

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA



ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

**Simulación de estrategias de baloncesto a
través de Aprendizaje Reforzado en un
entorno de juego de mesa**

Autor/a:

Leticia Martínez Limeres

Tutores:

David Losada Carril
Manuel González López

**Máster Universitario en Tecnologías de Análisis de
Datos Masivos: Big Data**

Julio 2025

Trabajo de Fin de Máster presentado en la Escuela Técnica Superior de Ingeniería de la Universidad de Santiago de Compostela para la obtención del Máster Universitario en Tecnologías de Análisis de Datos Masivos: Big Data

Agradecimientos

A todas las personas que forman parte de mi círculo cercano —familia, amigos y quienes me han acompañado en distintos momentos de este camino—, gracias por estar ahí. Por vuestro cariño incondicional, por las risas compartidas, por el apoyo en los momentos difíciles y por creer en mí incluso cuando yo misma dudaba.

Quiero dedicar una mención especial a mi abuelo, que aunque ya no está con nosotros, sigue muy presente. Su recuerdo, su ejemplo y su fuerza me han acompañado durante todo este proceso.

Y, por supuesto, gracias a mi tutor, por valorar mi trabajo, motivarme y guiarme con sus orientaciones a lo largo de este proyecto.

Índice general

Índice de abreviaturas	VI
Resumen	VII
1. Introducción	1
1.1. Estructura del documento	1
1.2. Motivación	2
1.3. Objetivos	3
2. Contexto y requisitos	4
2.1. Marco teórico y contextual	4
2.1.1. Introducción al Aprendizaje por Refuerzo	4
2.1.2. El Caso del Baloncesto	5
2.1.3. Soluciones Comerciales Existentes	6
2.2. Especificación de requisitos	6
2.2.1. Requisitos funcionales	6
2.2.2. Requisitos no funcionales	7
2.2.3. Requisitos de información	7
3. Diseño e implementación del entorno de simulación	8
3.1. Representación del entorno de juego	8
3.2. Arquitectura técnica y herramientas	10
4. Estrategia de entrenamiento	13
4.1. Versión 1: Entorno básico y controlado	13
4.2. Versión 2: Interacción y lógica de tiro	15
4.3. Versión 3: Defensa e intercepciones	16
4.4. Versión 4: Asignación de roles y datos de NBA	18
5. Evaluación experimental	22
5.1. Metodología de evaluación	22
5.2. Resultados por versión	22
6. Conclusiones y posibles ampliaciones	30
A. Entregables del proyecto	31
Bibliografía	32

Índice de figuras

2.1.	Esquema del funcionamiento básico del aprendizaje por refuerzo [5]	4
3.1.	Representación visual del tablero de baloncesto en Pygame	9
3.2.	Distribución inicial de los jugadores en el tablero al inicio del episodio	10
4.1.	Estado final del tablero tras un lanzamiento en la versión 1	14
4.2.	Estado final del tablero tras un lanzamiento en la versión 2	16
4.3.	Estado final del tablero tras un lanzamiento en la versión 3	17
4.4.	Dispersión de los tiros realizados por todos los jugadores seleccionados	19
4.5.	Dispersión individual de los tiros realizados por cada jugador	19
4.6.	Mapeo de los tiros realizados por cada jugador	20
4.7.	Evaluación del valor óptimo de k en el modelo KNN mediante validación cruzada para distintos jugadores	20
4.8.	Comparación entre el mapeo original de tiros y el resultado tras aplicar KNN para estimar probabilidades en celdas sin datos	21
5.1.	Frecuencia total de acciones por versión	23
5.2.	Distribución de lanzamiento según acciones del episodio	23
5.3.	Distribución del resultado de los lanzamientos por versión	24
5.4.	Distribución espacial de eventos durante los episodios en la versión 1	24
5.5.	Distribución espacial de eventos durante los episodios en la versión 2	25
5.6.	Distribución espacial de eventos durante los episodios en la versión 3	27
5.7.	Distribución espacial de eventos durante los episodios en la versión 4	27
5.8.	Porcentaje de aciertos y fallos en los tiros de cada jugador	28
5.9.	Representación de un episodio completo de la versión 1	29
5.10.	Representación de fragmentos de un episodio de la versión 4	29

Índice de tablas

5.1.	Resumen de métricas por iteraciones en la versión 1	25
5.2.	Resumen de métricas por iteraciones en la versión 2	26
5.3.	Resumen de métricas por iteraciones en la versión 3	27
5.4.	Resumen de métricas por iteraciones en la versión 4	28

Índice de abreviaturas

API	Application Programming Interface
CLI	Command Line Interface
DQN	Deep Q-Network
FGA	Field Goald Attempt
FIBA	Federación Internacional de Baloncesto
IA	Inteligencia Artificial
KNN	K-Nearest Neighbors
MSE	Mean Squarred Error
NBA	National Basketball Association
PPO	Proximal Policy Optimization
RL	Reinforcement Learning

Resumen

El Aprendizaje Reforzado es una técnica avanzada de inteligencia artificial que permite a un agente aprender a tomar decisiones óptimas mediante la interacción con su entorno y la retroalimentación recibida en forma de recompensas. Este Trabajo Fin de Máster explora su aplicación en el ámbito deportivo, concretamente en el baloncesto, mediante el desarrollo de un entorno simulado implementado con una arquitectura modular. Para ello, se ha utilizado la librería `Gymnasium` para la definición del entorno, `Pygame` para la visualización y `MLflow` para la gestión y seguimiento de los experimentos, lo que ha facilitado el análisis del agente.

El entorno, estructurado sobre un tablero dividido en celdas que representa el campo de baloncesto, permite a los jugadores desplazarse, pasar el balón o lanzar a canasta según unas reglas definidas. La simulación se ha desarrollado a través de cuatro versiones progresivas, cada una introduciendo un mayor nivel de complejidad: desde un agente aislado sin oposición, pasando por la incorporación de la acción de pase y probabilidades de acierto, hasta llegar a una versión avanzada con defensa activa, roles funcionales diferenciados y datos reales extraídos de la NBA.

A lo largo del entrenamiento, el agente ha mostrado una evolución clara: desde acciones impulsivas como lanzamientos inmediatos, hasta estrategias más elaboradas en las versiones avanzadas, priorizando la conservación de la posesión, la búsqueda de espacios y la selección del jugador más adecuado para lanzar. Para evaluar esta progresión se han definido métricas específicas como la recompensa media por episodio, la duración de las jugadas, el porcentaje de tiros tempranos y mapas de calor que muestran la distribución espacial de los eventos clave. La integración de datos reales de la NBA ha permitido analizar la capacidad del agente para adaptar sus decisiones a perfiles funcionales como bases, aleros o pívots, evidenciando una comprensión parcial de la lógica táctica del baloncesto.

Si bien el sistema presenta ciertas limitaciones, como el mapeo de datos reales al tablero o una penalización defensiva excesiva, los resultados obtenidos indican la viabilidad del aprendizaje reforzado para simular comportamientos tácticos deportivos. Además, este trabajo establece una base sólida para futuras ampliaciones, entre las que destacan el entrenamiento multiagente con coordinación descentralizada, el aprendizaje defensivo autónomo o la simulación de secuencias completas de juego que incluyan reinicios, faltas y cambios de posesión. Todo ello acerca el modelo a una representación más funcional y realista del baloncesto.

Palabras clave: aprendizaje por refuerzo, agentes inteligentes, baloncesto, simulación, toma de decisiones, inteligencia artificial.

Capítulo 1

Introducción

El baloncesto es un deporte dinámico que exige la aplicación de tácticas y estrategias precisas para alcanzar el máximo rendimiento durante el desarrollo del juego. La simulación de estas estrategias en entornos controlados, como un juego de mesa, constituye una alternativa eficaz para analizar, probar y perfeccionar diferentes jugadas sin necesidad de llevarlas a cabo en situaciones reales. Este enfoque no solo reduce los costes y riesgos inherentes a los entrenamientos tradicionales, sino que también permite un análisis detallado de las decisiones y movimientos implicados en el juego.

