 1 en el escenario Predict Position con DQN. Fuente: Autoría Propia.

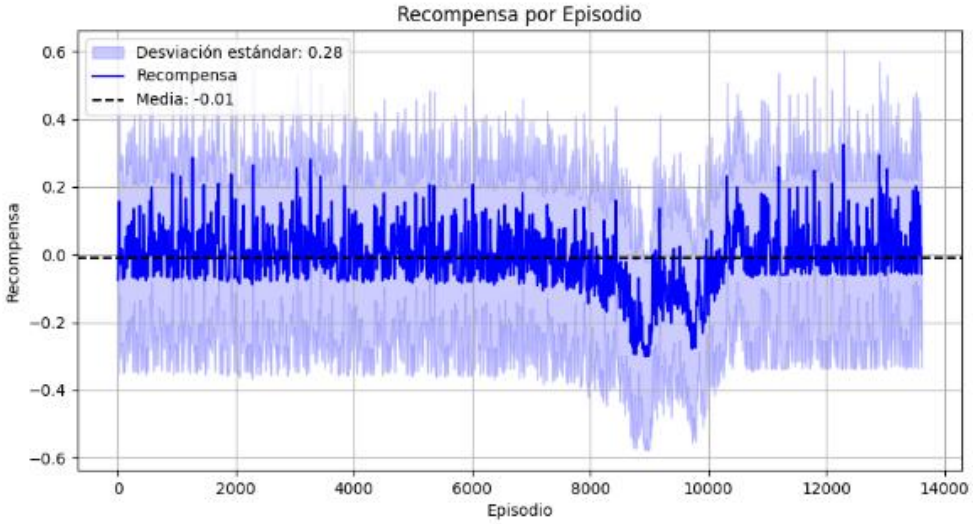


Figura 30: Gráfico Episodio Recompensa del entrenamiento n° 2 en el escenario Predict Position con DQN. Fuente: Autoría Propia.

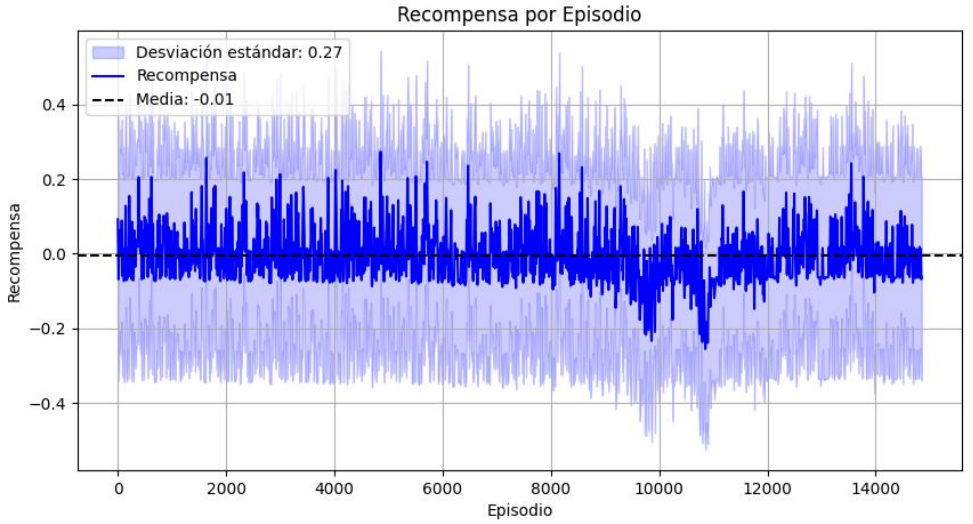


Figura 31: Gráfico Episodio Recompensa del entrenamiento n° 3 en el escenario Predict Position con DQN. Fuente: Autoría Propia.

En el caso de **PPO**, el entrenamiento fue más estable y con menos ajustes. En las primeras pruebas, **PPO** tampoco lograba aprender, mostrando un comportamiento pasivo en el que el agente esperaba sin disparar. No fue hasta que se aumentó la tasa de aprendizaje que **PPO** empezó a mejorar, logrando finalmente predecir el movimiento del enemigo y acertar disparos con mayor precisión. (**Figura 32 y 33**)

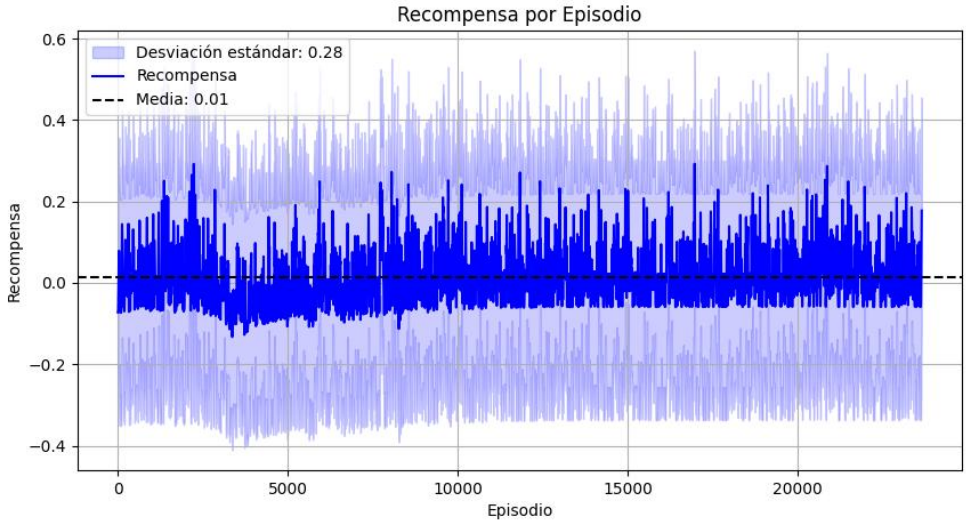


Figura 32: Gráfico Episodio Recompensa del entrenamiento en el escenario Predict Position con PPO. Fuente: Autoría Propia.

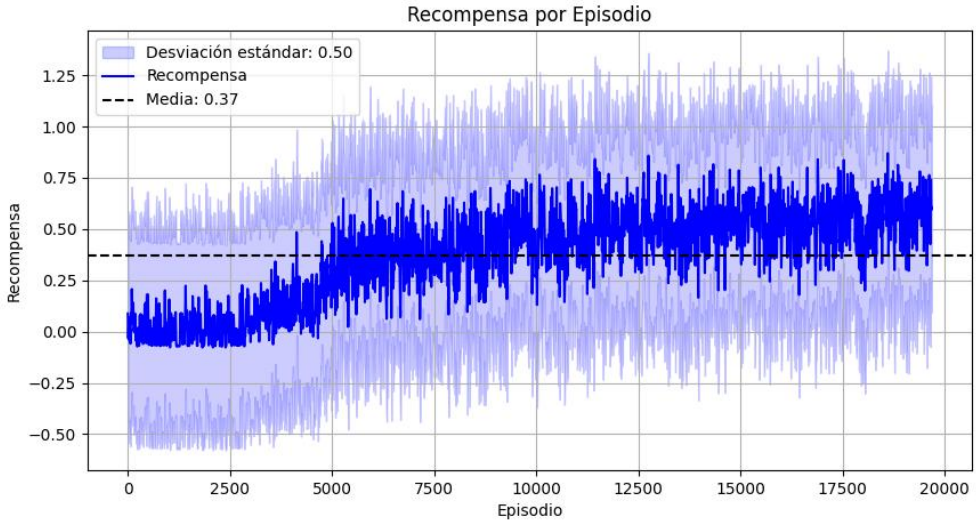


Figura 33: Gráfico Episodio Recompensa del entrenamiento final en el escenario Predict Position con PPO. Fuente: Autoría Propia.

Estos experimentos demostraron que **PPO** se adaptó mejor a este escenario que **DQN**, probablemente debido a su capacidad para manejar problemas de optimización más complejos. Sin embargo, aún existe margen de mejora en la configuración de **DQN** para este tipo de entornos.

4.3.2. Escenarios de Exploración y Supervivencia

4.3.2.1. Health Gathering

Este escenario presenta una dinámica completamente diferente a los mapas anteriores, ya que no involucra combate, sino que se centra en la exploración y supervivencia. Debido a esta diferencia fundamental, fue necesario abordar su entrenamiento con estrategias distintas a las utilizadas en los otros entornos.

El primer paso en la experimentación con este mapa fue la aplicación de tuning de hiperparámetros y el entrenamiento del agente con los valores obtenidos. Sin embargo, los resultados iniciales fueron poco favorables. Durante las pruebas, se observó que el agente mostraba un comportamiento ineficaz: en muchas ocasiones permanecía estático en el punto de inicio sin explorar el entorno, o simplemente giraba la cámara de manera continua, sin desplazarse ni recolectar los elementos de salud necesarios para sobrevivir. Esto derivó en un bajo desempeño del agente, ya que no lograba cumplir el objetivo del mapa. (**Figura 34 y 35**).

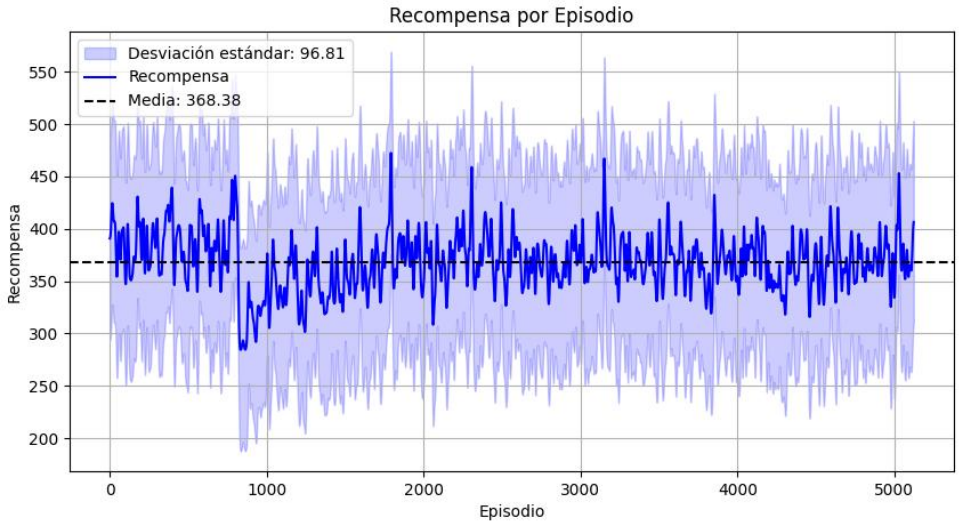


Figura 34: Gráfico Episodio Recompensa del entrenamiento en el escenario Health Gathering con DQN. Fuente: Autoría Propia.

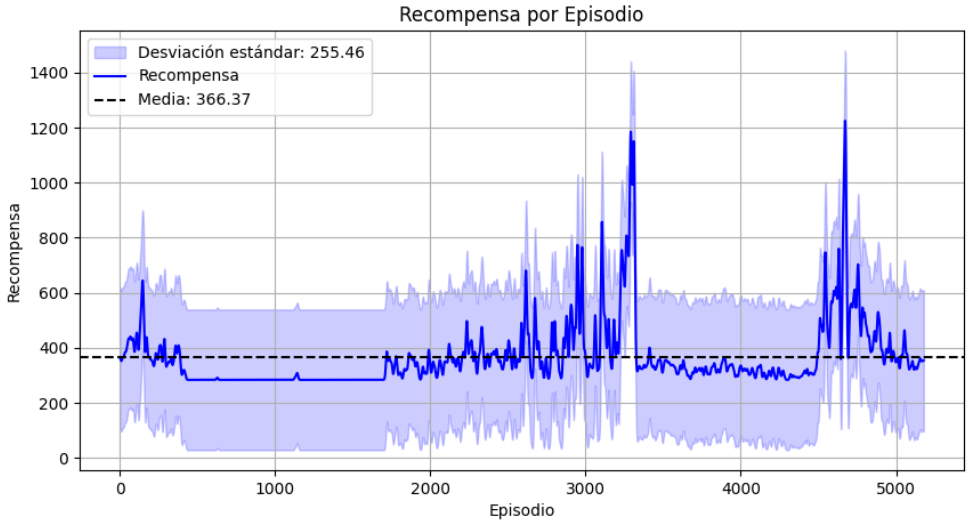


Figura 35: Gráfico Episodio Recompensa del entrenamiento en el escenario Health Gathering con PPO. Fuente: Autoría Propia.

Uno de los factores que pudo haber contribuido a estos resultados deficientes fue la decisión de no modificar el **Frame Skip**, que en esta primera fase se mantuvo en 4. Estudios previos sobre exploración en entornos similares sugerían que un **Frame Skip** más alto, en el rango de 10 o más, podía mejorar el desempeño en escenarios donde la navegación y el movimiento continuo son fundamentales. A pesar de esta observación, no se continuó con el tuning de hiperparámetros debido al alto consumo de recursos computacionales y la falta de mejoras significativas en los entrenamientos iniciales.

Tras la implementación de ajustes en **ε-greedy** y modificaciones adicionales en los hiperparámetros, se realizaron nuevos entrenamientos, esta vez aumentando el **Frame Skip** a 10. Estos cambios dieron resultados considerablemente mejores, permitiendo que el agente desarrollara una estrategia más eficiente para navegar por el entorno y recolectar salud.

En esta fase de experimentación, se evidenció una diferencia notable en el rendimiento de los algoritmos. En los gráficos de entrenamiento, **PPO** mostró una curva de aprendizaje mucho más estable y eficiente en comparación con **DQN**. De hecho, **DQN** no logró obtener buenos resultados finales, mostrando dificultades para aprender un comportamiento consistente en este mapa. Sin embargo, dado que **Stable-Baselines3** (**SB3**) guarda los pesos del mejor episodio, se logró identificar una instancia en la que el agente entrenado con **DQN** desarrolló una estrategia competitiva, alcanzando puntajes finales que rivalizaban con los obtenidos por **PPO**. (**Figura 36 y 37**).

