UNIVERSIDAD DE GRANADA

MÁSTER UNIVERSITARIO

EN CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES

Trabajo de Fin de Máster

Desarrollo y Optimización de un Agente Inteligente para Scripts of Tribute mediante Algoritmos Evolutivos

Autor

Francisco David Castejón Soto

Director

Dr. Pablo García Sánchez

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Granada, 1 Julio de 2025

Prefacio

Para la realización de este documento, se ha utilizado la plantilla LaTeX [9] específica para la ETSIIT.

Agradecimientos

A mi familia por el apoyo moral y económico que me han brindado de forma incondicional durante estos años fuera de casa. Y a mi pareja, por mostrarme más caminos de los que nunca hubiera pensado que existieran para mí.

Índice general

Índice de figuras

Índice de tablas

Parte I

Marco teórico

Capítulo 1

Introducción

La inteligencia artificial (IA) en videojuegos existe en una intersección entre la ciencia computacional y el arte del entretenimiento, dando vida a mundos virtuales y creando oponentes dignos de nuestras mejores estrategias. En este contexto, la IA se refiere únicamente a un conjunto de algoritmos y técnicas diseñadas para cumplir una función muy específica dentro del videojuego. Ejemplos de esto son la IA que "conduce" los coches en un juego de carreras, los Pokémon contra los que el jugador se enfrenta en una batalla o el simple movimiento de una línea de píxeles (a modo de pala contrincante) en el videojuego Pong. Esta acepción específica contrasta con la idea más extendida de IA, que posee un conjunto de definiciones más amplias y generalistas como "la máquina que aprende" o "el estudio y construcción de sistemas que hacen «lo correcto», en función de su objetivo" [**?**].

Independientemente de la definición que se utilice, lo cierto es que un factor clave en el desarrollo de un videojuego es la creación de una IA que resulte entretenida para el jugador. De la misma forma que en un libro o una película es necesaria una trama que plantee un desafío, como la lucha contra el Imperio Galáctico en Star Wars, o un compañero de aventuras que ayude al protagonista, como Sam en El Señor de los Anillos, en un videojuego es necesaria una IA que controle esos elementos que interactúan con el jugador, ya sean enemigos, aliados o incluso el propio entorno. Desde los propios inicios de la industria del videojuego, donde se construían máquinas específicas para ejecutar un juego concreto, como es el caso de Nim en 1948 [15], hasta los videojuegos modernos, donde se utilizan técnicas de IA más complejas, como el aprendizaje por refuerzo (más en la sección 3.3), la IA ha sido un componente esencial para crear experiencias de juego atractivas y desafiantes.

La intención de este trabajo es la de crear una IA contra la que jugar en un videojuego de estrategia. En concreto, la IA que controla a un agente autónomo (o simplemente "bot") en el videojuego de cartas *Tales of Tribute*. Es importante la distinción entre "juego" y "videojuego" en este caso, ya que las cartas no son

físicas, sino que únicamente existen en el universo digital del videojuego. Para guiar las decisiones del bot, un algoritmo evolutivo se encarga de ajustar los pesos que se utilizan para calcular la puntuación de cada jugada posible, eligiendo vorazmente la jugada que maximiza dicha puntuación. Así se ha conseguido crear un contrincante que juega de forma competitiva tanto contra el jugador humano como contra otros bots manejados por IAs totalmente diferentes.

Capítulo 2

Objetivos

2.1. Objetivo principal

El objetivo principal de este Trabajo de Fin de Máster se puede dividir en dos partes, la primera es ahondar en la investigación y entendimiento de un area tan extensa como las metaheurísticas, y en concreto los algoritmos evolutivos. La optimización de soluciones a problemas complejos mediante técnicas que tratan incluir conocimiento específico del dominio conjuntamente con la generación iterativa de otras soluciones parciales es un campo de estudio con un gran número de aplicaciones prácticas. Aunque el teorema de "No Free Lunch" nos advierte de que no existe una única técnica que sea la mejor para todos los problemas y desde todas las perspectivas [22], la experiencia empírica ha demostrado que los algoritmos evolutivos son una herramienta que se puede aplicar a la gran mayoría de problemas de optimización [18].