En este contexto, el aprendizaje por refuerzo [1, 2], una técnica avanzada en el ámbito de la inteligencia artificial, se presenta como una herramienta idónea para entrenar agentes capaces de tomar decisiones y diseñar estrategias de forma autónoma en entornos dinámicos. Esta metodología permite desarrollar sistemas que aprenden a través de la interacción continua con el entorno, ajustando su comportamiento a partir de la retroalimentación recibida tras cada acción.

En el marco de este proyecto, se propone el desarrollo de una simulación de baloncesto en un entorno de juego de mesa, en el que cada jugador ocupa una posición fija dentro del tablero. A medida que el proyecto avance, se incrementará progresivamente la complejidad del sistema, incorporando diversas restricciones derivadas del reglamento oficial del baloncesto FIBA [3], como el número de jugadores, la asignación de roles específicos o diversos factores que influyen directamente en el desarrollo del partido. De este modo, se aspira a construir una simulación lo más realista y funcional posible, que permita analizar y perfeccionar tácticas de juego mediante el aprovechamiento de las capacidades que ofrece la inteligencia artificial.

1.1. Estructura del documento

El presente documento se estructura en diversos capítulos, cada uno de los cuales aborda un aspecto específico del desarrollo del trabajo, desde los fundamentos teóricos hasta los resultados obtenidos.

En el [Capítulo 1](#) se presenta una visión general del proyecto, junto con la motivación que lo impulsa y los objetivos que se desea alcanzar.

El [Capítulo 2](#) presenta el contexto teórico y técnico en el que se enmarca el proyecto. En este capítulo se definen los requisitos del sistema y se analizan soluciones existentes tanto en el ámbito académico como en el comercial, lo que conforma el estado del arte que sirve como referencia.

En el [Capítulo 3](#) se describe en detalle el diseño e implementación del entorno de simulación. Se incluyen aspectos clave como la representación visual del tablero, la estructura interna del entorno, así como las herramientas y tecnologías utilizadas.

El [Capítulo 4](#) está dedicado a la estrategia de entrenamiento del agente. Se explican las distintas versiones o fases del entorno, cada una con un nivel creciente de complejidad, y se detalla cómo se ha guiado el aprendizaje del agente a través de la incorporación progresiva de nuevas reglas y dinámicas de juego.

En el [Capítulo 5](#) se presentan los resultados experimentales obtenidos tras entrenar al agente en cada una de las versiones. Se incluyen tanto métricas como visualizaciones que permiten analizar el comportamiento aprendido.

Por último, el [Capítulo 6](#) recoge las conclusiones y principales aportaciones del trabajo. Asimismo, se proponen posibles líneas de trabajo futuro y vías de mejora.

Adicionalmente, en el [Apéndice A](#) se incluye un enlace al repositorio que contiene todos los códigos entregables del proyecto.

1.2. Motivación

En la actualidad, la aplicación de la inteligencia artificial en el ámbito deportivo se encuentra en plena expansión, utilizándose en tareas como el análisis de datos, la optimización de entrenamientos o el diseño de estrategias. No obstante, muchas de estas soluciones requieren infraestructuras complejas o datos obtenidos en contextos reales, lo que limita su accesibilidad y dificulta su implementación a pequeña escala.

Este proyecto surge como respuesta a la necesidad de disponer de entornos accesibles y controlados que permitan simular y analizar estrategias de baloncesto de manera eficiente. La elección de un juego de mesa como entorno de simulación representa una alternativa sencilla, flexible y de bajo coste para experimentar con diferentes tácticas y modelos de toma de decisiones.

La propuesta se fundamenta en un interés creciente por explorar el potencial de la inteligencia artificial en distintos ámbitos, incluido el deportivo, con el objetivo de transformar los procesos de toma de decisiones estratégicas y optimizar el rendimiento. La elección del aprendizaje por refuerzo como técnica principal permite evidenciar la capacidad de la IA para generar comportamientos adaptativos, entrenando agentes capaces de mejorar su desempeño en entornos dinámicos.

Además, este Trabajo Fin de Máster ha sido tutelado por la empresa **FDS, a DXC Technology Company**, que apuesta por impulsar proyectos que integren deporte y tecnología, fomentando así la innovación aplicada en este ámbito.

1.3. Objetivos

El objetivo principal de este trabajo es desarrollar una simulación de estrategias de baloncesto en un entorno de juego de mesa, en el que un agente inteligente, entrenado mediante técnicas de Aprendizaje Reforzado y apoyado en el uso de Big Data e Inteligencia Artificial, sea capaz de tomar decisiones óptimas que maximicen el rendimiento del equipo en situaciones dinámicas.

De manera más concreta, se plantean los siguientes objetivos específicos:

- Diseñar y entrenar un agente inteligente, capaz de aprender y adaptarse de forma autónoma mediante técnicas de Aprendizaje Reforzado, optimizando la toma de decisiones en un entorno dinámico y cambiante.
- Definir e implementar una estrategia de entrenamiento progresiva por fases, incrementando gradualmente la complejidad del entorno mediante nuevas reglas, restricciones y mecanismos de toma de decisiones.
- Desarrollar un entorno de prueba controlado que permita evaluar el impacto de las estrategias aplicadas mediante diversas métricas y la visualización de episodios reproducibles, facilitando así el análisis cualitativo del comportamiento aprendido.
- Procesar y analizar los datos generados durante la ejecución de las distintas tácticas y movimientos implementados en la simulación, con el objetivo de extraer información relevante que permita evaluar y perfeccionar el rendimiento del agente.
- Incorporar tecnologías avanzadas de registro, almacenamiento y procesamiento de datos, en caso de que el volumen generado lo requiera, garantizando la eficiencia computacional y la escalabilidad del sistema.

A través de este enfoque, el proyecto explora cómo el Big Data y la Inteligencia Artificial pueden contribuir a optimizar la toma de decisiones estratégicas en el ámbito deportivo, facilitando el desarrollo de modelos predictivos capaces de adaptarse a situaciones variables de manera autónoma.

Capítulo 2

Contexto y requisitos

2.1. Marco teórico y contextual

Esta sección expone los fundamentos teóricos y contextuales del proyecto, organizados en tres partes: una introducción al aprendizaje por refuerzo como referencia teórica del sistema, una revisión de trabajos previos relacionados con su aplicación en el baloncesto y un repaso de soluciones comerciales basadas en esta técnica.

2.1.1. Introducción al Aprendizaje por Refuerzo

El aprendizaje por refuerzo (Reinforcement Learning, RL) es una técnica avanzada de inteligencia artificial que permite entrenar agentes capaces de tomar decisiones de forma autónoma mediante la interacción continua con su entorno. A diferencia del aprendizaje supervisado, basado en datos etiquetados, en RL el agente aprende mediante prueba y error, ajustando su comportamiento según las recompensas o penalizaciones recibidas por cada acción. Este proceso imita el aprendizaje humano basado en la experiencia y las consecuencias de los actos [1, 4].

El modelo básico de RL se compone de dos elementos clave: el agente, que toma decisiones y aprende; y el entorno, donde se llevan a cabo dichas decisiones y se reciben sus consecuencias [2]. En cada ciclo de interacción, el agente percibe un estado (la situación actual del entorno), ejecuta una acción y recibe una recompensa como retroalimentación. Esta señal guía su aprendizaje para maximizar la recompensa acumulada a lo largo del tiempo.

El proceso completo constituye un episodio o ciclo de retroalimentación continua, donde las decisiones del agente se refinan progresivamente con el aprendizaje.