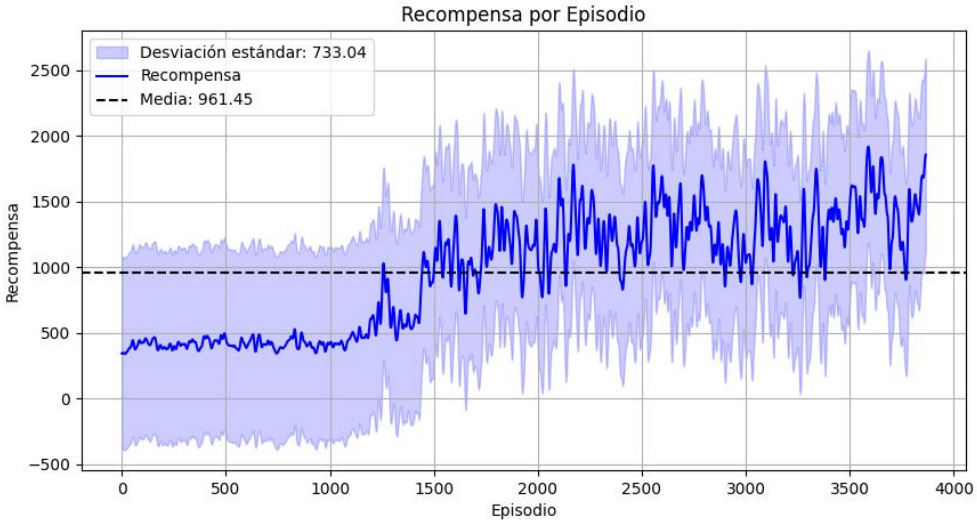


Figura 36: Gráfico Episodio Recompensa del entrenamiento final en el escenario Health Gathering con PPO. Fuente: Autoría Propia.

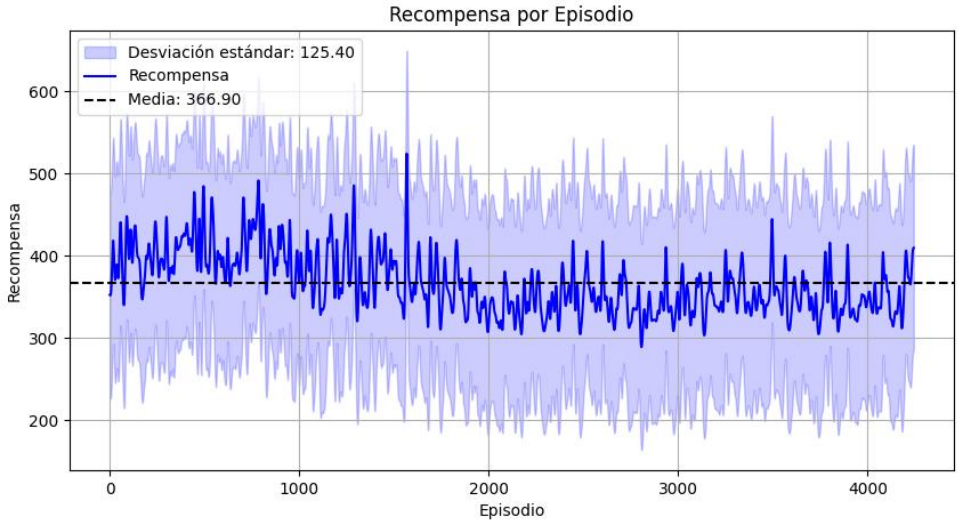


Figura 37: Gráfico Episodio Recompensa del entrenamiento final en el escenario Health Gathering con DQN. Fuente: Autoría Propia.

Con estos ajustes, el agente logró cumplir satisfactoriamente el objetivo del escenario, manteniéndose en el mapa y recolectando salud de manera eficiente. Aunque la estrategia final no representaba la solución óptima, sí se logró alcanzar un máximo local, permitiendo un desempeño adecuado en este entorno sin combate.

4.3.2.2. My Way Home

Este escenario se basa en la exploración dentro de un laberinto, presentando desafíos similares a **Health Gathering**, ya que el objetivo principal no es el combate, sino la navegación eficiente para alcanzar un punto específico dentro del mapa. Debido a esta naturaleza, el enfoque inicial de entrenamiento siguió el mismo esquema que en **Health Gathering**: se aplicó directamente ajuste de hiperparámetros antes de realizar los entrenamientos completos.

Los resultados iniciales no fueron favorables. Se observaron comportamientos ineficaces en el agente, como permanecer girando en el punto de inicio sin moverse significativamente o avanzar en línea recta hacia una pared sin intentar corregir su trayectoria. Estos comportamientos indicaban una falta de aprendizaje en la navegación del laberinto. Este problema también se reflejó en los gráficos de entrenamiento, donde no se observó una tendencia de mejora ni señales de que el agente estuviera aprendiendo a desplazarse correctamente. (**Figura 38 y 39**).

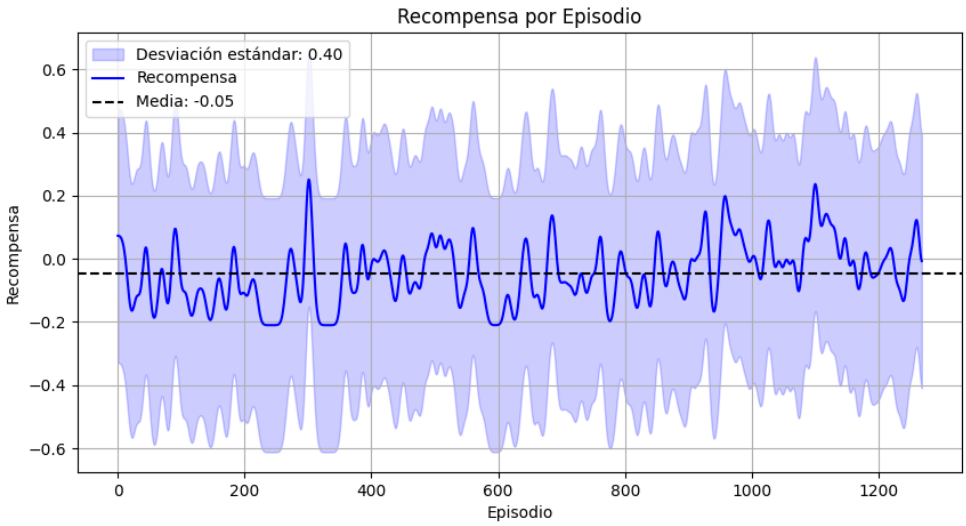


Figura 38: Gráfico Episodio Recompensa del entrenamiento en el escenario My Way Home con DQN. Fuente: Autoría Propia.

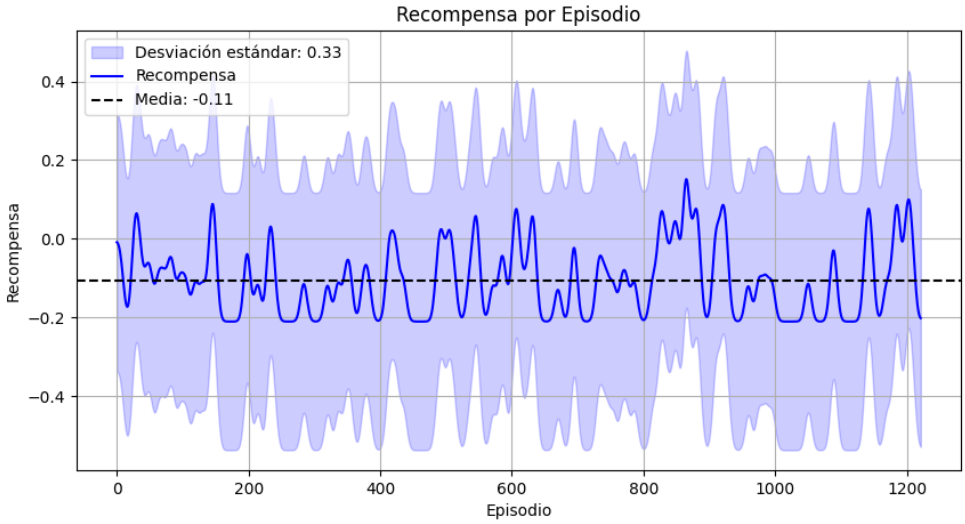


Figura 39: Gráfico Episodio Recompensa del entrenamiento en el escenario My Way Home con PPO. Fuente: Autoría Propia.

Ante estos resultados, se implementaron varias modificaciones en los entrenamientos posteriores. Se realizaron ajustes en el manejo de **ε** para **DQN**, además de modificar el **Frame Skip** para el escenario, aumentando su valor para facilitar el aprendizaje del movimiento en el laberinto. Adicionalmente, los hiperparámetros obtenidos en el tuning inicial fueron reemplazados por configuraciones más simples, permitiendo realizar cambios manuales progresivos de manera más eficiente.

Estos ajustes tuvieron un impacto positivo en **PPO**, logrando que el agente aprendiera a navegar el laberinto, recogiera la armadura y completara el objetivo del escenario. Sin embargo, **DQN** no obtuvo el mismo éxito, ya que nunca logró alcanzar la armadura dentro del laberinto, lo que sugiere que el algoritmo tenía más dificultades para aprender un patrón de navegación eficiente. (**Figura 40 y 41**).

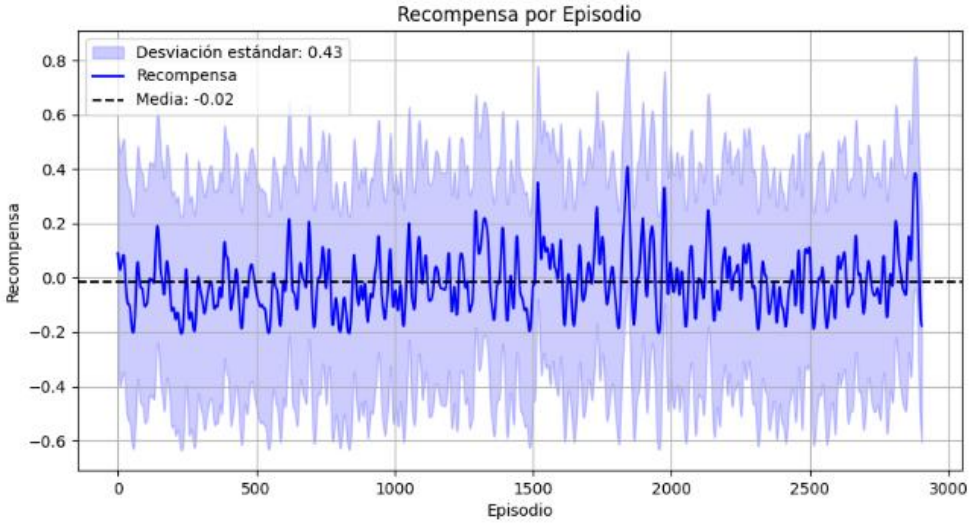


Figura 40: Gráfico Episodio Recompensa del entrenamiento final en el escenario My Way Home con DQN. Fuente: Autoría Propia.

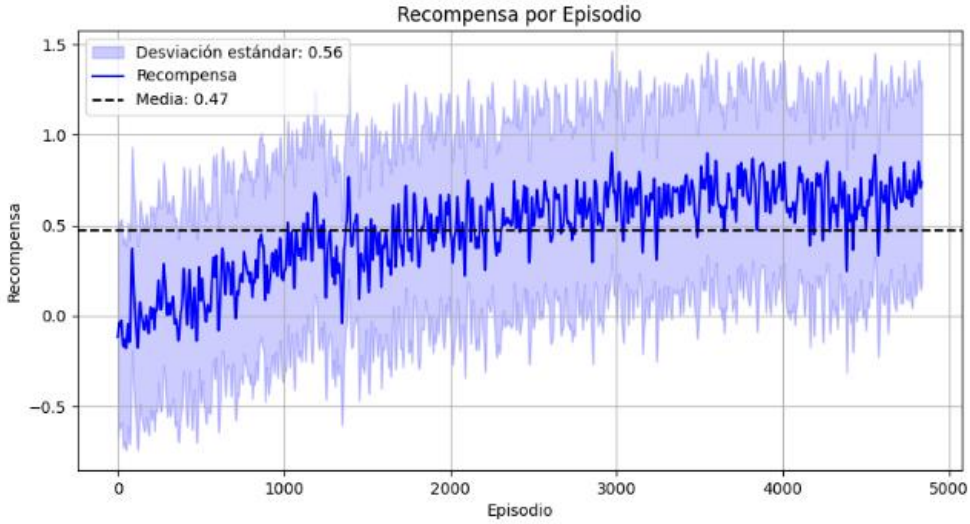


Figura 41: Gráfico Episodio Recompensa del entrenamiento final en el escenario My Way Home con PPO. Fuente: Autoría Propia.

Para mejorar el rendimiento de **DQN**, se llevaron a cabo experimentos adicionales en los que se aumentó el tiempo de entrenamiento y el buffer size, con el objetivo de mejorar la memoria del agente y permitirle recordar mejor la estructura del laberinto. No obstante, estos cambios no generaron mejoras significativas, y el agente continuó sin alcanzar el objetivo del mapa. (**Figura 42 y 43**).

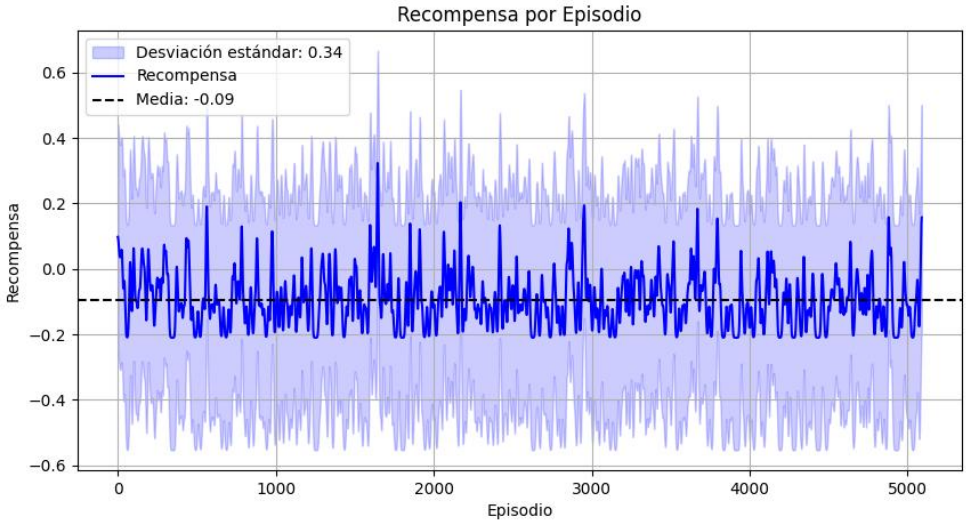


Figura 42: Gráfico Episodio Recompensa del primer entrenamiento extra en el escenario My Way Home con DQN. Fuente: Autoría Propia.