Es precisamente esa versatilidad la que ha propiciado su uso durante el desarrollo del segundo objetivo principal de este proyecto: la creación de un bot para un videojuegos de cartas. Este tipo de videojuegos cuentan con un inmenso número de variables, pues no solo se deben tener en cuenta las cartas existentes en las manos de cada jugador y en la pila de cartas, sino también las mecánicas[1] intrínsecas del juego, como la vida, los recursos de compra, la posibilidad de recuperar cartas usadas, o el uso de otro tipo de habilidades especiales. Todo esto hace que la creación de un bot que juegue de forma competitiva sea un reto interesante, y que el uso de técnicas de optimización evolutiva sea una herramienta adecuada para conseguirlo.

[1] Gran parte de lo que define un juego es sus mecánicas, sus reglas, lo que los jugadores deben llevar a cabo para ganar. La labor del creador de videojuegos es conseguir un conjunto de reglas que estén equilibradas y permitan disfrutar a los jugadores [21].

2.2. Objetivos específicos

- OG1: Analizar en profundidad las mecánicas del juego "Tales of Tribute", el entorno de desarrollo "Scripts of Tribute" y adquirir las competencias necesarias en el lenguaje de programación C#.
- OG2: Diseñar e implementar un agente inteligente en C# para "Scripts of Tribute", cuya toma de decisiones se base en una evaluación heurística del estado del juego mediante una función de fitness ponderada, incorporando conocimiento experto del dominio.
- OG3: Desarrollar un marco de optimización en Python para ajustar los pesos de la función de fitness del agente, implementando y comparando dos estrategias principales: algoritmos coevolutivos y entrenamiento supervisado contra agentes de referencia de "Scripts of Tribute".
- Desarrollar un conjunto de herramientas para la visualización y análisis de los datos generados durante el entrenamiento.
- OG5: Evaluar cuantitativamente el rendimiento del agente entrenado mediante las diferentes estrategias, utilizando métricas relevantes como la tasa de victorias contra distintos oponentes.
- OG6: Analizar comparativamente la efectividad y eficiencia de las estrategias de optimización implementadas (coevolución vs. entrenamiento contra referentes), discutiendo sus ventajas y desventajas en el contexto específico de "Scripts of Tribute".
- OG7: Investigar y contextualizar el enfoque desarrollado frente a otras técnicas predominantes en competiciones similares de IA en juegos, como Monte Carlo Tree Search (MCTS), analizando las razones de su éxito en ediciones anteriores.

Capítulo 3

Antecedentes y estado del arte

3.1. Inteligencia artificial en videojuegos comerciales

En esta sección se realiza un breve repaso a las técnicas de inteligencia artificial más utilizadas a lo largo de la historia de los videojuegos, así como algunos ejemplos del estado del arte en el ámbito de la industria comercial del videojuego. Los usos de estos algoritmos no solo abarcan su función más habitual de enemigo o aliado, sino que también abarcan otros aspectos como la búsqueda de caminos o la deformación realista de las mallas [2] de los personajes animados.

3.1.1. Funciones hash

Una de las técnicas más sencillas y antiguas para controlar partes específicas de un videojuego son las funciones hash. Este tipo de funciones simplemente reciben un conjunto de parámetros como entrada y devuelven un valor o acción a realizar. Dada su simplicidad, generan comportamientos predecibles, pero requieren de muy pocos recursos para su procesamiento y son fáciles de implementar. Un ejemplo clásico de IA basada en funciones hash es el en juego *Space Invaders* [20] donde los invasores estaban controlados con funciones hash.

3.1.2. Máquinas de estados finitos

Las máquinas de estados finitos son un modelo de computación conceptual que describe un sistema que solo puede encontrarse en un estado a la vez, y que puede cambiar a uno de sus otros estados como respuesta a ciertos eventos. Este mecanismo se ha utilizado y sigue utilizándose para un gran número de aplicaciones dentro de los videojuegos. Por ejemplo, en el juego *Pacman*, la IA de los

[2] Una "mesh", o malla en español, es una estructura tridimensional formada por una colección de vértices, aristas y caras que definen la forma de un objeto 3D dentro de un videojuego [3].

fantasmas se rije por el estado en el que se encuentran, como çazando.º "siendo cazados" [12]. Otro uso está en la gestión de las animaciones de los personajes, donde cada estado corresponde a una posición o movimiento específico de la malla, que al mezclarse mediante interpolaciones generan animaciones. Al igual que las funciones hash, requieren de pocos recursos para su procesamiento, pero su naturaleza determinista y la posibilidad de entrar en bucles sin salida si no están bien diseñadas, limitan su uso en los escenarios más exigentes.