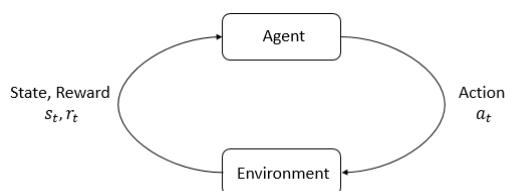


Figura 2.1: Esquema del funcionamiento básico del aprendizaje por refuerzo [5]

La estrategia que sigue el agente se define mediante una política, que puede ser determinista (elige siempre la misma acción dado un estado) o estocástica (asigna una probabilidad a cada acción posible), y que se mejora con la experiencia. En la práctica, es común emplear políticas predefinidas como `MlpPolicy`, basadas en redes neuronales multicapa para aproximar funciones de valor o de política y adecuada para entornos con espacios de observación vectoriales y relativamente estructurados.

El RL se implementa mediante distintos algoritmos. Entre los más utilizados destacan: Q-Learning, por su simplicidad y eficiencia en entornos discretos; DQN (Deep Q-Network) [6], que combina Q-Learning con redes neuronales profundas para entornos más complejos; y PPO (Proximal Policy Optimization) [7], un algoritmo basado en políticas conocido por su estabilidad y buen rendimiento en espacios continuos [5].

En los últimos años, el aprendizaje por refuerzo ha tenido una expansión significativa, con aplicaciones exitosas en videojuegos, simulaciones deportivas, sistemas autónomos y optimización industrial. Su capacidad de adaptación y toma de decisiones en entornos dinámicos lo ha convertido en una herramienta clave dentro de la inteligencia artificial.

Uno de sus hitos más representativos es AlphaGo, desarrollado por DeepMind, que logró derrotar a campeones mundiales del juego Go mediante entrenamiento basado en auto-juego [8]. Posteriormente, AlphaZero extendió este enfoque al ajedrez y al shogi, demostrando la eficacia del RL en una amplia variedad de entornos estructurados [9]. De forma similar, en juegos como el backgammon se ha empleado el aprendizaje por diferencia temporal para alcanzar niveles comparables a los de jugadores expertos [10].

2.1.2. El Caso del Baloncesto

La aplicación del aprendizaje por refuerzo en simulaciones deportivas ha abierto nuevas posibilidades para modelar estrategias complejas y mejorar la toma de decisiones. Un ejemplo es Fever Basketball, un entorno de simulación específicamente diseñado para entrenar agentes en situaciones de juego realistas. Este simulador permite evaluar algoritmos multiagente en distintos niveles de complejidad, incorporando recompensas distribuidas y ejecución asincrónica de acciones, lo que lo convierte en una plataforma ideal para probar estrategias avanzadas [11].

Más allá de las simulaciones, el RL se ha utilizado en el análisis y optimización del rendimiento de los jugadores en competiciones reales. En este contexto, Q-Ball es un modelo basado en RL que analiza los movimientos, acciones y rendimiento de los jugadores para generar puntuaciones de desempeño y recomendaciones tácticas. Este sistema ha sido validado con datos reales de partidos de la NBA, demostrando su capacidad para mejorar la toma de decisiones durante un partido [12].

Asimismo, ha mostrado su eficacia en el desarrollo de habilidades técnicas específicas dentro del ámbito deportivo. Un estudio reciente evaluó distintos tipos de retroalimentación en el entrenamiento de tiros libres, comparando el impacto del aprendizaje basado en errores con el refuerzo positivo (éxito/fallo). Los resultados

revelaron que el refuerzo positivo favorece la retención a largo plazo, mientras que la retroalimentación basada en errores resulta más efectiva durante la fase inicial. La combinación de ambos enfoques se presenta como una estrategia óptima para el aprendizaje deportivo eficiente [13].

2.1.3. Soluciones Comerciales Existentes

El aprendizaje por refuerzo no solo ha sido objeto de estudio en el ámbito académico, sino que también ha sido adoptado en diversas soluciones comerciales, destacando por su versatilidad.

- **IBM Watson:** incorpora técnicas de RL en su modelo de detección de intenciones, *IBM Watson Assistant*, cuya última versión mejora notablemente la velocidad, precisión y capacidad de categorización de las intenciones expresadas por los usuarios [14].
- **Microsoft Azure AI:** ofrece herramientas basadas en RL para optimizar procesos y estrategias industriales [15].
- **OpenAI:** desarrolló *OpenAI Five*, un sistema basado en RL que ha demostrado ser competitivo en videojuegos complejos, destacando su capacidad de toma de decisiones en entornos dinámicos [16].
- **Dispositivos wearables:** algunos dispositivos comerciales, como relojes inteligentes, emplean RL para personalizar recomendaciones de salud y ejercicio, aprendiendo los patrones de actividad, sueño y comportamiento del usuario.
- **Sistemas de recomendación de contenido:** plataformas como Netflix utilizan modelos de aprendizaje por refuerzo para personalizar la oferta de contenido según las preferencias y hábitos del usuario, mejorando así la experiencia y fidelización [17, 18].
- **Conducción autónoma:** la empresa Waymo combina aprendizaje por imitación con aprendizaje por refuerzo para optimizar la seguridad, eficiencia y adaptabilidad de sus vehículos autónomos [19].

2.2. Especificación de requisitos

El desarrollo del entorno de simulación de baloncesto requiere una adecuada identificación de los requisitos que debe cumplir el sistema. Para ello, se han clasificado en tres grupos principales descritos brevemente a continuación.

2.2.1. Requisitos funcionales

Los requisitos funcionales especifican las acciones del sistema y su comportamiento esperado.

- **[RF1]** El tablero debe representar visualmente un campo de baloncesto simplificado, dividido en una cuadrícula que facilite el modelado de posiciones y movimientos.
- **[RF2]** El sistema debe permitir simular acciones básicas como mover el balón, lanzar a canasta y alternar turnos entre jugadores.
- **[RF3]** La interfaz debe mostrar un marcador de puntuación actualizado dinámicamente durante la partida.
- **[RF4]** El entorno debe estar diseñado para que un agente inteligente pueda percibir el estado actual y ejecutar acciones en consecuencia.
- **[RF5]** Se debe almacenar el historial de jugadas y movimientos en formato estructurado para su análisis posterior y reutilización en el entrenamiento.

2.2.2. Requisitos no funcionales

Los requisitos no funcionales establecen condiciones de calidad, usabilidad e interoperabilidad que debe cumplir el sistema sin describir funcionalidades concretas.

- **[RNF1]** La aplicación debe desarrollarse en Python, utilizando las bibliotecas necesarias para la simulación y el entrenamiento.
- **[RNF2]** La representación visual debe ser clara, intuitiva y adecuada para interpretar el comportamiento del agente.
- **[RNF3]** El sistema debe soportar la ejecución automatizada de múltiples episodios para entrenamiento y evaluación.
- **[RNF4]** El sistema de entrenamiento debe completarse en un tiempo razonable (preferiblemente inferior a 10 minutos) para facilitar ciclos rápidos de experimentación y evaluación.

2.2.3. Requisitos de información

Estos requisitos definen cómo deben estructurarse, almacenarse y formatearse los datos del sistema.

- **[RI1]** El sistema debe registrar cada episodio simulado, incluyendo las acciones, recompensas, posiciones y métricas asociadas, en formato estructurado.
- **[RI2]** Los datos generados deben poder exportarse en formatos estándar como CSV o JSON para su análisis y uso en procesos de entrenamiento o evaluación.
- **[RI3]** El sistema debe garantizar la persistencia de los modelos entrenados mediante herramientas como **MLflow**.
- **[RI4]** Las entradas y salidas de cada episodio deben poder identificarse y reproducirse de forma única mediante identificadores.