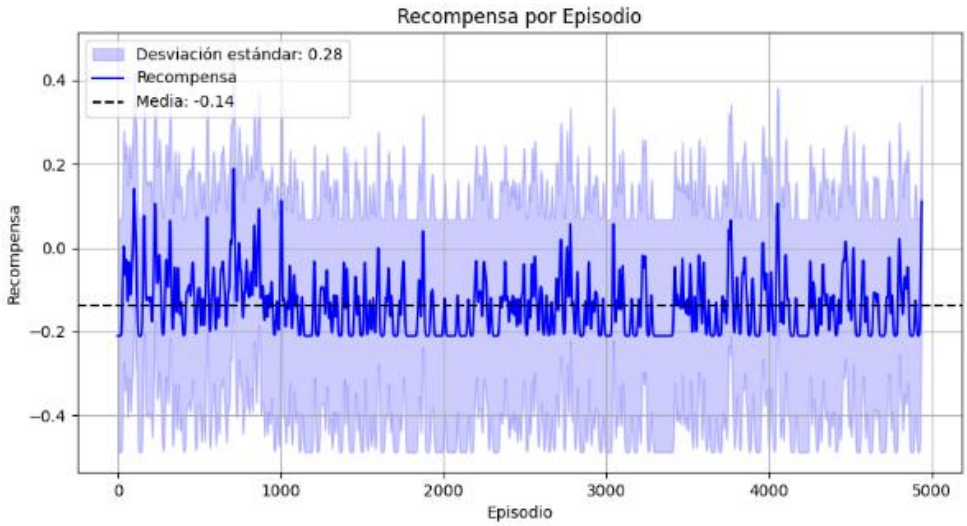


Figura 43: Gráfico Episodio Recompensa del segundo entrenamiento extra en el escenario My Way Home con DQN. Fuente: Autoría Propia.

Finalmente, aunque **DQN** mostró dificultades en este entorno, **PPO** logró aprender una estrategia efectiva, lo que sugiere que este método es más adecuado para escenarios que requieren exploración estructurada. A pesar de los resultados obtenidos, aún existe margen de mejora en la optimización del entrenamiento para **DQN** en este tipo de mapas.

4.3.2.3. Take Cover

Este mapa formó parte de la última tanda de experimentos, junto con **Predict Position**. Al igual que en el escenario anterior, se realizaron pruebas con distintas configuraciones de hiperparámetros para evaluar el rendimiento de **DQN** y **PPO**.

Para **DQN**, el enfoque principal fue ajustar el **Learning Rate** y la tasa de decaimiento de **ε** con el objetivo de encontrar una combinación que permitiera un buen aprendizaje. En las primeras pruebas, los resultados fueron bastante prometedores, logrando que el agente aprendiera a cubrirse y disparar de manera efectiva. (**Figura 44**). Sin embargo, cuando se realizaron más experimentos con otros valores de hiperparámetros, el rendimiento del agente disminuyó y su comportamiento se volvió menos eficiente. (**Figura 45**).

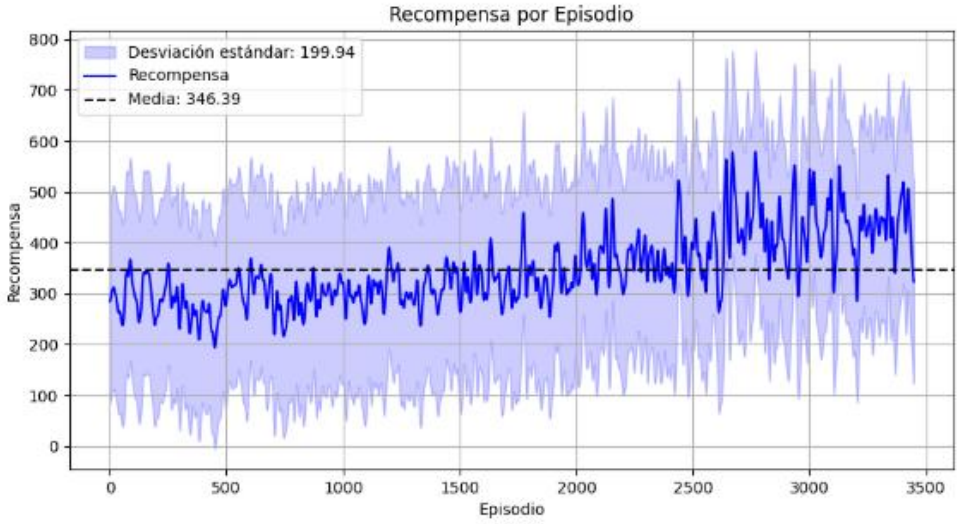


Figura 44: Gráfico Episodio Recompensa del entrenamiento en el escenario Take Cover con DQN. Fuente: Autoría Propia.

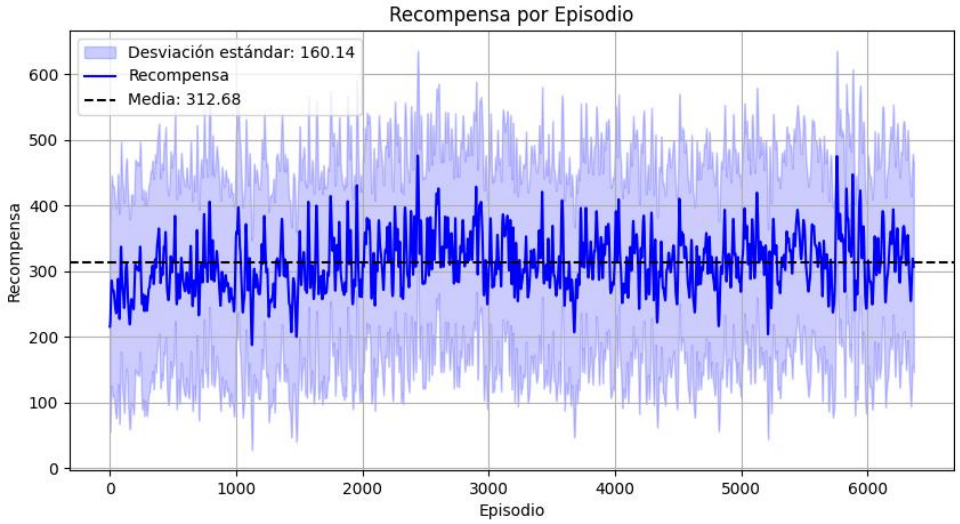


Figura 45: Gráfico Episodio Recompensa del entrenamiento final en el escenario Take Cover con DQN. Fuente: Autoría Propia.

En el caso de **PPO**, el enfoque fue más simple, ya que no se hicieron cambios significativos más allá del **Learning Rate**. Desde los primeros experimentos, los resultados fueron positivos, logrando un comportamiento eficiente sin necesidad de ajustes adicionales. (**Figura 46 y 47**).

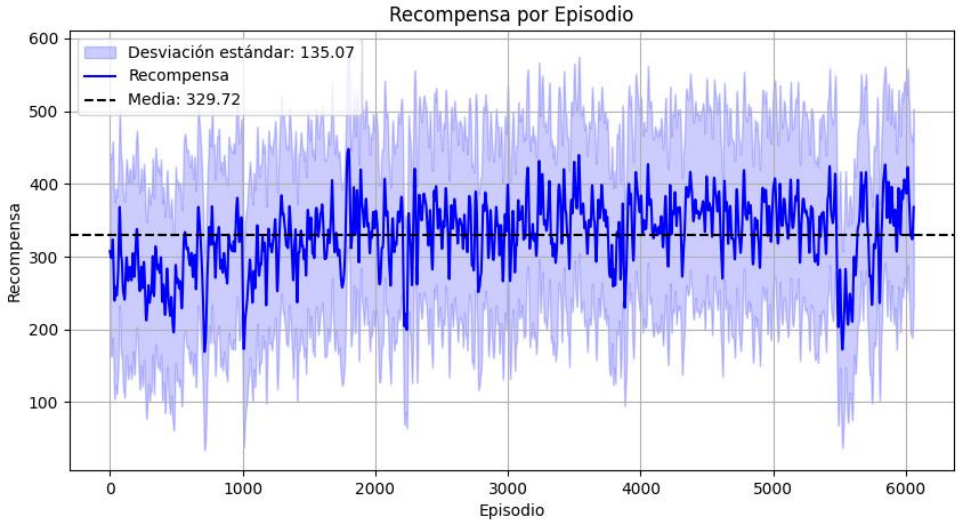


Figura 46: Gráfico Episodio Recompensa del entrenamiento en el escenario Take Cover con PPO. Fuente: Autoría Propia.

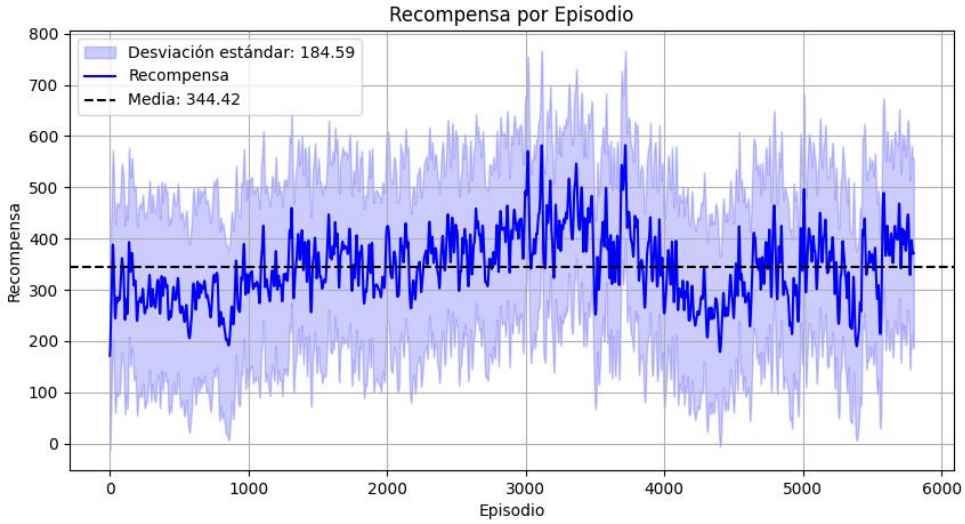


Figura 47: Gráfico Episodio Recompensa del entrenamiento final en el escenario Take Cover con PPO. Fuente: Autoría Propia.

Estos resultados sugieren que **DQN** es más sensible a cambios en los hiperparámetros, mientras que **PPO** logra un desempeño estable con menos ajustes. Debido a esto, **PPO** resultó ser la opción más efectiva para este escenario, aunque los primeros experimentos de **DQN** también lograron buenos resultados.

4.3.3. Escenarios Combinados

4.3.3.1. Deadly Corridor

Se llevaron a cabo una serie de experimentos en el escenario **Deadly Corridor** con el objetivo de diseñar un agente efectivo para navegar y cumplir con los objetivos del entorno. Los primeros experimentos se realizaron luego de aplicar un ajuste de parámetros iniciales, con el fin de comenzar con una base más controlada. Sin embargo, los resultados no fueron los esperados, ya que el agente mostraba comportamientos erráticos o inactivos. Debido a esto, se optó por utilizar hiperparámetros más conservadores, lo que permitió establecer una base más estable para continuar los entrenamientos. (**Figura 48 y 49**).

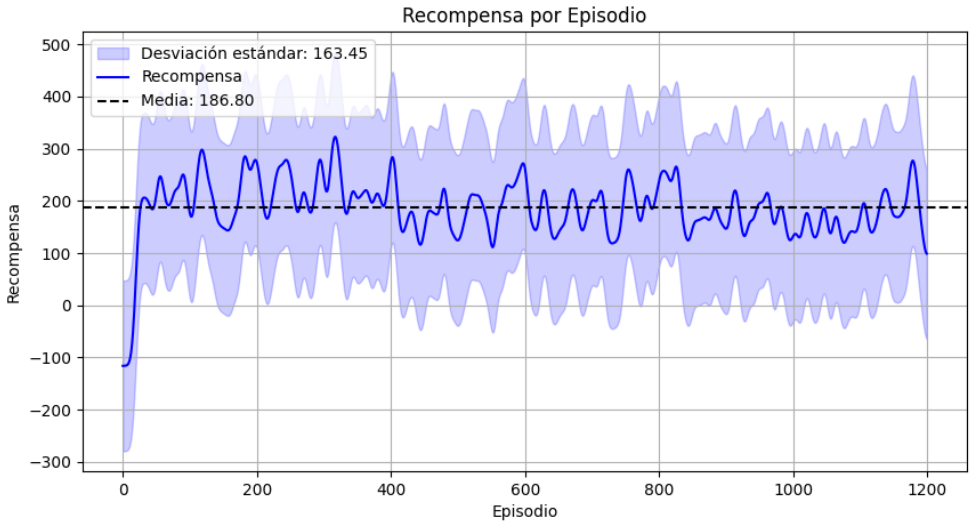


Figura 48: Gráfico Episodio Recompensa del entrenamiento con ajuste de parámetros en el escenario Deadly Corridor con DQN. Fuente: Autoría Propia.

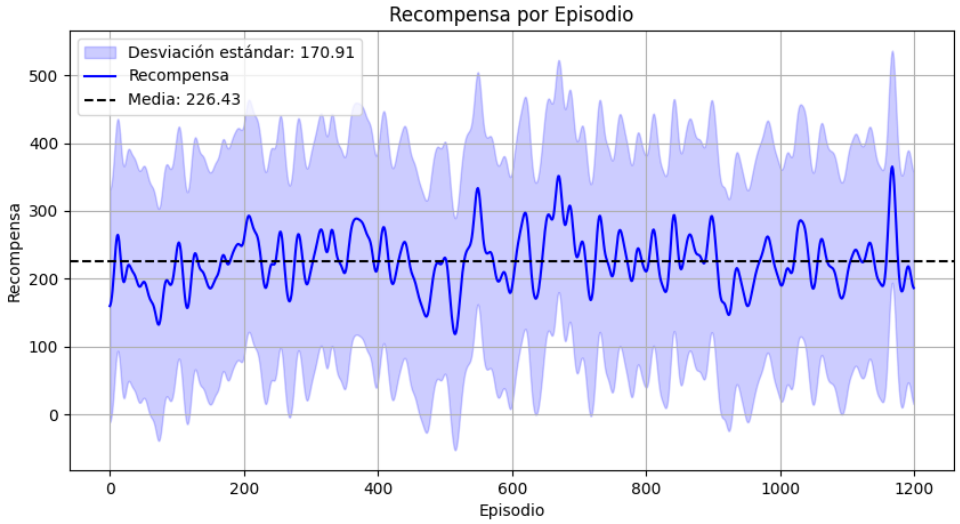


Figura 49: Gráfico Episodio Recompensa del entrenamiento con ajuste de parámetros en el escenario Deadly Corridor con PPO. Fuente: Autoría Propia.

Posteriormente, se implementó la técnica de **Curriculum Learning** (Narvekar et al., 2020), una estrategia de entrenamiento en la que el agente se enfrenta progresivamente a tareas de mayor dificultad, comenzando con niveles más simples y avanzando hacia los más complejos. En este caso, se utilizó un único agente que fue entrenado secuencialmente desde el nivel de dificultad 1 hasta el 5. Esta progresión tuvo como objetivo facilitar el aprendizaje de comportamientos útiles que pudieran luego generalizarse a escenarios más desafiantes.