3.1.3. Árboles de comportamiento

Como evolución a las máquinas de estados finitos, surgieron los árboles de comportamiento, permitiendo un mayor grado de complejidad y flexibilidad en la toma de decisiones de los personajes. Los árboles de comportamiento son grafos dirijidos acíclicos, con nodos hoja que representan acciones y varios tipos de control de flujo que determinan el orden de ejecución de las hojas [4]. Una de sus características más destacadas es su modularidad, lo que permite reutilizar y combinar comportamientos fácilmente. Un fantastico ejemplo de su correcto uso se expuso en la charla de la "Game Developers Conference" (GDC) de 2005, sobre la IA de *Halo 2*, donde se explica que su capacidad de reutilización de árboles y el ser más sencillos de entender para los diseñadores del juego, hizo que los programadores optaran por su implementación en lugar de las máquinas de estados finitos [10].

3.1.4. Planificación de acciones orientada a objetivos

En contraste con los árboles de comportamiento, que generan una jerarquía de caminos, la planificación de acciones orientada a objetivos (GOAP, por sus siglas en inglés) es más parecida a la planificación de rutas, donde se busca el camino más óptimo para alcanzar un objetivo en función del estado del mundo. Cada uno de los pasos que llevan al objetivo tiene una serie de precondiciones que deben cumplirse, a los cuales se les puede asignar un coste. Es tal su similitud con la planificación de rutas, que este tipo de IA suelen utilizar internamente algoritmos de búsqueda como A*[3] para encontrar el camino más óptimo. Una de las mejores implementaciones de este tipo de IA se encuentra en el título de 2005 *F.E.A.R.*, donde los enemigos y aliados utilizan GOAP para "planificar" su comportamiento una manera mucho más proactiva que sus predecesores [11].

3.1.5. Aprendizaje por refuerzo y redes neuronales

El aprendizaje por refuerzo (RL, por sus siglas en inglés) es un subcampo del aprendizaje automático que trata de mejorar el comportamiento de un agente a

[3] El algoritmo A* también se utiliza para la búsqueda de caminos tradicional dentro de los videojuegos, es decir, para saber la ruta que debe tomar un objeto por el mapa para llegar a un destino.

través del feedback que recibe de su entorno. Utilizando un sistema de recompensas y penalizaciones, el agente aprende a tomar decisiones que maximicen su recompensa total a lo largo del tiempo. Aunque el RL empezó con técnicas tabulares, los algoritmos recientes incorporan redes neuronales en diferentes partes del proceso [8]. Quizás el ejemplo que más se suele ver en la literatura de videojuegos es el uso de RL para controlar vehículos autónomos. Una implementación de esta tecnología en un videojuego comercial está en el título *Star Wars: Outlaws*, donde las motos controladas por IA utilizan un sistema de RL para moverse por el mapa o perseguir al jugador [6]. Sus desarrolladores argumentan que les permitió tener un sistema de control que se adapta a las diferentes rutas del juego sin importar los cambios en el mapa que haya entre versiones.

Pero el uso de redes neuronales no se limita a la toma de decisiones. Su capacidad para realizar inferencias rápidas a partir de datos complejos las hace ideales para otras tareas, como la deformación de mallas. El objetivo en este caso es evitar el fenómeno conocido como el "valle inquietante" (*Uncanny Valley*), que se produce cuando una réplica de un ser humano es muy realista, pero no perfecta, generando una sensación de extrañeza en el espectador. Un personaje que sonríe, por ejemplo, pero cuyos músculos faciales alrededor de los ojos permanecen inmóviles, es un caso clásico de este efecto. Para intentar superar este problema, Epic Games optó por crear un sistema para Unreal Engine que utiliza modelos de redes neuronales que simulan el comportamiento de la musculatura y los tejidos blandos bajo la piel al extenderse y contraerse [5]. Un ejemplo de uso de esta reciente tecnología se puede encontrar en la demo técnica que CD Projekt Red mostró en el Unreal Fest de 2025 [2]. En ella, se mostraba como se modelaba internamente la musculatura de un caballo para que sus animaciones de movimiento fueran mucho más realistas.