Capítulo 3

Diseño e implementación del entorno de simulación

Este capítulo describe el diseño conceptual y la implementación práctica del entorno de simulación desarrollado para el proyecto. Se abordan aspectos clave como la representación del tablero de juego, la segmentación por zonas, las herramientas utilizadas para la visualización, así como la arquitectura modular que permite desacoplar el entrenamiento del agente respecto a la visualización. Todo ello constituye la base técnica sobre la que se construyen las distintas versiones del entorno.

3.1. Representación del entorno de juego

El entorno de simulación desarrollado tiene como objetivo representar, de forma simplificada pero funcional, una cancha de baloncesto que permita modelar y entrenar agentes inteligentes en contextos de toma de decisiones estratégicas. Para ello, se ha diseñado un tablero estructurado que reproduce el campo de juego, diferenciando zonas clave que influyen tanto en la probabilidad de éxito de las acciones como en el sistema de recompensas asignadas.

El tablero se ha definido como una cuadrícula de 15 filas por 28 columnas, representando a escala 1:1 los 15 x 28 metros de un campo de baloncesto reglamentario. Este diseño genera un total de 420 casillas, cuya distribución se muestra en la [Figura 3.1](#). Cada casilla pertenece a una zona concreta del campo, lo que determina su relevancia estratégica y su impacto en la recompensa que recibe el agente tras una acción. Se han definido cuatro tipos de casillas según su valor táctico, diferenciadas visualmente por colores:

- **Casillas grises:** representan las zonas exteriores del perímetro, comúnmente conocidas como la línea de triple. Desde estas casillas, la probabilidad de acierto en el tiro es menor, pero la recompensa es mayor, otorgando 3 puntos en caso de anotación, conforme al reglamento oficial.
- **Casillas naranjas:** corresponden a zonas interiores cercanas al aro. Los tiros realizados desde estas posiciones tienen una mayor probabilidad de éxito, aunque con una recompensa inferior, concretamente 2 puntos por canasta.

- **Casillas naranjas oscuras:** situadas justo debajo del aro, constituyen el área de mayor probabilidad de anotación debido a la cercanía a la canasta. Al igual que las casillas naranjas, permiten sumar 2 puntos en caso de acierto.
- **Casilla gris oscura:** representa la canasta dentro del tablero.

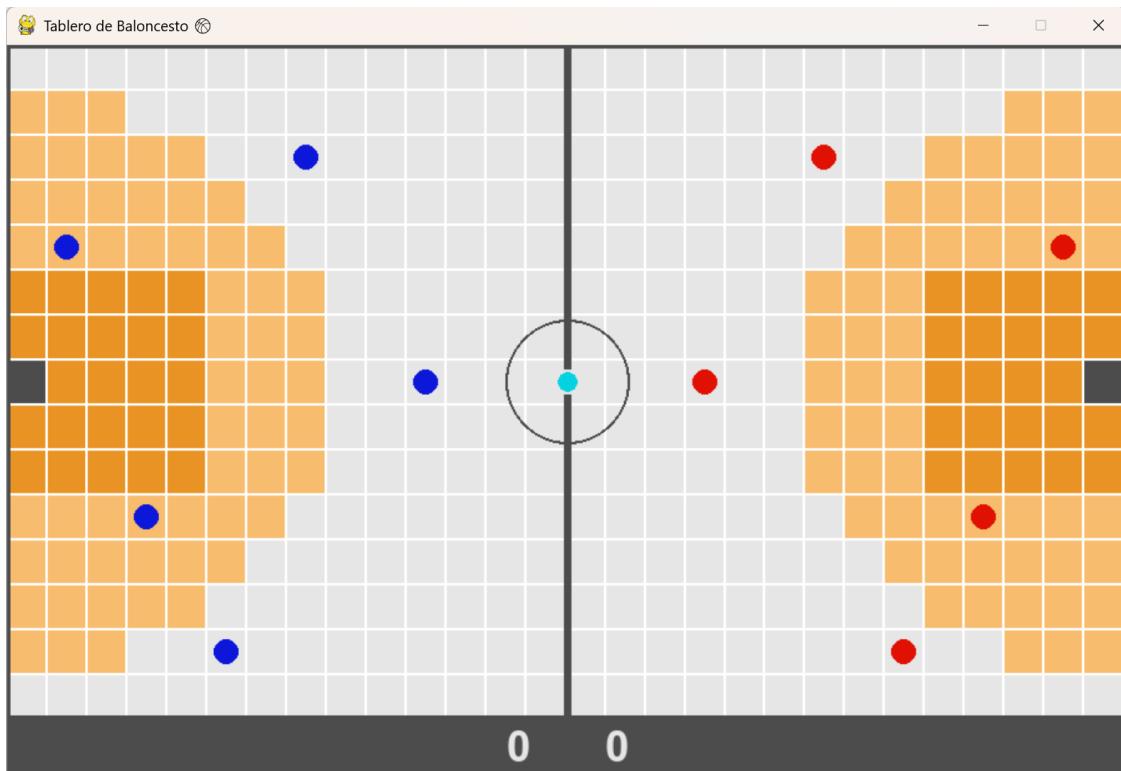


Figura 3.1: Representación visual del tablero de baloncesto en Pygame

El diseño de estas zonas se ha basado en una interpretación simplificada de las reglas oficiales de la FIBA [3], respetando las proporciones del terreno de juego y adaptándolas a una representación rectangular mediante cuadrículas. Esta decisión se ha tomado con el objetivo de evitar ambigüedades: si se hubieran representado curvas como en un campo real, una misma casilla podría intersectar varias zonas, dificultando así la gestión del entorno. Con el enfoque actual, cada casilla tiene un valor único asociado, lo que simplifica la implementación y garantiza una gestión coherente de zonas, posiciones y movimientos.

Además de la zona de juego, la interfaz incluye en su parte inferior un marcador digital que refleja en tiempo real la puntuación acumulada por cada equipo durante el episodio (véase Figura 3.1). Este marcador está sincronizado con el sistema de recompensas: al anotar desde una determinada zona, se actualiza tanto la puntuación visible como la retroalimentación interna que recibe el agente. Cabe destacar que dicho marcador se reinicia al comienzo de cada episodio de simulación.

Aunque el tablero representa el campo completo, en este proyecto se ha restringido la acción al medio campo ofensivo, es decir, a la mitad inicial del terreno de juego desde el punto de vista del equipo atacante. En esta área interactúan

dos equipos: el equipo rojo, encargado de ejecutar acciones ofensivas (movimientos, pases y lanzamientos), y el equipo azul, que actúa como defensor. Esto reduce la complejidad inicial del entorno y permite focalizar el entrenamiento del agente en situaciones clave del juego ofensivo.

La disposición inicial de los jugadores sobre el tablero al comienzo de un episodio se muestra en la [Figura 3.2](#), donde el jugador en posesión del balón se identifica mediante un círculo interior de color azul cián. Esta configuración ha sido diseñada bajo un criterio de simplicidad tanto visual como funcional. Los jugadores del equipo rojo (ofensivo) se ubican próximos al eje central del campo, mientras que los jugadores del equipo azul (defensivo) se alinean frente al aro. Dicha colocación no responde a esquemas tácticos reales del baloncesto, sino que ha sido seleccionada de forma intencionada para facilitar la comprensión visual del entorno, incluso para usuarios sin conocimientos previos del deporte. Asimismo, favorece el aprendizaje del agente en las fases iniciales, al evitar complejidades innecesarias derivadas de la interpretación de formaciones estratégicas o patrones avanzados de juego.