Esto permitió al agente comenzar a desplazarse por el entorno y acercarse a la armadura, uno de los objetivos principales del mapa. Sin embargo, en estos niveles más sencillos, el agente tendía a avanzar en línea recta recibiendo gran cantidad de daño, ya que no aprendía a eliminar enemigos ni a esquivarlos (**Figura 50 y 52**). Este comportamiento resultó ineficaz en dificultades más altas, donde la supervivencia depende de una estrategia de combate más compleja. Esto se refleja en las **Figuras 51 y 53**, donde no se observa una mejora progresiva en el aprendizaje: en las dificultades 2 y 3 las recompensas se mantienen constantes, mientras que en las dificultades 4 y 5 estas se vuelven negativas.

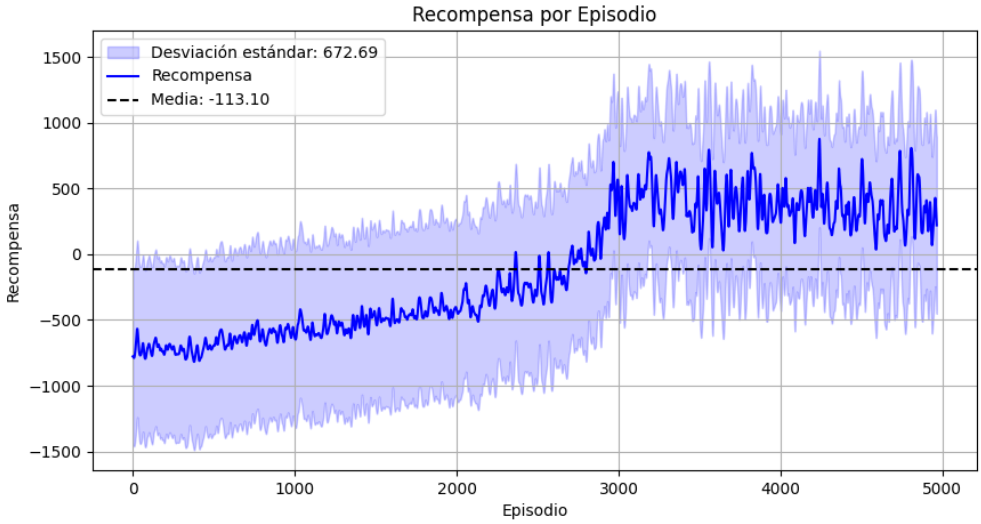


Figura 50: Gráfico Episodio Recompensa del entrenamiento con dificultad 1 en el escenario Deadly Corridor con DQN. Fuente: Autoría Propia.

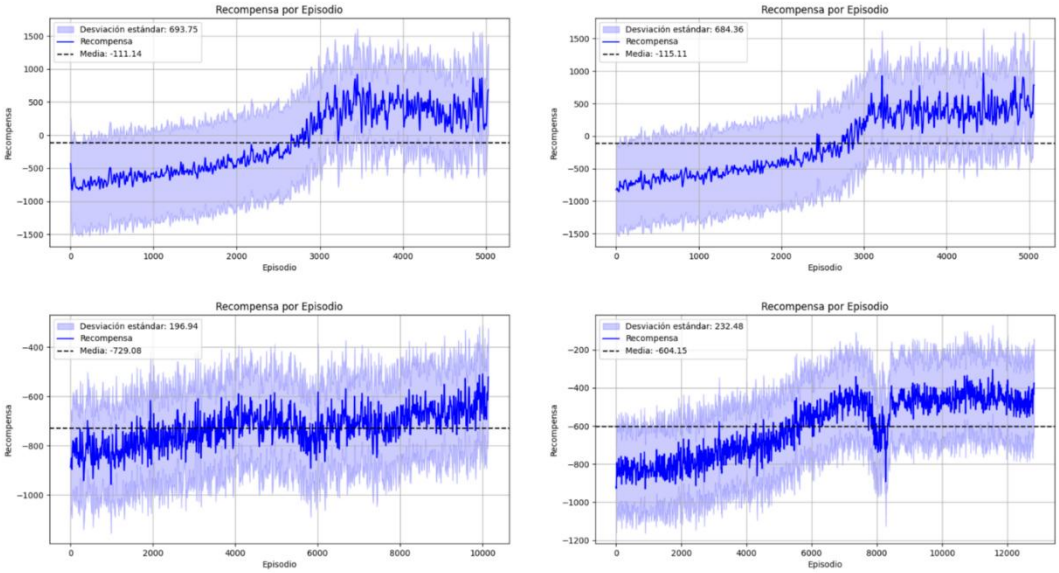


Figura 51: Gráfico Episodio Recompensa del entrenamiento con dificultades variadas en el escenario Deadly Corridor con DQN. De arriba abajo y de izquierda a derecha son dificultades 2, 3, 4, 5. Fuente: Autoría Propia.

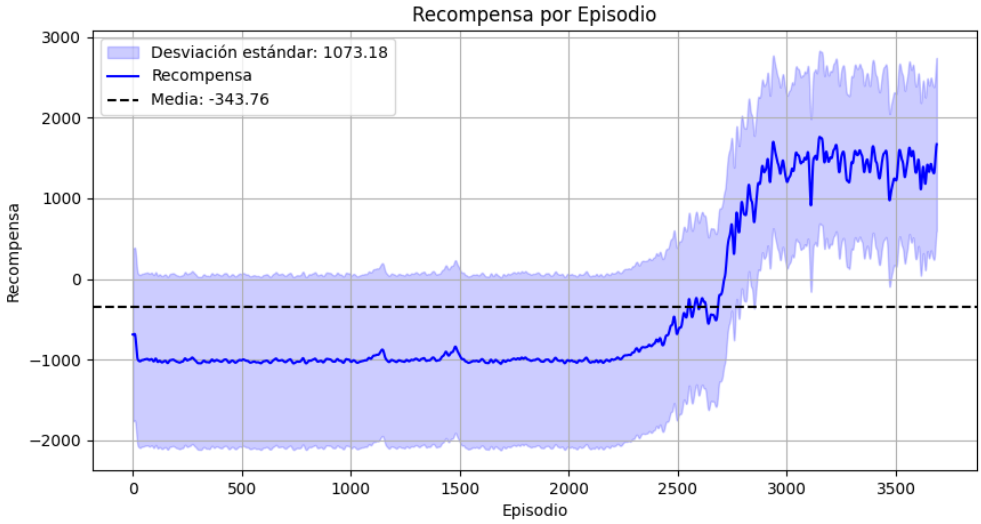


Figura 52: Gráfico Episodio Recompensa del entrenamiento con dificultad 1 en el escenario Deadly Corridor con PPO. Fuente: Autoría Propia.

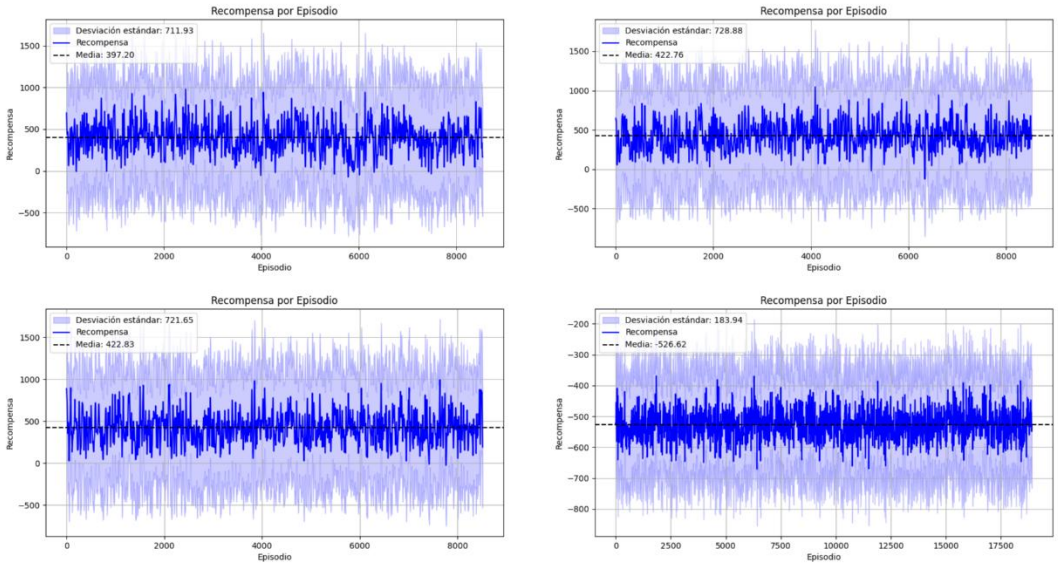


Figura 53: Gráfico Episodio Recompensa del entrenamiento con dificultades variadas en el escenario Deadly Corridor con PPO. De arriba abajo y de izquierda a derecha son dificultades 2, 3, 4, 5. Fuente: Autoría Propia.

Para mejorar el aprendizaje del agente, se implementó una estrategia de **Reward Shaping**, otorgando recompensas por eliminar enemigos y penalizaciones por recibir daño o disparar sin impactar. Esta técnica buscaba fomentar la interacción activa con los enemigos. Sin embargo, en algunos casos, esto derivó en un comportamiento no deseado: el agente se quedaba quieto al inicio del mapa esperando a que los enemigos reaparecieran, con el fin de eliminarlos repetidamente y así maximizar la recompensa, sin avanzar hacia la armadura.

Durante los entrenamientos se detectó también una limitación técnica relacionada con la biblioteca **SB3**: al cargar modelos previamente entrenados, los hiperparámetros no se reiniciaban automáticamente. Esto generaba que un agente entrenado en una dificultad baja mantuviera configuraciones subóptimas al pasar a niveles superiores. Para resolver esta limitación, se optó por guardar únicamente los pesos del modelo entrenado y cargarlos manualmente en nuevos entrenamientos, lo que permitió configurar los hiperparámetros desde cero sin arrastrar ajustes inadecuados.

Una vez solucionado este inconveniente, se decidió entrenar agentes en niveles de dificultad media, restringiendo su espacio de acción para que se enfocaran únicamente en disparar sin necesidad de desplazarse (**Figura 54 y 55**). Esta estrategia, combinada con el **Reward Shaping**, contribuyó significativamente a mejorar la puntería del agente. También se exploró la reutilización de pesos de agentes entrenados en escenarios similares. Ambas estrategias mostraron avances, especialmente en la eliminación temprana de enemigos, aunque no se evidenció una diferencia significativa entre ellas. Por esta razón, se optó por continuar con el modelo más consistente del primer experimento exitoso.

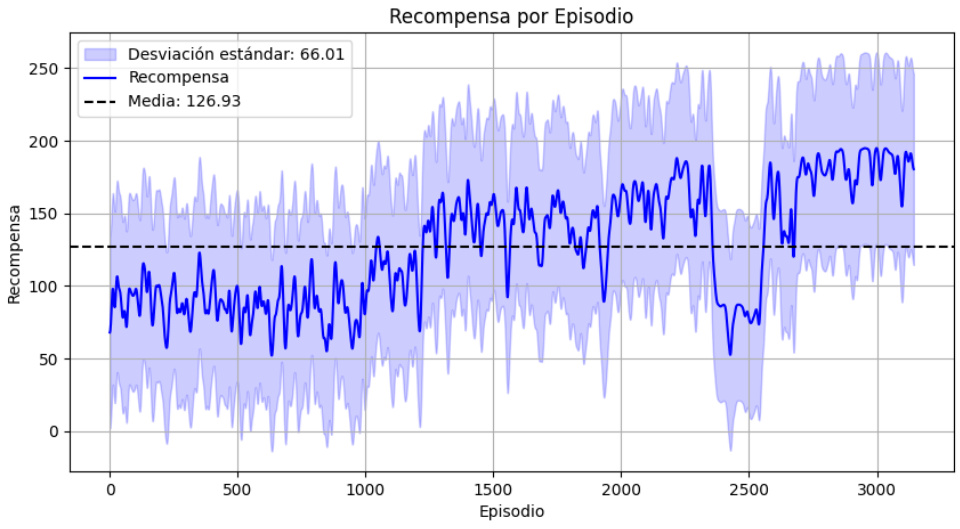


Figura 54: Gráfico Episodio Recompensa del entrenamiento solo disparo en el escenario Deadly Corridor con DQN. Fuente: Autoría Propia.

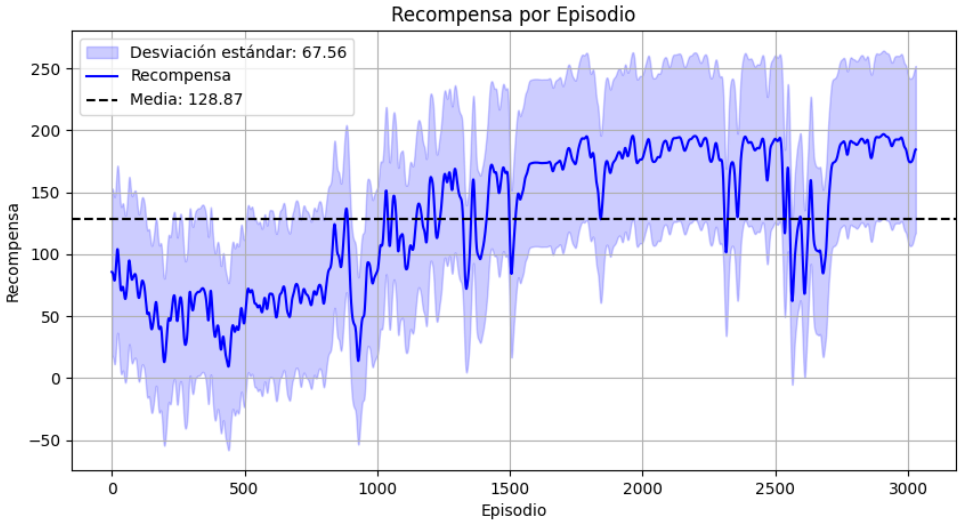


Figura 55: Gráfico Episodio Recompensa del entrenamiento solo disparo en el escenario Deadly Corridor con PPO. Fuente: Autoría Propia.

Finalmente, con todas estas mejoras implementadas, se continuaron los entrenamientos en niveles de dificultad elevada y con el espacio de acción completo. Esta etapa, junto con un ajuste manual de hiperparámetros, siendo clave la reducción del **Learning Rate**, que permitió al agente ajustar su política de manera más gradual y precisa, permitió que el agente evolucionara hacia un comportamiento más alineado con los objetivos del escenario. En esta fase, el agente demostró una mejor capacidad para enfrentar situaciones de combate complejas, eliminar enemigos con precisión y llegar hasta la armadura sin morir. (**Figura 56 y 57**).