3.2. El problema de la IA en los videojuegos comerciales

En la anterior sección se han hablado exclusivamente de videojuegos que han salido al mercado para el público general. Pero, salvo para funciones específicas como el movimiento de las motos en *Star Wars: Outlaws*, no se ha mencionado el uso de técnicas de inteligencia artificial "avanzadas" para el control de los personajes o contrincantes virtuales. Sin embargo, este tipo de IAs sí existen. Se han desarrollado con éxito bots capaces de jugar a alto nivel videojuegos como *StarCraft II* [19], *Dota 2* [14] o *Rocket League* [13], entre otros. Todos estos artículos científicos, aclamados por la comunidad debido a su complejidad, tienen algo en común: se han creado una vez el juego ya había sido lanzado y era estable.

A menudo, los videojuegos con componentes multijugador de éxito, acaban siendo actualizados y mejorados durante años. Este clima de constance cambio, el cual se ve acentuado aun más durante la etapa inicial de su desarrollo, hace

que el uso de técnicas como el aprendizaje por refuerzo, los algoritmo evolutivos o las redes neuronales sean especialmente difíciles de implementar. El ejemplo de *Star Wars: Outlaws* es un caso muy reciente, con tecnología especialmente desarrollada para su motor gráfico y con un equipo detrás de cientos de personas con capacidad de delegar personal en enfoques más vanguardistas. Y aun así, sólo usaron el aprendizaje por refuerzo para un aspecto muy específico del videojuego.

Además, se podría considerar un problema del tipo "el huevo o la gallina", se necesita que el juego esté prácticamente listo para entrenar al bot, lo que permite crear un bot específico para esa versión, pero si el juego cambia, incluso con pequeños ajustes de balanceo de poder, entonces el rendimiento del bot puede verse afectado. Por eso todos esos artículos científicos se centran en juegos ya terminados o en una versión específica del videojuego. En aquellos casos en los se ha conseguido crear un bot que maneje todos los aspectos de un videojuego (jugador virtual), como lo fue para AlphaStar [19] o OpenAI Five [14], se han necesitado una gran cantidad de datos que solo se podrían haber obtenido gracias a la comunidad de jugadores de sus respectivos videojuegos y no antes de su lanzamiento.

Sin embargo, esto no significa que no se puedan crear este tipo de bots para videojuegos que están en desarrollo. Simplemente se deben utilizar para funciones específicas dentro de estos o en el caso de videojuegos que no requieran de un escenario tan complejo como los mencionados anteriormente [1]. Un ejemplo de esto sería el caso de Sophy, la IA de conducción de *Gran Turismo 7* [23], que fue entrenada durante el desarrollo del videojuego para manejar los vehículos del videojuego de carreras.

A día de hoy es posible que esa sea la forma más adecuada de utilizar este tipo de técnicas: o bien para pequeñas características del juego, o bien para entornos más reducidos, como el caso de los videojuegos de cartas. En ese tipo de videojuegos, aunque el número de variables es alto, la cantidad de acciones posibles en cada turno es limitada, lo que permite entrenar a los bots de forma más eficiente. En las siguientes secciones se revisarán algunas de las investigaciones científicas que intentan aplicar este mismo enfoque a sus implementaciones.