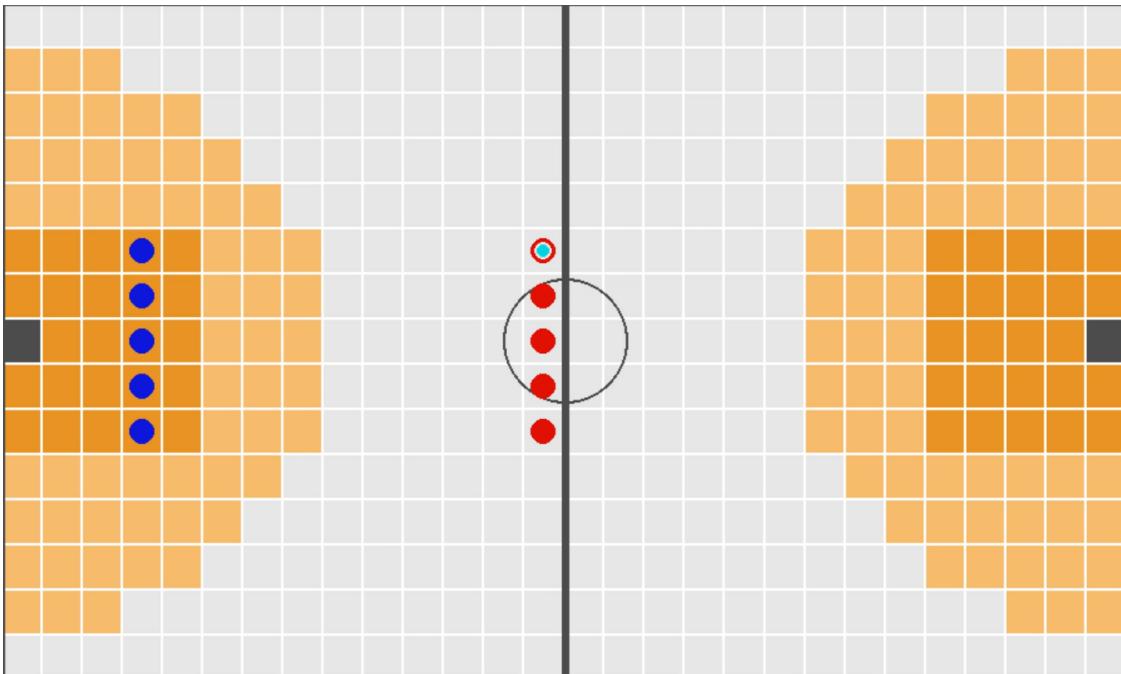


Figura 3.2: Distribución inicial de los jugadores en el tablero al inicio del episodio

3.2. Arquitectura técnica y herramientas

El desarrollo del entorno de simulación se ha estructurado sobre una arquitectura modular, orientada a desacoplar los componentes de visualización, entrenamiento y evaluación. Esta organización permite escalar el proyecto de forma progresiva, facilita la comparación entre versiones y mejora la gestión del ciclo de vida de los experimentos.

Todo el sistema ha sido implementado íntegramente en el lenguaje de programación Python, aprovechando su amplio ecosistema de librerías especializadas en

inteligencia artificial, simulación y visualización. La mayoría de los módulos están organizados como scripts independientes con extensión `.py`, mientras que los análisis de resultados, pruebas exploratorias y visualizaciones complementarias se desarrollan en notebooks interactivos (`.ipynb`) mediante el entorno `Jupyter`. También se utilizan archivos `.txt` para almacenar información puntual, formatos estructurados como `.csv` y `.json` para la exportación de datos, y scripts `.bat` destinados a automatizar procesos vinculados al sistema de guardado.

La base del entorno está construida sobre el framework `Gymnasium` [20], sucesor de OpenAI Gym, que proporciona una interfaz estándar para el diseño de entornos de aprendizaje por refuerzo. Esta herramienta ofrece mayor robustez, extensibilidad y soporte activo, y resulta completamente compatible con las bibliotecas empleadas en este trabajo, como la librería `Stable-Baselines3` [21].

Cada versión del entorno implementa su propia lógica de acciones, recompensas y condiciones de finalización, lo que permite modularidad y una evolución gradual de la complejidad. Para maximizar la eficiencia computacional durante el entrenamiento, se ha desarrollado una versión principal del entorno sin renderizado gráfico, ejecutada desde la línea de comandos (CLI), mientras que la representación visual de los episodios se gestiona por separado mediante `Pygame` [22]. Esta biblioteca, orientada al desarrollo de videojuegos en Python, permite una visualización clara, dinámica e intuitiva de las simulaciones, facilitando la comprensión de las dinámicas del entorno. Gracias a esta separación entre entrenamiento y visualización, se evitan ralentizaciones y se optimiza el rendimiento. También, se registra automáticamente cada episodio en archivos `.csv` para su posterior análisis o reproducción gráfica.

El agente se entrena mediante la librería `Stable-Baselines3` [21], que implementa algoritmos de aprendizaje por refuerzo compatibles con `Gymnasium` [20]. En las primeras versiones del entorno se emplea DQN [6], y en versiones posteriores se adopta PPO [7] por su mayor estabilidad en entornos con espacios de acción más complejos. Esta evolución del enfoque se analiza en el [Capítulo 4](#). Ambos algoritmos utilizan la política `MlpPolicy`, términos introducidos previamente en el [Capítulo 2](#).

Durante el entrenamiento, se registra el rendimiento del agente utilizando `MLflow` [23], herramienta de código abierto para el seguimiento de experimentos. Cada ejecución almacena automáticamente los modelos entrenados, sus métricas asociadas, los hiperparámetros utilizados y cualquier archivo adicional generado, lo que garantiza la reproducibilidad y trazabilidad de los resultados.

Espacio de estados, acciones y recompensas

El **espacio de acciones** varía en función de la versión del entorno implementada. En la versión básica, el agente controla a un único jugador mediante un espacio de acciones discreto `Discrete(5)`, donde cada valor representa una acción:

- 0: mover una casilla hacia arriba
- 1: mover una casilla hacia abajo

- 2: mover una casilla hacia la izquierda
- 3: mover una casilla hacia la derecha
- 4: lanzar a canasta

En las versiones posteriores, el entorno evoluciona hacia un enfoque multiagente, donde el agente controla simultáneamente a los cinco jugadores del equipo mediante un espacio de acciones $\text{MultiDiscrete}([6] * 5)$. Este permite que cada jugador tome decisiones de forma autónoma, seleccionando entre seis posibles acciones: desplazarse en cualquiera de las cuatro direcciones, lanzar a canasta o pasar el balón. No obstante, las acciones de pase y tiro solo pueden ser ejecutadas por el jugador que tenga la posesión del balón, mientras que los demás únicamente pueden moverse. Esta evolución en la estructura de acciones refleja el aumento progresivo de complejidad táctica.

El **sistema de recompensas** está inspirado en la lógica de puntuación propia del baloncesto, aunque ha sido adaptado a las particularidades del entorno de simulación. Si bien no todas las versiones del entorno implementan la totalidad de estas recompensas, la política general aplicada se resume en los siguientes casos:

- +3: si el lanzamiento resulta exitoso desde una casilla gris (zona de triple).
- +2: si se anota desde una casilla naranja o naranja oscura (zonas interiores).
- -1: si se pierde la posesión del balón, ya sea por intercepción o acción inválida.
- 0: si el lanzamiento no resulta en canasta o si se ejecuta una acción que no conlleva recompensa directa, como moverse o pasar el balón sin que ocurra ningún evento significativo.

Capítulo 4

Estrategia de entrenamiento

Una vez mencionada la arquitectura técnica del entorno y definidos sus componentes clave, se procede al diseño de la estrategia de entrenamiento del agente inteligente. Esta estrategia adopta un enfoque incremental, estructurado en versiones sucesivas que introducen de forma progresiva nuevas reglas, restricciones y elementos tácticos propios del baloncesto.