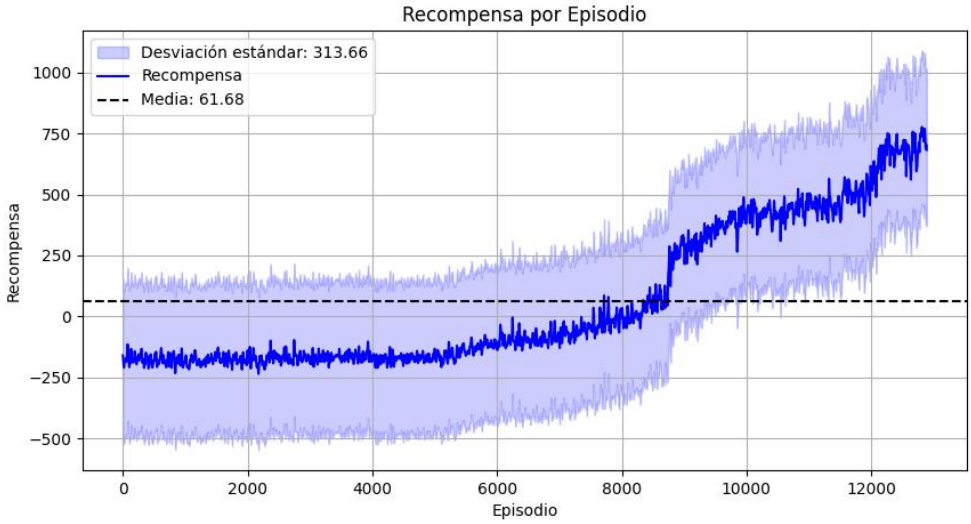


Figura 56: Gráfico Episodio Recompensa del entrenamiento final en el escenario Deadly Corridor con DQN. Fuente: Autoría Propia.

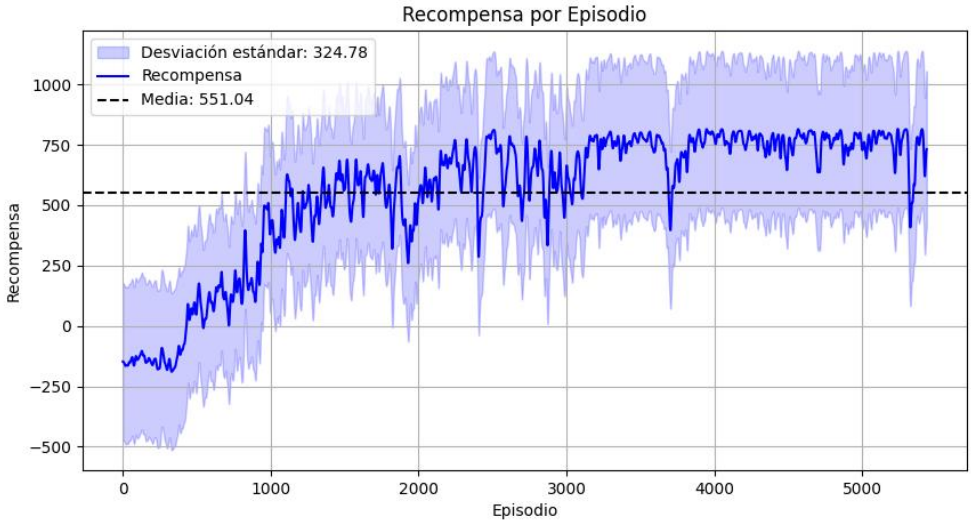


Figura 57: Gráfico Episodio Recompensa del entrenamiento final en el escenario Deadly Corridor con PPO. Fuente: Autoría Propia.

4.3.3.2. Deathmatch

El escenario **Deathmatch** presentó múltiples desafíos debido a la combinación de exploración, eliminación de enemigos y recolección de recursos. A lo largo del desarrollo del proyecto, se realizaron diversas pruebas para mejorar el desempeño del agente en este entorno. En las primeras fases de experimentación, no se realizaron modificaciones significativas en los parámetros de entrenamiento, limitándose únicamente a extender el tiempo de entrenamiento. Sin embargo, como era previsible, los resultados obtenidos no fueron favorables. (**Figura 58 y 59**).

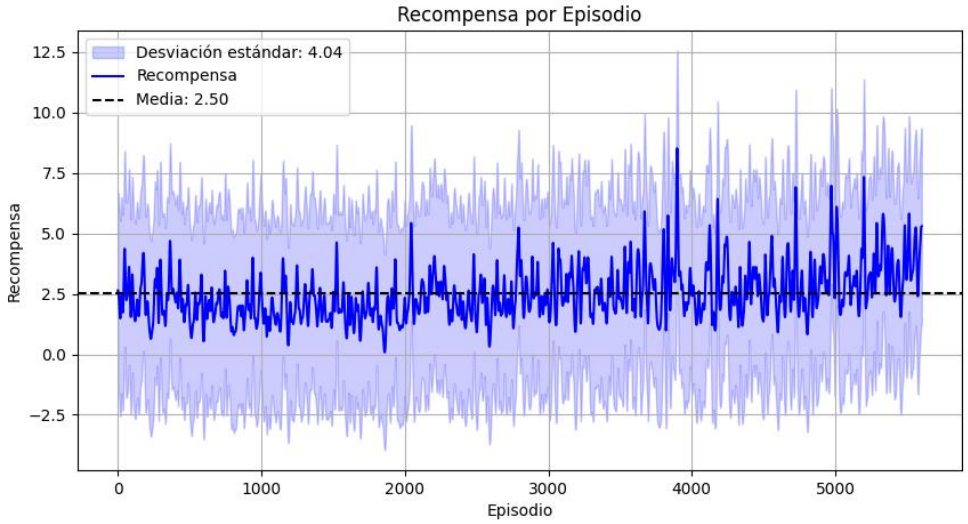


Figura 58: Gráfico Episodio Recompensa del entrenamiento en el escenario Deathmatch con DQN. Fuente: Autoría Propia.

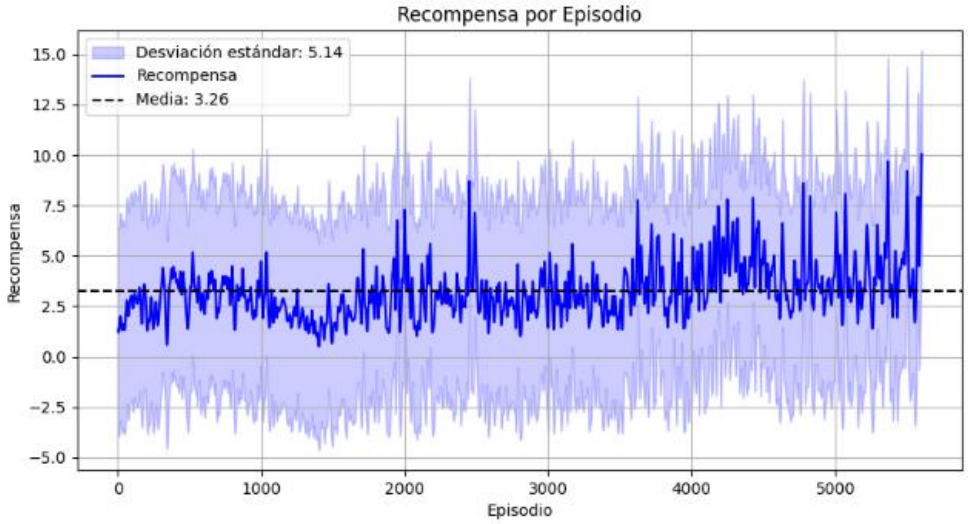


Figura 59: Gráfico Episodio Recompensa del entrenamiento en el escenario Deathmatch con PPO. Fuente: Autoría Propia.

Para abordar la complejidad del mapa, se introdujeron ajustes en la configuración del agente. Se aumentó el **Frame Skip** de 4 a 7, tomando como referencia estudios previos donde esta modificación había mostrado mejoras en mapas de exploración. Adicionalmente, se ajustaron parámetros clave como la tasa de aprendizaje y el valor de **ε** para **DQN**. También se limitó el espacio de acción eliminando botones innecesarios, como la opción de correr o cambios de arma y así tener un entrenamiento con acciones acotadas. A pesar de estas modificaciones, los resultados continuaron siendo insatisfactorios. (**Figura 60 y 61**).

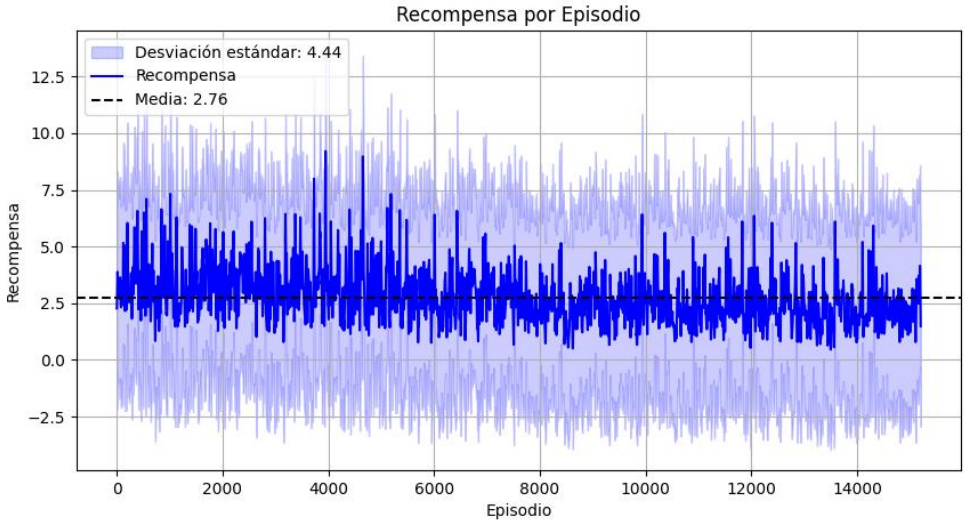


Figura 60: Gráfico Episodio Recompensa del entrenamiento en el escenario Deathmatch con DQN usando el triple de Time Steps y Frame Skip de 4. Fuente: Autoría Propia.

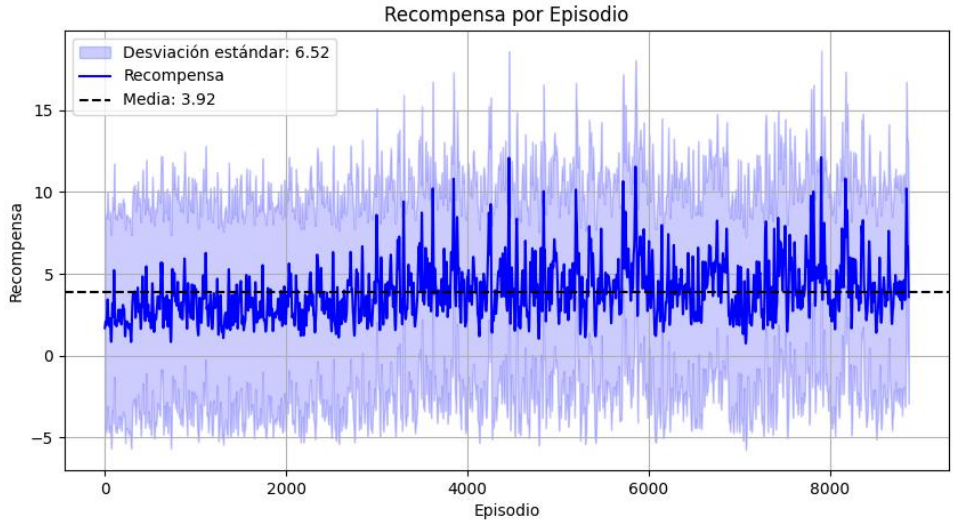


Figura 61: Gráfico Episodio Recompensa del entrenamiento en el escenario Deathmatch con DQN usando un Frame Skip de 7. Fuente: Autoría Propia.

Durante las pruebas, se observaron comportamientos emergentes en el agente, como la eliminación de enemigos sin una estrategia visual clara o la tendencia a permanecer en salas específicas que contenían múltiples recursos, lo que sugiere un patrón aprendido basado en la disponibilidad de ítems. Basándose en un artículo sobre este escenario (Lample & Chaplot, 2016), se decidió implementar técnicas avanzadas de entrenamiento para mejorar el desempeño del agente.

El artículo consultado proponía tres técnicas principales además de una configuración personalizada de parámetros, que incluía un alto número de **Time Steps**. La primera técnica consistía en **Reward Shaping**, similar a lo aplicado en **Deadly Corridor**, lo que facilitó su implementación en **Deathmatch**. La segunda técnica proponía el uso de dos modelos independientes, uno enfocado en la exploración y otro en el combate. Este enfoque permitía que el agente alternara entre ambos modelos según fuera necesario, optimizando su desempeño al no estar permanentemente en modo de combate. La tercera técnica añadía un sistema de asistencia en la puntería, ajustando automáticamente la mira cuando un enemigo aparecía en pantalla, facilitando así el aprendizaje de apuntar y disparar con precisión. Estas últimas dos técnicas no fueron implementadas, ya que, si bien podrían mejorar el desempeño del agente, introducen un nivel adicional de complejidad y asistencia externa que se aleja del objetivo principal del estudio: evaluar y comparar técnicas de aprendizaje por refuerzo profundo en condiciones lo más generales y comparables posible. Por lo tanto, se optó por mantener el diseño experimental más limpio y centrado en las capacidades propias de cada técnica.

En su lugar, se realizaron experimentos aplicando tres modificaciones: incremento del tiempo de entrenamiento, aumento de la memoria en **DQN** y uso de **Reward Shaping**. Sin embargo, estos enfoques no generaron mejoras significativas. Primero, el sistema no soportaba el incremento en la memoria recomendado en el estudio previo, por lo que fue necesario reducirla. Además, el alto número de **Time Steps** recomendado generaba interrupciones en el entrenamiento, lo que llevó a restringir estos valores debido a limitaciones del sistema, posiblemente afectando el rendimiento. Aunque el **Reward Shaping** tuvo cierto impacto positivo, la mejora general en el entrenamiento de **Deathmatch** fue mínima. (**Figura 62 y 63**)

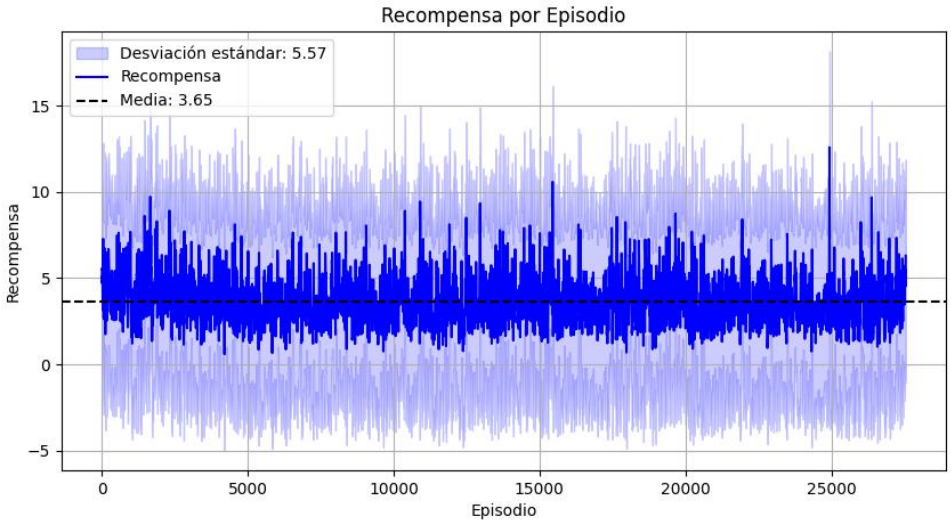


Figura 62: Gráfico Episodio Recompensa del entrenamiento en el escenario Deathmatch con DQN con modificaciones. Fuente: Autoría Propia.