3.3. Metaheurísticas en la investigación científica sobre videojuegos

Una de esas aplicaciones específicas es la generación procedural de contenido, la cual trata de unir y mezclar diferentes bloques de construcción prehechos para crear nuevos elementos dentro del videojuego. Por ejemplo, se podrían utilizar varias texturas de manchas, ruido, metales y óxido para crear una única textura que muestre un material deteriorado por el tiempo, pero que tenga un aspecto único cada vez que se use. Otro ejemplo sería el de colocar diferentes plantas, rocas y árboles sobre un terreno para crear un bosque. Ya en 2011, Togelius et al. [17] publicaron una review sobre los diferentes enfoques que se estaban

utilizando para la generación procedural de contenido en videojuegos, y cómo las metaheurísticas estaban siendo utilizadas para este fin. Un ejemplo que revisaron fue el sistema Ludi, que codificaba las reglas del juego como árboles de expresión y usaba programación genética para generar conjuntos de reglas balanceadas para el videojuego. En su artículo, los autores dicen los sistemas revisados, los cuales usan enfoques basados en la simulación de procesos naturales, deben mejorar la consistencia, la rapidez y la personalización para el usuario si se quisieran utilizar estas técnicas en videojuegos comerciales.

Otro ejemplo de uso de metaheurísticas en la generación procedural de contenido es el trabajo de Geijtenbeek et al. [7], donde se presenta un sistema de locomoción para criaturas bípedas que utiliza un algoritmo evolutivo para optimizar los pesos de las articulaciones necesarios para que la criatura se mueva adecuadamente. En su artículo, los autores muestran como los controladores eran capaces de adaptarse a diferentes terrenos con desnivel e incluso perturbaciones externas.

Más recientemente, en 2021, Snell et al. [16] presentaron un enfoque evolutivo para el balanceo de videojuegos de estrategia en tiempo real. En su trabajo, utilizaron diferentes variables dentro del videojuego para conseguir variaciones en la dificultad de un nivel sin desviarse demasiado de la experiencia original del videojuego. Concluyeron que su enfoque era capaz de generar niveles similares, pero que se sentían diferentes, lo que permitía al jugador disfrutar de una experiencia más variada sin ser injusta.

3.4. Algoritmos evolutivos en juegos de estrategia por turnos

3.4.1. Optimización de agentes de Hearthstone mediante un algoritmo evolutivo

3.4.2. Desarrollo de un agente para el concurso de IA Tales of Tribute

TODO: Analizar la estrategia de MCTS de los ganadores de la competición de IA de Tales of Tribute OG7.

Capítulo 4

Plataforma de trabajo

4.1. El videojuego: Tales of Tribute

4.2. El entorno de simulación: Scripts of Tribute

4.2.1. La competición IEEE-CoG

4.2.2. Arquitectura del entorno

Capítulo 5

Planificación del proyecto

En este capítulo se detallan los aspectos relacionados con los recursos que se han utilizado para el desarrollo del proyecto, así como la metodología que se ha seguido y la distribución temporal utilizada en cada tarea.

5.1. Recursos utilizados

5.1.1. Recursos software

El proyecto se puede dividir en tres partes: el desarrollo del bot en C#, el ajuste de los pesos en Python y la redacción del informe en LaTeX. Afortunadamente, existe una herramienta de código abierto y en constante mejora en la que se pueden realizar todas estas tareas: Visual Studio Code (VSCode). Este entorno de desarrollo es la navaja suiza que ha estado presente en todo el proceso, permitiendo la edición de código, la gestión de versiones, la redacción del informe en incluso la conexión con interfáz gráfica entre máquinas por SSH. Todo ello gracias a su robusto sistema de extensiones (y la gran comunidad que lo apoya), que permite añadir prácticamente cualquier funcionalidad que se necesite. En el caso de la gestión de versiones, utilizando la integración con Git de VSCode, en todo momento se ha mantenido un repositorio actualizado al día en GitHub, lo que ha permitido la transferencia del progreso entre las dos máquinas utilizadas durante el desarrollo.

Por último, destacar el uso de Zotero como software de gestión de referencias bibliográficas, así como del propio Scripts of Tribute, la base sobre la que se ha erguido el resto del código.

5.1.2. Recursos hardware

Se han contado con dos ordenadores principales, el primero es el ordenador personal del autor, mientras que el segundo fue proporcionado por el director del

proyecto para realizar los experimentos. Estas son sus características:

- **Ordenador personal (Windows 11):**
  - Procesador: AMD Ryzen 7 5800X
  - Memoria RAM: 32 GB DDR5
  - Tarjeta gráfica: NVIDIA GeForce RTX 3070
- **Ordenador de experimentación (Ubuntu 22.04):**
  - Procesador: Intel i9-12900KF
  - Memoria RAM: 128 GB DDR4
  - Tarjeta gráfica: NVIDIA GeForce RTX 4060

5.2. Metodología de desarrollo

TODO: No tiene mucho sentido que esta sección de metodología este fuera de la parte 2, que literalmente se llama Metodología. Revisarlo.