Este planteamiento escalonado no solo permite que el agente adquiera competencias de forma gradual, adaptándose progresivamente a situaciones de juego cada vez más complejas, sino que también ha facilitado la implementación general del proyecto. Al introducir nuevas funcionalidades de forma controlada, se evita una sobrecarga inicial de variables que dificultaría el aprendizaje efectivo. Cada versión se construye sobre la anterior, heredando su lógica y estructura base, pero añadiendo mejoras que modifican la dinámica del entorno y enriquecen el proceso.

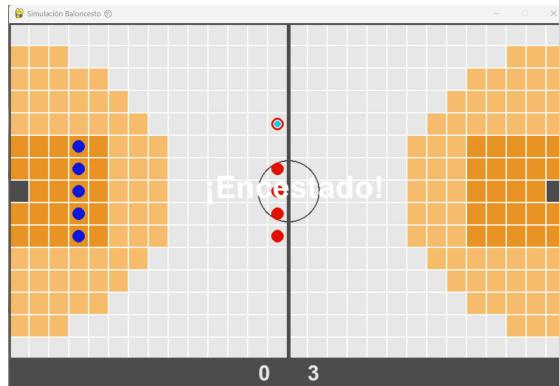
4.1. Versión 1: Entorno básico y controlado

La primera versión del entorno constituye el punto de partida sobre el que se ha construido todo el sistema. Su desarrollo ha permitido establecer la estructura modular del proyecto, definir la lógica básica de interacción y sentar las bases tanto del entrenamiento del agente como de la visualización del entorno.

En esta etapa inicial se implementan los aspectos descritos en el [Capítulo 3](#). El agente, en su primera versión, controla únicamente a un jugador del equipo rojo, quien comienza cada episodio en posesión del balón. El resto de los jugadores, tanto aliados como rivales, permanecen estáticos, actuando como obstáculos fijos sobre el tablero. El espacio de acciones disponible para el agente se limita a cinco opciones: moverse una casilla en cualquiera de las cuatro direcciones cardinales (arriba, abajo, izquierda o derecha) o lanzar a canasta desde su posición actual.

La lógica del entorno impide que varios jugadores ocupen la misma casilla simultáneamente, evitando colisiones y garantizando la consistencia del estado del tablero. Cada episodio finaliza inmediatamente después del intento de lanzamiento, lo que reduce la complejidad del ciclo de aprendizaje.

El sistema de recompensas está definido de forma determinista en función de la zona desde la que se ejecuta el lanzamiento, siguiendo la lógica descrita en capítulos anteriores. En esta versión, se asume una probabilidad de acierto del 100 % al lanzar. Así, se otorga una recompensa de +3 si lanza desde una casilla gris, +2 desde una casilla naranja o naranja oscura, y 0 si realiza un movimiento sin lanzar a canasta ([Figura 4.1](#)).



[Figura 4.1:](#) Estado final del tablero tras un lanzamiento en la versión 1

Para el entrenamiento del agente se utiliza el algoritmo DQN [\[6\]](#), adecuado para entornos con un único agente y espacio de acción discreto, como el planteado en este escenario. Junto con la política `MlpPolicy`, idónea para modelar relaciones espaciales simples entre la posición del jugador y la canasta.

En cuanto a los parámetros utilizados durante el entrenamiento, se han definido los siguientes hiperparámetros específicos [\[24\]](#), seleccionados con el objetivo de asegurar una dinámica de aprendizaje estable y efectiva desde las primeras etapas:

- **Número total de pasos:** representa la cantidad total de acciones que el agente ejecuta a lo largo de todo el entrenamiento, sumando los pasos de todos los episodios. A diferencia de un episodio, que comprende una secuencia completa de interacciones hasta un estado final, cada paso corresponde a una única acción tomada por el agente. Este parámetro determina cuántas veces el agente interactúa con el entorno para aprender y actualizar su política, y se ajusta según su evolución para optimizar la duración y efectividad del entrenamiento.
- **Tasa de aprendizaje (learning rate):** 0.0003. Este valor regula el tamaño de los ajustes aplicados sobre los pesos de la red durante la retropropagación. Se ha optado por una tasa baja para favorecer una convergencia estable y gradual sin provocar oscilaciones excesivas.
- **Tamaño del buffer de experiencia (buffer size):** 100.000 transiciones. Este parámetro determina la capacidad de memoria utilizada para almacenar experiencias pasadas, también llamadas transiciones. Al reutilizar estas transiciones durante el entrenamiento, el agente puede aprender de una mayor diversidad de situaciones y reducir el riesgo de sobreajuste a interacciones recientes.

- **Tamaño del batch:** 32 muestras por actualización. Este valor representa un equilibrio común entre precisión y eficiencia computacional.
- **Factor de descuento (gamma):** 0.99. Este coeficiente pondera la importancia de las recompensas futuras frente a las inmediatas, lo cual es clave en entornos donde una secuencia de acciones desemboca en un resultado final valioso, como una anotación.

En esta versión se configura el sistema de registro y trazabilidad de experimentos mediante [MLflow \[23\]](#), lo que permite documentar de forma completa el proceso de entrenamiento, mejorar la reproducibilidad de los resultados y facilitar la comparación entre distintas configuraciones. Además, para agilizar el acceso a su interfaz gráfica, se incorpora un archivo ejecutable `.bat` que lanza automáticamente la aplicación web en el navegador, evitando la ejecución manual de comandos.

4.2. Versión 2: Interacción y lógica de tiro

La segunda versión del entorno representa un avance significativo en cuanto a complejidad y realismo con respecto a la anterior. Su desarrollo introduce la posibilidad de controlar de forma simultánea a los cinco jugadores del equipo ofensivo, mediante una acción conjunta codificada como un vector de decisiones individuales. Cada jugador puede ejecutar de manera independiente una de las siguientes acciones: moverse en las cuatro direcciones cardinales, lanzar a canasta (si tiene el balón) o pasar el balón a un compañero.

Esta ampliación transforma el entorno en una simulación con dinámica multi-agente, donde el agente debe coordinar los movimientos del equipo, optimizar las posiciones y decidir cuándo y a quién pasar el balón. Los jugadores sin balón adquieren un rol activo, desplazándose por el tablero para generar espacios o facilitar opciones de pase, lo que fomenta comportamientos colaborativos más elaborados y una interacción más dinámica.

Otra innovación introducida en esta versión es la incorporación de una lógica de tiro basada en probabilidad, que reemplaza el sistema determinista de recompensas utilizado previamente. A partir de este punto, cada intento de lanzamiento ya no garantiza un resultado fijo, sino que se evalúa en función de una probabilidad de éxito que depende directamente de la posición del jugador en el tablero.

Para modelar esta probabilidad, se ha optado por una función de caída cuadrática en función de la distancia a la canasta. Esta elección responde a la necesidad de representar de manera realista la creciente dificultad de los tiros lejanos. Frente a una función lineal, que penalizaría de forma insuficiente las posiciones alejadas del aro, y una función cúbica, que las castigaría en exceso reduciendo el margen táctico del agente, la función cuadrática ofrece un equilibrio más adecuado. Penaliza progresivamente los lanzamientos a medida que aumenta la distancia, sin llegar a descartarlos por completo, lo que favorece un comportamiento más estratégico por parte del agente. Entre las ventajas que presenta este enfoque, destacan:

- Promueve decisiones más inteligentes, incentivando que el agente se acerque al aro antes de lanzar.
- Penaliza los tiros lejanos sin llegar a eliminarlos del espacio de acción, manteniendo un equilibrio razonable entre riesgo y recompensa.

En consecuencia, la recompensa obtenida por un tiro exitoso ya no es constante, sino que se calcula como el producto entre el valor asignado a la zona de lanzamiento (2 o 3 puntos, según se trate de casillas naranjas o grises) y la probabilidad de éxito estimada desde esa posición. Esto aporta una capa de incertidumbre y realismo adicional, obligando al agente a evaluar no solo el valor de la acción, sino también su viabilidad.