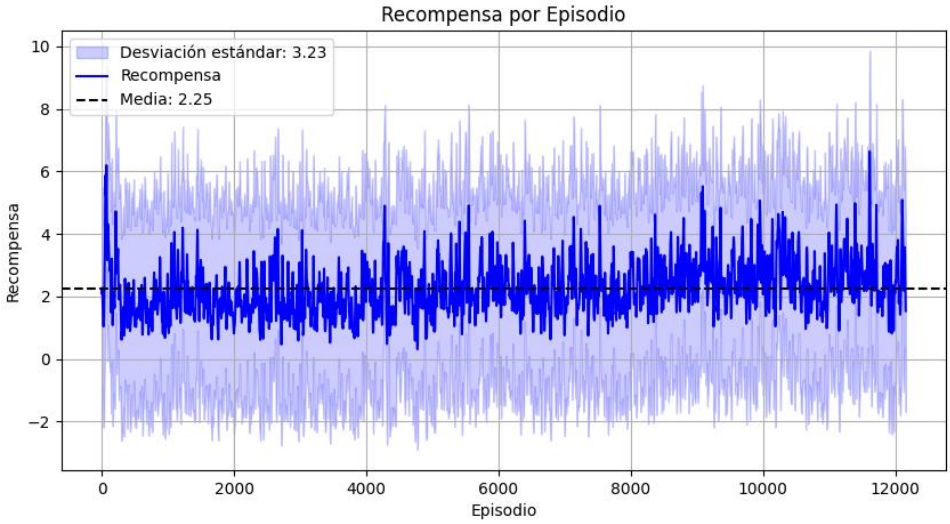


Figura 63: Gráfico Episodio Recompensa del entrenamiento en el escenario Deathmatch con PPO con modificaciones. Fuente: Autoría Propia.

Adicionalmente, se exploró la posibilidad de proporcionar al agente un arma más efectiva desde el inicio, como un lanzamisiles, para facilitar la eliminación de enemigos. Se investigaron métodos para modificar la asignación de armas dentro del código .wad utilizando el software **SLADE**, pero esta modificación no tuvo éxito. También se intentó ajustar la configuración del mapa mediante comandos específicos de **DOOM**, pero la única opción funcional fue la manipulación de la cantidad de munición, lo que no era suficiente para impactar significativamente el rendimiento del agente. Dado que no se encontraron métodos efectivos para modificar las armas iniciales, esta estrategia fue descartada.

5. Resultados

5.1.1. Escenarios de Disparo

5.1.1.1. Defend the Center

Este fue el primer escenario utilizado para realizar pruebas, sirviendo como base para testear los distintos cambios implementados en el código. Tal como se explicó previamente, los resultados iniciales no fueron favorables. No obstante, tras diversos experimentos y ajustes especialmente en los hiperparámetros y en la estrategia **ε-greedy** se logró que el agente cumpliera con éxito el objetivo del mapa: sobrevivir el mayor tiempo posible.

El agente entrenado con **DQN** alcanzó recompensas aceptables, aunque presentó ciertas limitaciones. Su puntería era imprecisa, especialmente con enemigos a larga distancia, lo que resultaba en un desperdicio considerable de munición. Además, **DQN** mostraba una conducta pasiva: una vez eliminaba a los enemigos directamente frente a él, dejaba de mover la cámara, esperando a que nuevos enemigos aparecieran en su campo de visión o hasta recibir daño desde los extremos o desde atrás.

Por el contrario, el agente entrenado con **PPO** demostró un comportamiento más efectivo. Su puntería era significativamente más precisa, lo que reducía el uso innecesario de munición. Además, la cámara se mantenía en constante movimiento, lo que facilitaba la detección de enemigos en todas las direcciones y promovía una estrategia de búsqueda activa.

En resumen, ambos algoritmos lograron cumplir con el objetivo del escenario, pero **PPO** obtuvo un rendimiento superior gracias a su mejor precisión y su capacidad para detectar y eliminar enemigos con mayor eficiencia.

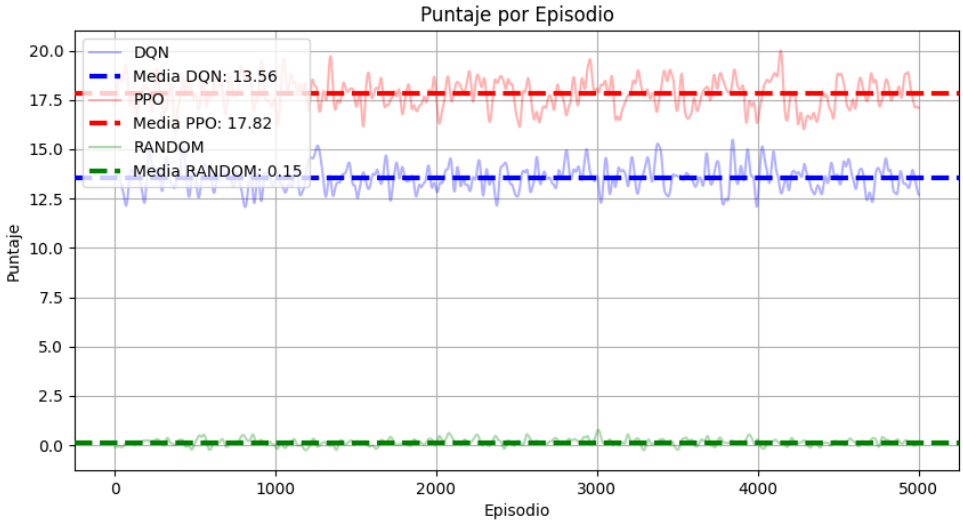


Figura 64: Gráfico Episodio Puntaje de la evaluación en el escenario Defend the Center con la media de los métodos DQN y PPO con la comparación de un agente aleatorio. Fuente: Autoría Propia.

5.1.1.2. Defend the Line

Este escenario presenta una mecánica similar a la de **Defend the Center**, donde el objetivo principal es sobrevivir el mayor tiempo posible eliminando enemigos. La estrategia ideal consiste en priorizar a los enemigos que disparan primero, seguidos por aquellos que simplemente se acercan al agente.

Tanto **DQN** como **PPO** obtuvieron resultados similares en este escenario. Ambos agentes lograron desempeñarse adecuadamente, eliminando a los enemigos que se presentaban y prolongando su supervivencia de manera efectiva. Si bien se observaron ligeras diferencias en el comportamiento, estas no fueron significativas: **DQN** tendía a mantener la mira centrada, realizando movimientos laterales limitados, mientras que **PPO** presentaba un movimiento más dinámico de cámara. Sin embargo, no se pudo establecer con certeza si alguno de los agentes lograba priorizar consistentemente a los enemigos que disparaban, lo cual representa la estrategia óptima en este escenario.

En conclusión, ambos agentes cumplieron con el objetivo del escenario, pero no se observaron diferencias claras en su desempeño ni en la priorización de amenazas, siendo esta una posible área de mejora en futuros entrenamientos.

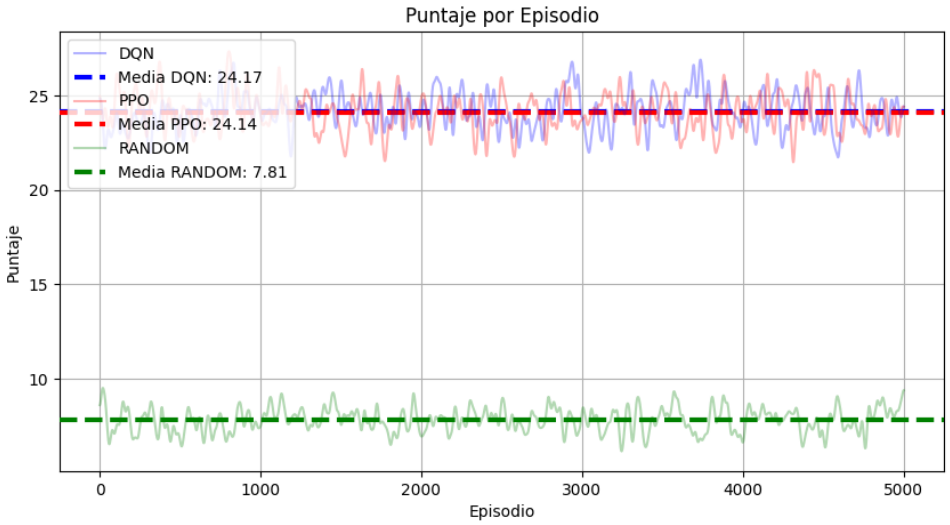


Figura 65: Gráfico Episodio Puntaje de la evaluación en el escenario Defend the Line con la media de los métodos DQN y PPO con la comparación de un agente aleatorio. Fuente: Autoría Propia.

5.1.1.3. Predict Position

En este escenario, el objetivo es que el agente aprenda a predecir el movimiento del enemigo y calcular el momento preciso para disparar un lanzamisiles, cuyo proyectil tiene un retardo antes de impactar. Esto exige una coordinación temporal más compleja que en otros mapas donde el disparo es instantáneo.

El agente entrenado con **PPO** logra cumplir satisfactoriamente con este objetivo, mostrando un comportamiento en el que sigue el movimiento del enemigo y dispara en el momento justo. La mayoría de los episodios reflejan esta conducta eficiente, aunque ocasionalmente se observan errores por precipitaciones al disparar sin seguir bien al objetivo, o por movimientos aleatorios y bruscos con la cámara. Este comportamiento podría mejorar con un ajuste fino del entrenamiento, orientado a suavizar el movimiento del agente.

Por otro lado, **DQN** no logra cumplir con el objetivo del escenario. Su estrategia aprendida consiste en disparar de forma automática al comienzo de cada episodio, apuntando en una dirección fija, sin verificar si el enemigo está en la mira. Aunque esta estrategia genera alguna recompensa cuando el enemigo coincide por azar con la trayectoria del misil, no implica una verdadera predicción del movimiento, sino que responde a una mecánica del mapa: disparar rápidamente minimiza la penalización por inactividad. Esto sugiere que el agente encontró una manera de minimizar la pérdida de recompensa sin aprender la tarea real.

Aun así, se observa que en algunos episodios **DQN** sigue brevemente al enemigo con la mira antes de disparar, mostrando cierta intención de interacción, aunque sin éxito en la ejecución.

En resumen, **PPO** cumple de forma clara con el objetivo del escenario, mostrando una estrategia alineada con lo requerido: anticiparse al movimiento del enemigo y disparar en el momento oportuno. **DQN**, aunque presenta comportamientos que podrían indicar una intención de seguimiento, adopta una estrategia que maximiza la recompensa sin aprender la lógica del mapa.

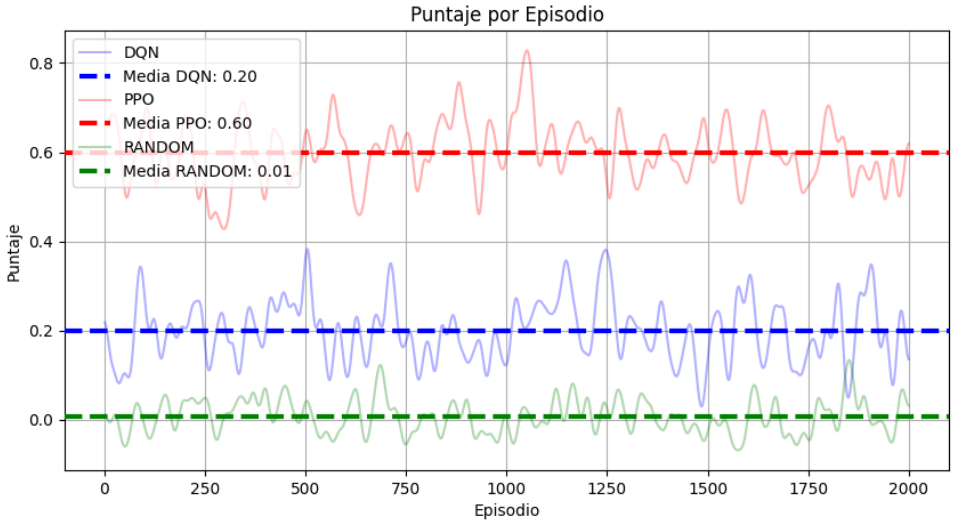


Figura 66: Gráfico Episodio Puntaje de la evaluación en el escenario Predict Position con la media de los métodos DQN y PPO con la comparación de un agente aleatorio. Fuente: Autoría Propia.

5.1.2. Escenarios de Exploración y Supervivencia

5.1.2.1. Health Gathering

Este escenario se centra en la exploración y la supervivencia, donde el agente debe recoger botiquines repartidos aleatoriamente por el mapa para contrarrestar el daño constante producido por el entorno. El objetivo es sobrevivir el mayor tiempo posible, siendo el comportamiento ideal aquel en el que el agente se desplaza de manera eficiente entre los botiquines, sin perder tiempo ni quedar atrapado.

En los entrenamientos realizados, ambos métodos **DQN** y **PPO** lograron aprender una estrategia sólida: evitar colisiones con las paredes y recorrer el borde del mapa en movimiento constante. Justo antes de colisionar con una pared, el agente gira 90 grados y continúa su trayecto, lo que le permite cubrir el escenario de forma efectiva. Dado que los botiquines aparecen aleatoriamente, este patrón de movimiento asegura que el agente recoja la mayoría de ellos, manteniendo su salud en niveles óptimos durante gran parte del episodio.