Para el desarrollo de este Trabajo de Fin de Máster se ha seguido una metodología ágil a base de "sprints" regulares. Desde el 12 de noviembre de 2024, se mantuvieron reuniones de forma constante con el director del proyecto, Pablo García Sánchez, aproximadamente cada 3 semanas. En cada una, primero se procedía a repasar el progreso desde la última reunión, a menudo mostrando en tiempo real las nuevas características o los resultados obtenidos. Sobre esta base, el director proponía mejoras o correcciones a realizar para la próxima ocasión. Luego, entre ambos se decidían los siguientes pasos a seguir, mediante una comunicación abierta, lo que permitía ajustar el rumbo del proyecto según las necesidades y los resultados obtenidos.

Aunque las primeras fases del proyecto se realizaron enteramente en el PC personal del autor, en cuanto se necesitó ejecutar el script de ajuste de pesos (o "entrenador"), se comenzó a utilizar el ordenador proporcionado por el director para ese propósito. Esto permitió realizar experimentos de forma continuada y sencilla, mientras se seguía trabajando en el código del bot en el PC personal. Así se pudo mantener un flujo de trabajo fluido y eficiente, generando código funcional de forma regular y obteniendo resultados cada vez más desarrollados constantemente.

5.3. Distribución temporal y cronograma

Para visualizar la distribución temporal de las diferentes tareas realizadas durante el desarrollo de este TFM, se ha elaborado el siguiente diagrama de Gantt (Figura 5.1).

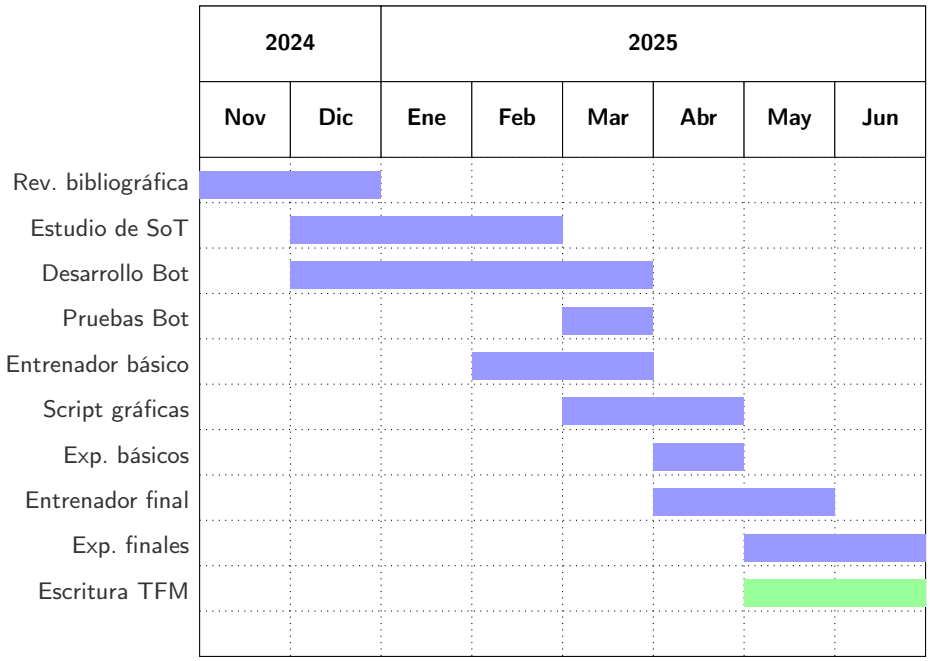

Figura 5.1: Distribución temporal de las tareas del proyecto

Como se puede observar en el diagrama, la mayoría de tareas se han desarrollado de forma secuencial, aunque algunas se han solapado parcialmente cuando ha sido posible trabajar en paralelo. La escritura del TFM se concentró en el último mes y medio del proyecto, una vez que se contaba con resultados experimentales suficientes para documentar adecuadamente el trabajo realizado. Este es el contenido de cada una de las tareas:

- **Revisión bibliográfica:** se empezó revisando los trabajos ya descritos en la sección 3.4, además de aprender a jugar a Tales of Tribute y definir el alcance del proyecto.
- **Estudio de Scripts of Tribute:** en esta etapa se estudió el código fuente de Scripts of Tribute, y se empezó a trabajar con C# como lenguaje de programación, ya que este nunca había sido utilizado por el autor antes.
- **Desarrollo del bot:** se diseñó y desarrolló el bot a base de iteraciones de otros bots existentes, así como del trabajo de Pablo et al. TODO: cita aquí y otras referencias.
- **Pruebas del bot:** primeras pruebas para comprobar que el funcionamiento del bot era correcto.
- **Entrenador básico:** en un principio, el entrenador solo tenía un modo de funcionamiento básico y estaba peor organizado, pero el ágil desarrollo de esta versión permitió alcanzar resultados rápidamente a modo de prototipo.

- **Script de gráficas:** se creó un script para generar gráficas que visualizan los resultados de los experimentos realizados, así como la función del entrenador que crea los datasets necesarios para su funcionamiento.
- **Experimentos básicos:** experimentos iniciales para evaluar si el rendimiento del bot mejoraba con el ajuste de pesos.
- **Entrenador final:** desarrollo de la versión final del entrenador, incluyendo los diferentes modos de entrenamiento que se describirán más adelante en la sección 7.5.
- **Experimentos finales:** experimentos pensados para la evaluación del rendimiento de los distintos modos de entrenamiento del entrenador final.
- **Escritura del TFM:** finalmente se redactó este informe, documentando todo el proceso realizado durante el desarrollo del proyecto.

Parte II

Metodología

Capítulo 6

Diseño del agente autónomo

6.1. Arquitectura de la toma de decisiones

6.2. Definición de los pesos

6.3. Lógicas específicas y heurísticas

Capítulo 7

Sistema de entrenamiento evolutivo

Parte III

Experimentación y conclusiones

Capítulo 8

Diseño experimental

TODO: Probar diferentes combinaciones en el modo hibrido. TODO: Hablar sobre los individuos generalistas (todos los pesos parecidos) y especialistas (0-1 en los pesos) en los mejores pesos. TODO: Ver la diferencia entre usar el hall of fame y no usarlo ya que no debería añadir info de una train run a otra. Mirar en la literatura que es exactamente hall of fame. TODO: Para comparar los modos de entrenamiento, intentar que todos los algoritmos se ejecuten durante la misma cantidad de tiempo (aunque tengan tamaños de poblaciones diferentes). TODO: Lanzar sin el hall of fame para evitar las diferencias entre runs.

8.1. Configuración de los experimentos

8.2. Métricas de evaluación

Capítulo 9

Resultados y discusión

9.1. A nivel de población

9.1.1. Evolución del fitness

9.1.2. Pesos medios

9.1.3. Pesos a lo largo de las generaciones

9.1.4. Variación de los pesos finales

9.2. A nivel de líderes

9.2.1. Evolución del fitness

9.2.2. Pesos medios

9.2.3. Pesos a lo largo de las generaciones

9.2.4. Variación de los pesos finales

Capítulo 10

Conclusiones

10.1. Objetivos alcanzados

10.2. Líneas de trabajo futuro

Bibliografía

[1] AI and Games. Why is It Difficult to Make Good AI for Games? | AI 101, January 2024. URL: https://www.youtube.com/watch?v=qCkqpRnk1oU.

[2] CD Projekt Red. The Witcher 4 Unreal Engine 5 Tech Demo 4K | State of Unreal | Unreal Fest Orlando, June 2025. URL: https://www.youtube.com/watch?v=aorRfK478RE.

[3] Universidad Europea. ¿Qué es mesh en el desarrollo de videojuegos? | Blog CC, April 2025. Section: Blog. URL: https://creativecampus.universidadeuropea.com/blog/meshes/.