Desde el punto de vista de la visualización, el entorno incorpora mensajes informativos tras cada lanzamiento, indicando si el tiro ha sido encestado o fallado (véase [Figura 4.2](#)), lo que permite realizar un seguimiento más cualitativo del comportamiento del agente durante las pruebas.

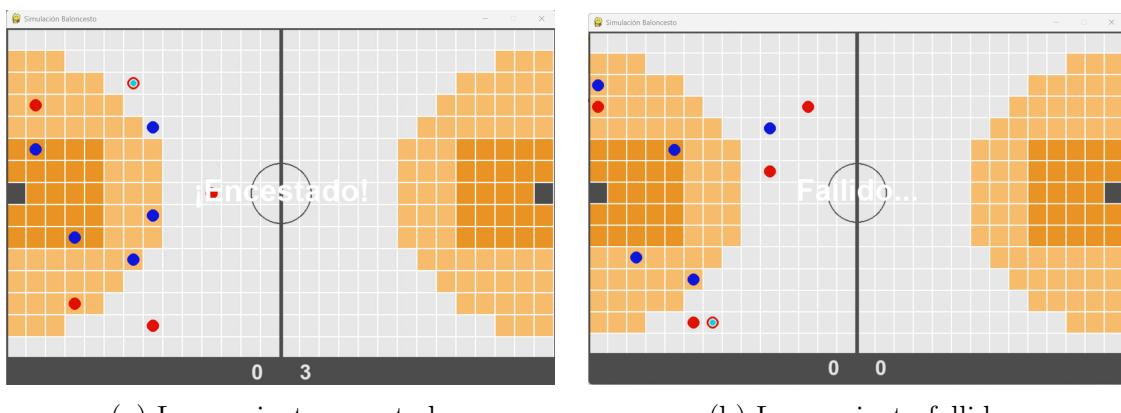


Figura 4.2: Estado final del tablero tras un lanzamiento en la versión 2

Para el entrenamiento se utiliza el algoritmo PPO [7], que se adapta mejor a entornos multiagente y espacios de acción de tipo `MultiDiscrete`. Esta elección permite que el modelo aprenda políticas conjuntas para todo el equipo ofensivo de manera más eficiente, favoreciendo la coordinación en la toma de decisiones. Salvo este cambio, el resto de los hiperparámetros del modelo se mantienen sin modificaciones respecto a la versión anterior.

En resumen, esta versión consolida la transición desde un entorno individual y determinista a uno colectivo y táctico, abriendo la puerta al aprendizaje de estrategias ofensivas más complejas y realistas.

4.3. Versión 3: Defensa e intercepciones

La tercera versión del entorno incorpora un componente fundamental del baloncesto real: la defensa activa. Hasta este punto, el agente se había centrado exclusivamente en la ejecución ofensiva, sin tener en cuenta la presión ejercida por el

equipo rival. Con esta nueva versión, se introduce una lógica defensiva que añade una mejora significativa, obligando al agente a tomar decisiones más informadas y a valorar mejor el contexto de su entorno.

Una de las principales novedades es la penalización sobre la probabilidad de acierto en los lanzamientos, determinada por la cercanía de los defensores al tirador. Cuanto más próximos se encuentren los jugadores rivales, mayor será la penalización aplicada. El sistema establece un aumento progresivo en la dificultad del tiro: si un defensor está en una casilla adyacente, la penalización es notable; si se encuentra a dos casillas de distancia, el impacto es menor, pero aún significativo. A partir de esa distancia, la influencia defensiva se considera despreciable y no se aplica ninguna penalización adicional. Además, el efecto es acumulativo si varios defensores rodean al tirador, aunque se establece un umbral mínimo para asegurar que la probabilidad de encestar nunca se reduzca completamente, evitando que el agente descarte por completo la opción de lanzar.

Adicionalmente, se introduce un sistema de intercepciones que afecta tanto a los tiros como a los pases. La probabilidad de que una acción ofensiva sea interceptada depende de la cercanía del defensor al objetivo (ya sea el aro o el jugador receptor). Esta probabilidad se modela mediante una relación inversamente proporcional a la distancia: cuanto más cerca esté un defensor del punto objetivo, mayor será la probabilidad de intercepción; a medida que aumenta la distancia, dicho riesgo disminuye proporcionalmente.

En caso de producirse una intercepción, el episodio finaliza de forma inmediata y el agente recibe una penalización de -1, ampliando así el sistema de recompensas disponible (+3, +2, 0, -1). Esta retroalimentación negativa tiene como objetivo fomentar una toma de decisiones más prudente, incentivando al agente a evitar zonas de alto riesgo y a desarrollar estrategias más sólidas frente a la presión defensiva.

Desde el punto de vista visual, el entorno refuerza esta dinámica mediante la aparición de un mensaje claro en pantalla indicando “Pelota Perdida” cada vez que se produce una intercepción, lo que facilita el análisis cualitativo del comportamiento del agente durante las pruebas.

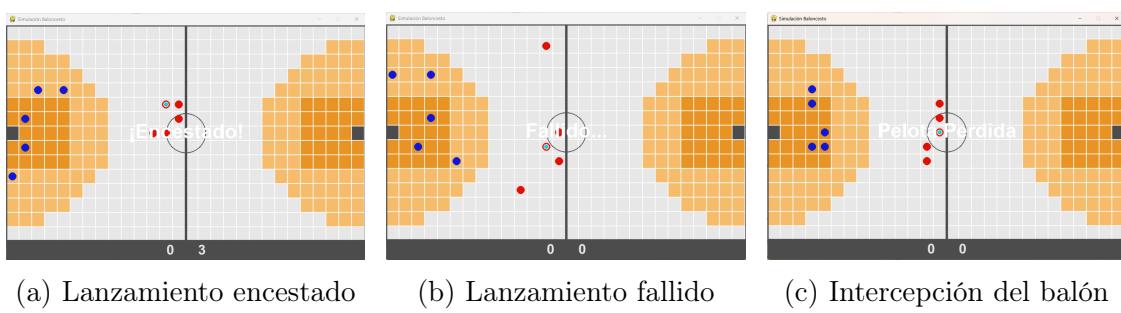


Figura 4.3: Estado final del tablero tras un lanzamiento en la versión 3

Para el entrenamiento se sigue utilizando PPO [7] con `MlpPolicy`, manteniendo los mismos hiperparámetros que en la versión anterior, ya que continúan siendo adecuados para este nivel de complejidad.

En definitiva, esta versión obliga al agente a considerar múltiples factores en cada acción: no solo su posición y la de sus compañeros, sino también la presión ejercida por el rival. Como resultado, el proceso de toma de decisiones se vuelve más sofisticado y realista, acercando el entorno simulado al baloncesto real.

4.4. Versión 4: Asignación de roles y datos de NBA

La cuarta versión del entorno supone un avance clave en términos de realismo y complejidad. En esta etapa se integran datos reales obtenidos a través de la *API oficial de la NBA* [25], lo que permite contextualizar las decisiones del agente a partir de situaciones observadas en partidos profesionales.

Con este objetivo, se han recopilado datos de la temporada 2024–2025 correspondientes a cinco jugadores del equipo Los Angeles Clippers: James Harden, Kris Dunn, Norman Powell, Kawhi Leonard e Ivica Zubac. La selección responde al propósito de representar distintos perfiles funcionales dentro de una plantilla profesional. Según la información extraída, Harden, Dunn y Powell figuran como escoltas o bases; Leonard aparece como alero; y Zubac como pívot.

Esta variedad posicional permite construir perfiles de comportamiento diferenciados y analizar, a partir de datos reales, cómo se desempeñan distintos tipos de jugadores en zonas concretas del campo. Esta distinción cobra especial relevancia ya que se asignan roles explícitos a los jugadores del equipo ofensivo y se ajusta la probabilidad de acierto en los lanzamientos en función de la información registrada para cada perfil. De este modo, se refuerza la coherencia entre el comportamiento del agente y los patrones estratégicos observados en el baloncesto profesional.