Aunque ambos métodos cumplen exitosamente con el objetivo, **PPO** obtiene un puntaje promedio superior. Esto puede atribuirse a varios factores: **PPO** realiza giros más suaves y mantiene una trayectoria más centrada en el mapa, lo cual podría incrementar las probabilidades de recoger botiquines. En contraste, **DQN** tiende a desplazarse más cerca de las paredes, donde la frecuencia de aparición de botiquines puede ser menor. Además, es importante señalar que el agente **DQN** evaluado corresponde al modelo guardado en el mejor episodio, no al último del entrenamiento. Esto se debe a que el modelo final mostraba un comportamiento no deseado, quedándose en el centro del mapa girando en círculo, obteniendo así el puntaje mínimo. Este comportamiento podría estar relacionado con cómo **SB3** gestiona la selección de los “mejores modelos”, evaluando en ventanas promedio de varios episodios donde un solo episodio bueno podría no ser suficiente para influir significativamente en el promedio general.

En conclusión, ambos agentes cumplen con el objetivo principal del mapa, demostrando una estrategia de navegación efectiva y una buena capacidad de supervivencia, con **PPO** mostrando un rendimiento más consistente y eficiente en términos de puntaje.

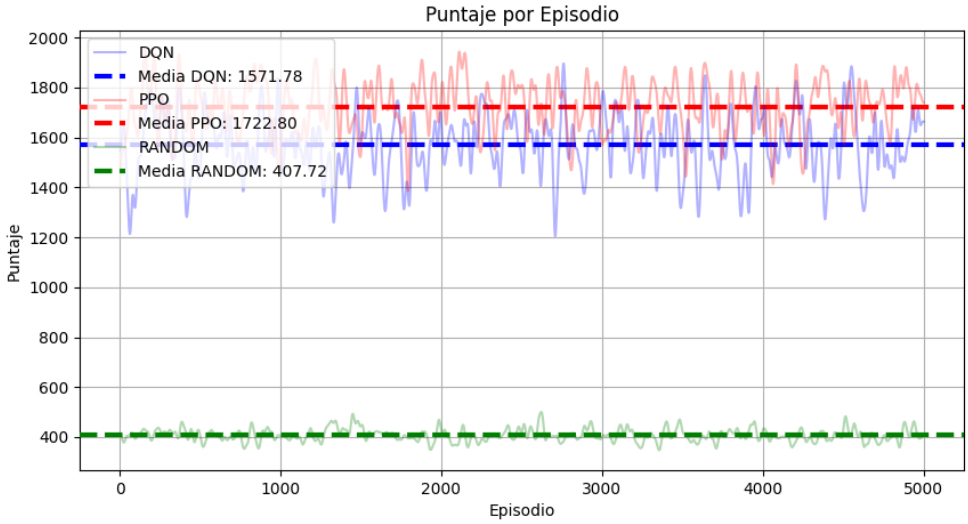


Figura 67: Gráfico Episodio Puntaje de la evaluación en el escenario Health Gathering con la media de los métodos DQN y PPO con la comparación de un agente aleatorio. Fuente: Autoría Propia.

5.1.2.2. My Way Home

Este escenario tiene como objetivo que el agente llegue a una sala específica del laberinto donde se encuentra una armadura, optimizando el tiempo y evitando quedar atrapado en otras habitaciones. El comportamiento ideal es que el agente reconozca la estructura del laberinto y llegue directamente al objetivo sin desviaciones innecesarias.

Durante los entrenamientos, se observó una marcada diferencia entre los dos métodos. El agente entrenado con **PPO** logró aprender la ruta hacia la armadura de forma eficiente, desplazándose directamente hacia la sala objetivo. Si bien en ocasiones colisiona con algunas paredes, el agente aprende a corregir su trayectoria, evitando quedar atrapado, lo cual le permite cumplir consistentemente con la tarea.

Por el contrario, el agente entrenado con **DQN** no logró cumplir con el objetivo del escenario. A pesar de mostrar cierta intención en algunos episodios, como dirigirse a salas cercanas al objetivo, el agente no podía salir de habitaciones alejadas una vez que se encontraba dentro. Esto provocaba que quedara atascado, chocando continuamente contra las paredes, y finalizara el episodio sin llegar a la armadura. El resultado es que **DQN** obtiene un rendimiento comparable al de un agente que toma decisiones aleatorias, sin una estrategia clara para navegar el laberinto.

En conclusión, **PPO** logra resolver el escenario de forma exitosa, navegando por el laberinto sin quedar atascado y alcanzando el objetivo de manera consistente. En cambio, **DQN** falla en aprender una política efectiva, quedando muy por debajo del rendimiento esperado.

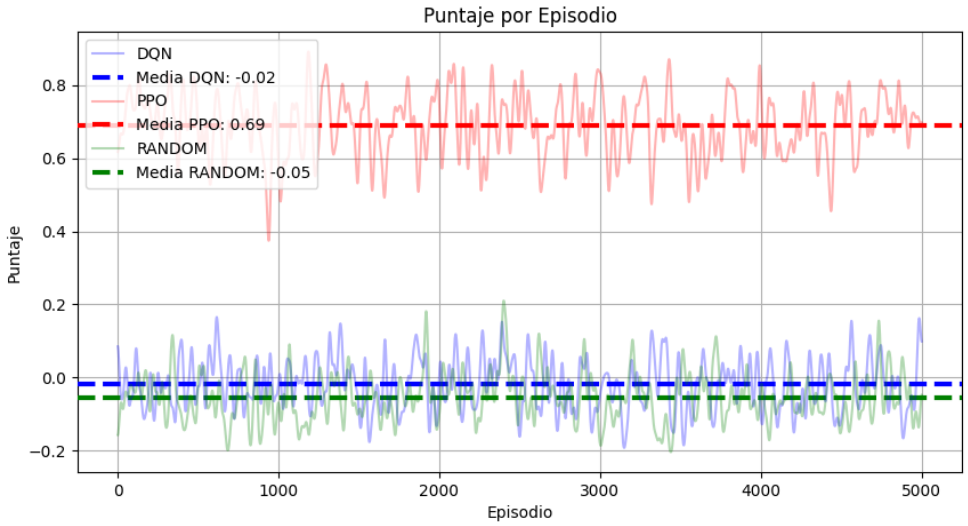


Figura 68: Gráfico Episodio Puntaje de la evaluación en el escenario My Way Home con la media de los métodos DQN y PPO con la comparación de un agente aleatorio. Fuente: Autoría Propia.

5.1.2.3. Take Cover

Este escenario tiene como propósito que el agente aprenda a esquivar los disparos enemigos, calculando el tiempo entre el disparo y la llegada del proyectil para moverse de manera eficiente. El caso ideal sería que el agente logre sobrevivir hasta el final del episodio sin recibir daño, aunque esto se complica a medida que aumenta la cantidad de enemigos disparando simultáneamente.

En los experimentos realizados, ambos métodos logran adaptarse al entorno y esquivar varios disparos, lo cual demuestra un aprendizaje efectivo. Sin embargo, hay diferencias importantes en sus estrategias. **DQN** tiende a esquivar los disparos cuando ya están cerca, lo que genera una mayor tasa de impacto. Además, se observa que el agente se detiene tras esquivar, esperando la siguiente ronda de disparos para volver a moverse, lo cual puede ser útil en ciertos momentos, pero también puede aumentar el riesgo.

En cambio, el agente entrenado con **PPO** se mantiene en constante movimiento, anticipándose a los disparos desde que el enemigo comienza a disparar, lo cual reduce su tasa de impacto. Esta estrategia demuestra una mayor capacidad de anticipación y reacción, generando recompensas más altas de forma consistente.

Cabe destacar que ambas estrategias tienen ventajas y desventajas. El movimiento constante de **PPO** puede provocar que los disparos se distribuyan aleatoriamente por el escenario, dificultando su evasión. Por otro lado, la estrategia de **DQN** de detenerse puede concentrar los disparos en una sola posición, facilitando esquivarlos si el agente logra moverse justo a tiempo.

En conclusión, ambos métodos cumplen con el objetivo del escenario, aunque con estrategias diferentes. **PPO** obtiene mejores resultados promedio gracias a su capacidad de anticipación, mientras que **DQN** muestra una estrategia más reactiva, que aún puede ser útil dependiendo de las condiciones. Ambos agentes muestran un comportamiento adaptativo que puede ser mejorado en entrenamientos futuros.

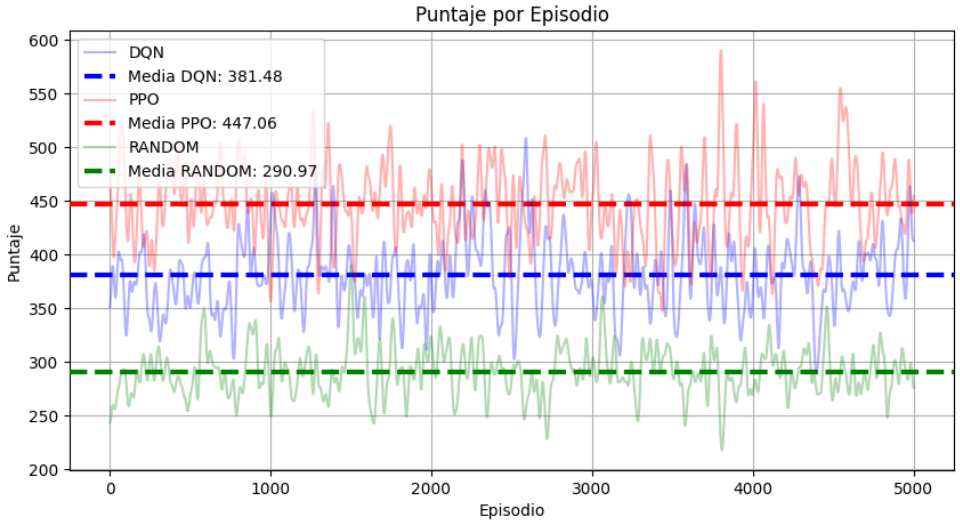


Figura 69: Gráfico Episodio Puntaje de la evaluación en el escenario Take Cover con la media de los métodos DQN y PPO con la comparación de un agente aleatorio. Fuente: Autoría Propia.

5.1.3. Escenarios Combinados

5.1.3.1. Deadly Corridor

En este escenario se llevaron a cabo múltiples experimentos, aplicando técnicas como **Reward Shaping**, **Parameter Tuning** y **Curriculum Learning**. Los mejores resultados se obtuvieron tras reducir significativamente el **Learning Rate** y aplicar recompensas adicionales por eliminar enemigos. Estas estrategias permitieron que el agente pudiera completar el objetivo de eliminar enemigos en las dificultades más altas y obtener la armadura de forma más segura, reduciendo la probabilidad de morir en el intento.

En el caso de **DQN**, el agente logró eliminar a los enemigos en los niveles de mayor dificultad, sin embargo, no siempre alcanzaba la armadura. Esto ocurría por dos razones: en algunos casos, el agente perdía la vida antes de llegar; en otros, permanecía estático en el centro del pasillo. Se plantea como hipótesis que este comportamiento podría deberse al **Reward Shaping**: al aprender que eliminar enemigos otorga una alta recompensa, el agente podría haber desarrollado una política de esperar la reaparición de enemigos para seguir acumulando puntos, ignorando el verdadero objetivo del escenario.

Por otro lado, el comportamiento de **PPO** fue más alineado con los objetivos del mapa. El agente no solo eliminaba a los enemigos con alta precisión frecuentemente con un solo disparo, sino que también se dirigía directamente a la armadura sin demoras innecesarias ni acciones improductivas. Este comportamiento se mantuvo en la mayoría de los episodios, incluso en los de mayor dificultad.

En conclusión, **PPO** logró cumplir de manera efectiva con el objetivo principal del escenario, alcanzando la armadura sin morir incluso en la dificultad máxima. En contraste, **DQN** no consiguió completar el escenario en sus últimos entrenamientos, desarrollando en su lugar una estrategia alternativa centrada en maximizar recompensas mediante la eliminación repetida de enemigos, lo cual, si bien puede parecer eficiente desde el punto de vista de la recompensa, no representa el objetivo real del mapa y es una consecuencia del enfoque utilizado en el **Reward Shaping**.

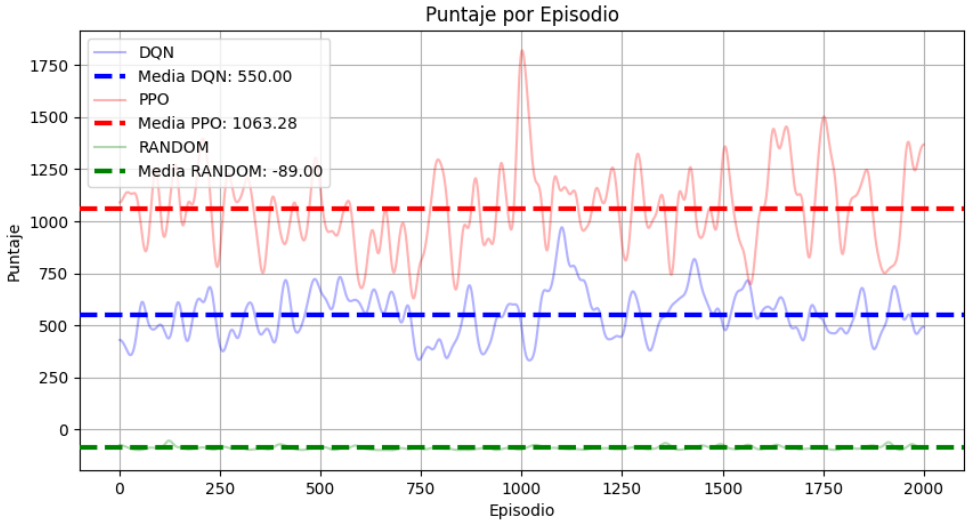


Figura 70: Gráfico Episodio Puntaje de la evaluación en el escenario Deadly Corridor con la media de los métodos DQN y PPO con la comparación de un agente aleatorio. Fuente: Autoría Propia.

5.1.3.2. Deathmatch

Tal como se anticipó en la sección de Desarrollo, el escenario **Deathmatch** no arrojó los resultados esperados. Los agentes entrenados con ambos métodos no lograron adaptarse adecuadamente a los desafíos del entorno, mostrando un comportamiento más cercano al de un agente aleatorio que al de uno entrenado. Aunque en algunos casos se observaban acciones que podrían interpretarse como dirigidas a un objetivo, como desplazarse hacia una sala con objetos al inicio del episodio, estas eran muy puntuales y no reflejaban un aprendizaje consistente.

Como se muestra en el gráfico correspondiente, **Deathmatch** es el único escenario en el que el agente entrenado con **DQN** obtuvo un promedio de recompensa superior al de **PPO**. Sin embargo, este resultado no implica un mejor desempeño real, ya que incluso una política aleatoria logra un puntaje comparable. Esto podría deberse a que **DQN** tiende a mostrar un rendimiento relativamente más estable en etapas iniciales del entrenamiento, mientras que **PPO** requiere mayor tiempo de interacción para desarrollar estrategias efectivas. No obstante, ambos métodos mostraron rendimientos bajos, ya que las pocas recompensas obtenidas fueron mayoritariamente por recoger ítems, y no por cumplir el objetivo principal: eliminar enemigos. De hecho, no se observó un comportamiento consistente de combate, y el agente rara vez apuntaba o disparaba correctamente.