[4] Epic Games. Behavior Tree in Unreal Engine - Overview | Unreal Engine 5.6 Documentation | Epic Developer Community, 2025. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/behavior-tree-in-unreal-engine---overview.

[5] Epic Games. ML Deformer Framework in Unreal Engine | Unreal Engine 5.6 Documentation | Epic Developer Community, 2025. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/ml-deformer-framework-in-unreal-engine.

[6] Colin Gaudreau and Andreas Lasses. Game AI Summit: No Brakes! Machine Learning Vehicles in 'Star Wars Outlaws', 2025. URL: https://gdcvault.com/play/1035556/Game-AI-Summit-No-Brakes.

[7] Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Trans. Graph.*, 32(6):206:1–206:11, November 2013. doi:10.1145/2508363.2508399.

[8] Majid Ghasemi, Amir Hossein, and Dariush Ebrahimi. Comprehensive Survey of Reinforcement Learning: From Algorithms to Practical Challenges, February 2025. URL: https://arxiv.org/html/2411.18892v2#bib.bib84.

[9] Juan Julián Merelo Guervós. JJ/plantilla-TFG-ETSIIT, September 2024. original-date: 2019-03-19T11:58:46Z. URL: https://github.com/JJ/plantilla-TFG-ETSIIT.

[10] Damian Isla. Managing Complexity in the Halo 2 AI System, 2005. URL: https://www.gdcvault.com/play/1020270/Managing-Complexity-in-the-Halo.

[11] Orkin Jeff. GDC Vault - Three States and a Plan: The AI of F.E.A.R., 2006. URL: https://gdcvault.com/play/1013282/Three-States-and-a-Plan.

[12] Mike. Game Programming Concepts–Finite State Machines, June 2016. URL: https://gamefromscratch.com/game-programming-concepts-finite-state-machines/.

[13] Vasileios Moschopoulos, Pantelis Kyriakidis, Aristotelis Lazaridis, and Ioannis Vlahavas. Lucy-SKG: Learning to Play Rocket League Efficiently Using Deep Reinforcement Learning, May 2023. arXiv:2305.15801 [cs]. URL: http://arxiv.org/abs/2305.15801, doi:10.48550/arXiv.2305.15801.

[14] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning, December 2019. arXiv:1912.06680 [cs, stat]. URL: http://arxiv.org/abs/1912.06680, doi:10.48550/arXiv.1912.06680.

[15] Raymond Redheffer. A Machine for Playing the Game Nim. *The American Mathematical Monthly*, 55(6):343–349, June 1948. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00029890.1948.11999249. doi:10.1080/00029890.1948.11999249.

[16] Jacob Snell, Martin Masek, and Chiou Lam. An evolutionary approach to balancing and disrupting real-time strategy games. *MODSIM2021, 24th International Congress on Modelling and Simulation*, January 2021. URL: https://ro.ecu.edu.au/ecuworkspost2013/11752, doi:10.36334/modsim.2021.M8.snell.

[17] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games*,

3(3):172–186, September 2011. URL: https://ieeexplore.ieee.org/document/5756645, doi:10.1109/TCIAIG.2011.2148116.

[18] Jose Torres-Jiménez and Juan Pavón. Applications of metaheuristics in real-life problems. *Progress in Artificial Intelligence*, 2(4):175–176, July 2014. doi:10.1007/s13748-014-0051-8.

[19] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, November 2019. Publisher: Nature Publishing Group. URL: https://www.nature.com/articles/s41586-019-1724-z, doi:10.1038/s41586-019-1724-z.

[20] Wikipedia. Artificial intelligence in video games, May 2025. Page Version ID: 1292216335. URL: https://en.wikipedia.org/w/index.php?title=Artificial_intelligence_in_video_games&oldid=1292216335.

[21] Wikipedia. Diseño de juegos, May 2025. Page Version ID: 167396849. URL: https://es.wikipedia.org/w/index.php?title=Dise%C3%B1o_de_juegos&oldid=167396849.

[22] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997. URL: https://ieeexplore.ieee.org/document/585893, doi:10.1109/4235.585893.

[23] Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, February 2022. Publisher: Nature Publishing Group. URL: https://www.nature.com/articles/s41586-021-04357-7, doi:10.1038/s41586-021-04357-7.