En conclusión, **Deathmatch** resultó ser un escenario especialmente desafiante. A pesar de que ambos agentes fueron entrenados bajo condiciones equivalentes, ninguno logró adaptarse eficazmente. Este resultado refleja las limitaciones de las técnicas utilizadas en contextos complejos como este, pero también deja abierta la posibilidad de mejoras. Como se mencionó en la sección de Desarrollo, existen técnicas adicionales, como modelos especializados por tarea o asistencia en la puntería, que podrían contribuir a un mejor rendimiento en este tipo de entornos.

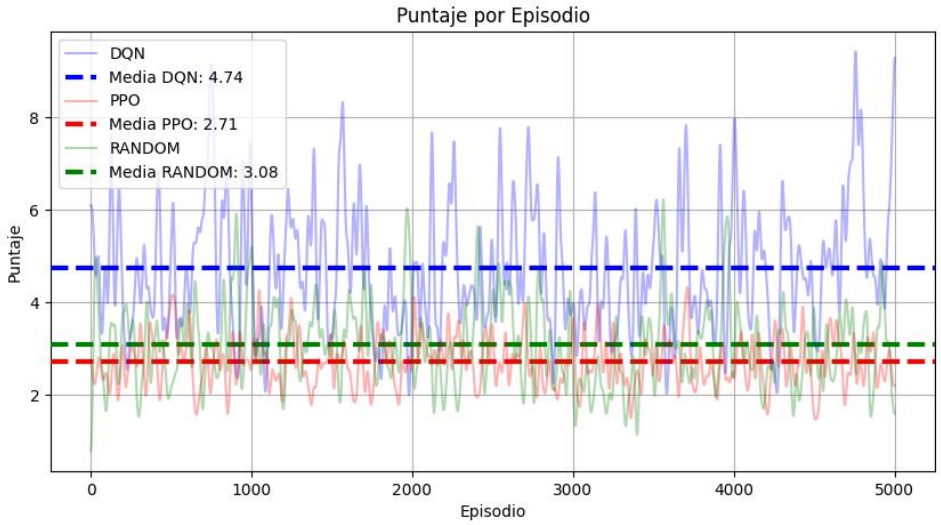


Figura 71: Gráfico Episodio Puntaje de la evaluación en el escenario Deathmatch con la media de los métodos DQN y PPO con la comparación de un agente aleatorio. Fuente: Autoría Propia.

5.2.1. Tabla Resumen

A continuación, se presenta una tabla resumen con los resultados obtenidos en cada escenario. Se incluye tanto la recompensa promedio alcanzada por cada método como una evaluación cualitativa de su comportamiento, lo que permite observar las diferencias en desempeño entre DQN y PPO en los distintos entornos evaluados.

Escenario	Método	Recompensa Promedio	Evaluación Cualitativa
Defend the Center	DQN	13.56	Cumple el objetivo, pero con puntería imprecisa. Espera en lugar de buscar enemigos activamente.
	PPO	17.82	Estrategia más proactiva y eficiente. Mejor puntería y movimiento constante que facilita el ver a los enemigos.
Defend the Line	DQN	24.17	Comportamiento correcto, pero con movimiento más limitado. No se observó una priorización clara de los enemigos.
	PPO	24.14	Movimiento de cámara más dinámico. Rendimiento similar a DQN, sin diferencia en la estrategia aplicada.
Predict Position	DQN	0.20	No logra cumplir con el objetivo. Dispara en momentos fijos sin seguir al enemigo. La conducta muestra una evasión de penalización.
	PPO	0.60	Comportamiento correcto. Anticipa el movimiento del enemigo y dispara en el momento adecuado, con pequeños errores puntuales.
Health Gathering	DQN	1571.78	Navegación funcional, pero su trayectoria es menos eficiente y se acerca más a las paredes.
	PPO	1722.80	Movimiento más fluido y centrado. Navegación eficiente que resulta en mayor supervivencia.
My Way Home	DQN	-0.02	No logra llegar al objetivo. Se atasca frecuentemente en habitaciones.
	PPO	0.69	Aprende la ruta optima y navega el laberinto correctamente. Cumple consistentemente con la tarea.
Take Cover	DQN	381.48	Reacción tardía a disparos. Estrategia basada en esquivar y detenerse, con resultados aceptables, pero con mayor exposición.

	PPO	447.06	Movimiento constante y anticipado. Menor probabilidad que le impacte y mejor adaptación al objetivo del escenario.
Deadly Corridor	DQN	550	Elimina enemigos eficientemente, pero a costa del objetivo real. A veces permanece estático. Influenciado por Reward Shaping.
	PPO	1063	Se dirige directamente a la armadura tras eliminar enemigos. Precisión alta y acciones consistentes con los objetivos.
Deathmatch	DQN	4.74	Obtiene una recompensa ligeramente mayor que PPO, pero el comportamiento sigue siendo aleatorio. Recoge ítems sin mostrar combate efectivo.
	PPO	2.71	Bajo desempeño. No logra adaptarse al entorno ni desarrollar estrategias de ataque. Las acciones son inconsistentes y poco orientadas al objetivo.

Tabla 2: Tabla Resumen de Resultados con su Recompensa Promedio y la Evaluación Cualitativa del Agente.

6. Conclusiones

Este proyecto tuvo como objetivo evaluar y comparar el desempeño de los algoritmos **Deep Q-Network** y **Proximal Policy Optimization** en distintos escenarios del videojuego **DOOM**, utilizando el entorno **ViZDoom**. A través de una implementación estructurada y una serie de experimentos, se buscó analizar cómo estas técnicas de aprendizaje por refuerzo profundo enfrentan tareas complejas que combinan combate, exploración y supervivencia.

El desarrollo comenzó con una etapa de familiarización con **ViZDoom**, donde se identificaron y clasificaron los mapas según el tipo de tarea y nivel de dificultad. En esta fase inicial también se implementó **DQN** desde cero utilizando **PyTorch**, pero los primeros entrenamientos no arrojaron buenos resultados. Esto evidenció la complejidad de configurar correctamente todos los componentes de un algoritmo de RL, especialmente aspectos como la arquitectura de red, el buffer de experiencias y la política de actualización. Por esta razón, se optó por utilizar las implementaciones estables provistas por la biblioteca **Stable-Baselines3**, lo que permitió centrarse en el análisis de desempeño en vez de los detalles técnicos de la implementación.

Para mejorar la eficiencia y estabilidad del entrenamiento, se aplicaron técnicas como **ajuste de hiperparámetros** usando la herramienta **Optuna**, y **Reward Shaping**, para personalizar las funciones de recompensa en escenarios donde la señal original era insuficiente para guiar correctamente al agente. Si bien estas estrategias ofrecieron mejoras notables en algunos casos, también se enfrentaron limitaciones importantes: el uso de **Optuna** resultó altamente costoso en términos de tiempo y recursos, especialmente en mapas de exploración, lo que obligó a simplificar ciertos procesos. A pesar de esto, el uso de funciones de recompensa personalizadas permitió guiar mejor a los agentes en entornos complejos como **Deadly Corridor** y **Deathmatch**.

Los resultados de los entrenamientos mostraron que **PPO** logró un rendimiento más consistente y robusto en la mayoría de los escenarios. En mapas que requerían exploración estructurada o toma de decisiones complejas, como **Health Gathering**, **My Way Home** o **Deadly Corridor**, **PPO** demostró una capacidad superior de adaptación, logrando políticas eficientes sin necesidad de ajustes excesivos. En cambio, **DQN** mostró un buen desempeño en mapas más simples, como **Defend the Center** y **Defend the Line**, pero su sensibilidad a los hiperparámetros y su tendencia a comportamientos pasivos o repetitivos lo limitaron en entornos más desafiantes.

En varios escenarios, sin embargo, los resultados no fueron satisfactorios. En **My Way Home**, por ejemplo, el agente entrenado con **DQN** fue incapaz de encontrar rutas efectivas hacia el objetivo. En **Deathmatch**, incluso aplicando técnicas como **Reward Shaping**, reducción del espacio de acción y aumento del tiempo de entrenamiento, los agentes no lograron desarrollar estrategias claras ni sostenidas. También se observaron comportamientos no deseados, como agentes que se quedaban quietos esperando enemigos para maximizar recompensas o que giraban sin rumbo dentro del mapa, lo que refleja la dificultad de alinear correctamente los incentivos con los objetivos reales del entorno.

Otro aspecto importante fue la aplicación de **Curriculum Learning**, entrenando primero en escenarios más simples o en niveles de menor dificultad para facilitar la transferencia de conocimiento hacia escenarios más exigentes. Esta estrategia mostró resultados mixtos: si bien permitió ciertos avances iniciales, también reveló limitaciones cuando el agente se especializaba en comportamientos útiles en mapas fáciles pero inútiles en dificultades más altas.

En definitiva, si bien el proyecto logró su objetivo principal de comparar y evaluar **DQN** y **PPO** en escenarios variados, los resultados dejan en claro que aún existe un amplio margen de mejora. Factores como la elección y personalización de recompensas, la gestión de la exploración, la arquitectura de red utilizada y el tiempo de entrenamiento son elementos críticos que influyen fuertemente en el desempeño del agente.

A futuro, sería interesante explorar técnicas complementarias como aprendizaje continuo, redes neuronales recurrentes, agentes multi-modelo (exploración vs combate), o incluso incorporar sistemas de asistencia como auto apuntado para facilitar la adquisición de habilidades clave. También sería útil implementar herramientas de análisis más profundas que permitan interpretar por qué un agente toma ciertas decisiones, o cómo mejora (o no) a lo largo del tiempo.

En términos generales, este estudio pone en evidencia la complejidad de entrenar agentes inteligentes en entornos dinámicos y de alta dimensionalidad como **ViZDoom**. Al mismo tiempo, demuestra que, mediante el uso adecuado de herramientas como el ajuste de parámetros, la modificación de recompensas y estrategias de entrenamiento progresivo es posible obtener agentes con comportamientos útiles y relativamente eficientes. **PPO** destacó por su robustez y capacidad de adaptación en la mayoría de los escenarios, mientras que **DQN**, aunque más limitado, mostró buen rendimiento en contextos más controlados. Si bien los resultados no son concluyentes a gran escala, el trabajo realizado entrega aprendizajes valiosos sobre el uso práctico de algoritmos de **DRL** en entornos complejos como **ViZDoom**, así como también evidencia las dificultades comunes en su implementación, ofreciendo un punto de partida útil para trabajos similares o más específicos en este entorno.

7. Bibliografía

1. Ibrahim, S., Mostafa, M., Jnadi, A., Salloum, H., & Osinenko, P. (2024). Comprehensive Overview of Reward Engineering and Shaping in Advancing Reinforcement Learning Applications. *IEEE Access*, *12*, 175473–175500. https://doi.org/10.1109/ACCESS.2024.3504735
2. id Software. (1993). *Doom*.
3. Kempka, M., Wydmuch, M., Runc, G., Toczek, J., & Jaśkowski, W. (2016). ViZDoom: A Doom-based AI research platform for visual reinforcement learning. *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. https://doi.org/10.1109/CIG.2016.7860433
4. Lample, G., & Chaplot, D. S. (2016). *Playing FPS Games with Deep Reinforcement Learning*. http://arxiv.org/abs/1609.05521
5. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236
6. Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., & Stone, P. (2020). Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. En *Journal of Machine Learning Research* (Vol. 21).
7. Polyphony Digital. (2022). *Gran Turismo 7*.
8. Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.
9. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. http://arxiv.org/abs/1707.06347
10. Tomilin, T., Fang, M., Zhang, Y., & Pechenizkiy, M. (2023). *COOM: A Game Benchmark for Continual Reinforcement Learning*.
11. Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., … Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354. https://doi.org/10.1038/s41586-019-1724-z
12. Wikipedia contributors. (2025). Doom (1993 video game). En *Wikipedia, The Free Encyclopedia*.
13. Wurman, P. R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T. J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., Gilpin, L., Khandelwal, P., Kompella, V., Lin, H., MacAlpine, P., Oller, D., Seno, T., Sherstan, C., Thomure, M. D., … Kitano, H. (2022). Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, *602*(7896), 223–228. https://doi.org/10.1038/s41586-021-04357-7
14. ZDoom. (s. f.). En *Zdoom.org*.

8. Anexo

Escenario	Método	Hiperparámetros Finales
Defend the Center	DQN	batch=32, lr=1e-4, buffer=15000, gamma=0.92, expl_frac=0.28, eps=0.016, start=1e4, decay=0 a 150000
	PPO	steps=2048, batch=64, lr=1e-4, gamma=0.92, lambda=0.95
Defend the Line	DQN	batch=32, lr=1e-4, buffer=10000, gamma=0.97, expl_frac=0.15, eps=0.01, start=1e4, decay=0 a 150000
	PPO	steps=2048, batch=64, lr=1e-4, gamma=0.94, lambda=0.93
Predict Position	DQN	batch=64, lr=1e-4, buffer=15000, gamma=0.99, expl_frac=0.4, eps=0.001, start=40000, decay=40000 a 350000
	PPO	steps=2048, batch=64, lr=1e-4, gamma=0.94, lambda=0.93
Health Gathering	DQN	batch=64, lr=1.81e-5, buffer=2976, gamma=0.966, expl_frac=0.179, eps=0.047, decay=358283 a 751885
	PPO	steps=2048, batch=64, lr=1e-5, gamma=0.9893, lambda=0.8687
My Way Home	DQN	batch=32, lr=1e-3, buffer=300000, gamma=0.99, eps=0.05, start=10000, decay=0 a 200000
	PPO	steps=2048, batch=64, lr=1e-5, gamma=0.9345, lambda=0.8147
Take Cover	DQN	batch=64, lr=1e-4, buffer=15000, gamma=0.98, expl_frac=0.4, eps=0.001, start=40000, decay=40000 a 300000
	PPO	steps=2048, batch=64, lr=1e-4, gamma=0.94, lambda=0.93
Deadly Corridor	DQN	batch=128, lr=5e-4, buffer=50000, gamma=0.99, eps=0.01, start=10000, decay=200000 a 700000
	PPO	steps=1024, batch=128, lr=5e-4, gamma=0.99, lambda=0.95, clip=0.2, ent=0.01, vf=0.5, clip_vf=0.2, kl=0.04
Deathmatch	DQN	batch=32, lr=1e-3, buffer=10000, gamma=0.99, eps=0.05, start=1000, decay=0 a 100000
	PPO	steps=2048, batch=64, lr=1e-4, gamma=0.94, lambda=0.93, clip=0.2, ent=1e-4, vf=0.5, clip_vf=0.2, kl=0.01

Tabla 3: Hiperparámetros Finales por Escenario y Método.