La recopilación de los datos se ha automatizado mediante un script que gestiona todo el flujo de trabajo: desde la búsqueda del identificador de cada jugador hasta la recuperación de su posición oficial en cancha y la descarga de los tiros realizados durante la temporada. Para ello, se utilizan los endpoints `commonplayerinfo` y `shotchartdetail` de la *API oficial de la NBA* [25], que permiten obtener, respectivamente, la posición de juego y las coordenadas de cada intento de tiro (*Field Goal Attempts, FGA*). El resultado es un conjunto de datos estructurados por jugador, almacenados en archivos CSV, listos para su posterior análisis y mapeo al entorno.

Estos datos se procesaron inicialmente usando la extensión `Data Wrangler` [26], una herramienta en Python diseñada para facilitar la limpieza, transformación y exploración de datos estructurados. Posteriormente, fueron representados gráficamente mediante notebooks de `Jupyter` para analizar su distribución espacial.

La [Figura 4.4](#) muestra la dispersión conjunta de todos los tiros intentados, coloreados según la zona del campo (*shot zone area*). Se observa que las acciones se concentran hacia una única canasta ya que cada jugada registrada está orientada hacia la misma dirección de ataque. Esto proporciona una visión clara del comportamiento espacial de los lanzamientos, especialmente en zonas perimetrales (como las esquinas o el centro del perímetro) y debajo del aro.

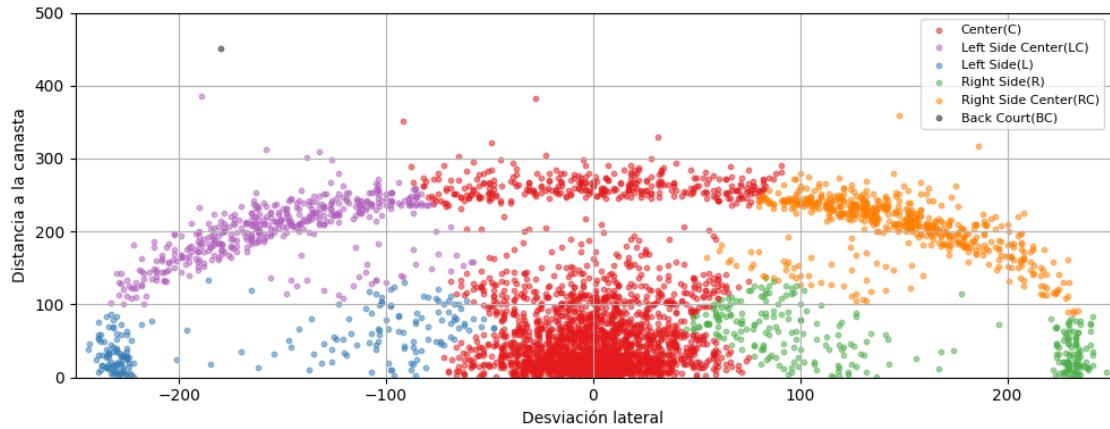


Figura 4.4: Dispersión de los tiros realizados por todos los jugadores seleccionados

En la [Figura 4.5](#) se representa la dispersión individual de los tiros por jugador. En ella se evidencian diferencias notables en los patrones de lanzamiento: por ejemplo, Ivica Zubac concentra sus tiros en la zona cercana al aro (puntos rojos), característica propia de un pívot; mientras que, por ejemplo, James Harden presentan una mayor presencia en el perímetro, especialmente en zonas de triple (naranja y morado), lo que refleja su rol de tiradores exteriores. Esta información resulta esencial para ajustar la probabilidad de éxito en función del perfil ofensivo de cada jugador, permitiendo una personalización más precisa del entorno.

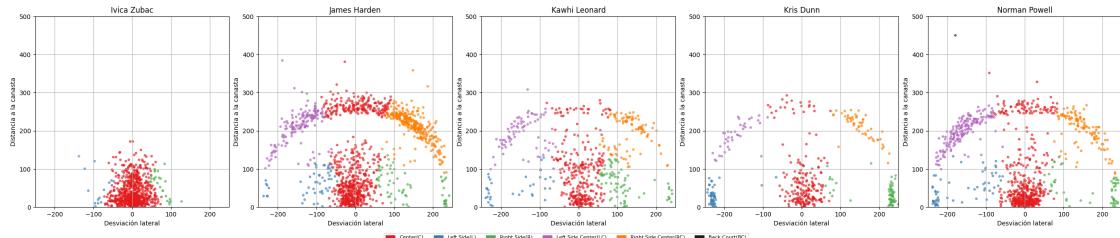


Figura 4.5: Dispersión individual de los tiros realizados por cada jugador

Una vez procesados y comprendidos los datos extraídos, se procedió a su integración en el entorno de simulación. Dado que las coordenadas de lanzamiento estaban expresadas en el sistema de referencia del campo NBA, fue necesario adaptarlas al tablero diseñado, estructurado como una cuadrícula de 15 filas por 28 columnas. Sin embargo, como el entorno se restringe a medio campo, solo se consideraron las columnas de la 0 a la 13, siendo la posición (7, 0) la canasta.

Este proceso de mapeo se llevó a cabo mediante un único script en Python, encargado de rotar, escalar y normalizar las coordenadas originales de lanzamiento para adaptarlas al formato del entorno simulado. Dado que el campo oficial de la NBA está orientado verticalmente, mientras que el tablero utilizado se dispone de forma horizontal, fue necesario transformar las posiciones para que encajaran correctamente en la cuadrícula. El resultado de esta conversión puede observarse en la [Figura 4.6](#), donde se visualiza cómo los tiros han sido proyectados adecuadamente sobre el tablero del entorno.

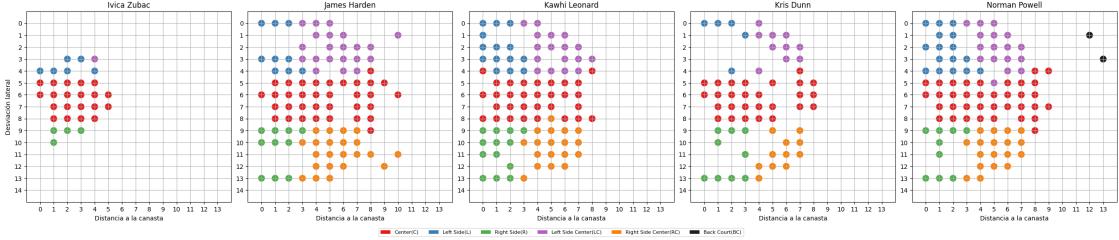


Figura 4.6: Mapeo de los tiros realizados por cada jugador

Sin embargo, tras este mapeo directo, algunas celdas del entorno quedaron sin valores asignados, al no existir lanzamientos reales registrados en esas posiciones. Para cubrir completamente el tablero, requisito importante para poder calcular recompensas en cualquier punto durante el entrenamiento del agente, se recurrió al algoritmo K-Nearest Neighbors (KNN) [27], con el fin de estimar la probabilidad de acierto en las celdas vacías.

Antes de aplicar el modelo, se llevó a cabo un análisis exhaustivo para determinar el número óptimo de vecinos k . Mediante validación cruzada con tres particiones (KFold) y empleando el error cuadrático medio (MSE) como métrica principal, se evaluaron valores de k comprendidos entre 1 y 10. Los resultados obtenidos, representados en la Figura 4.7, indican que un valor entre 8 y 10 ofrece el mejor equilibrio para la mayoría de jugadores. Finalmente, se fijó $k=10$ como valor definitivo.

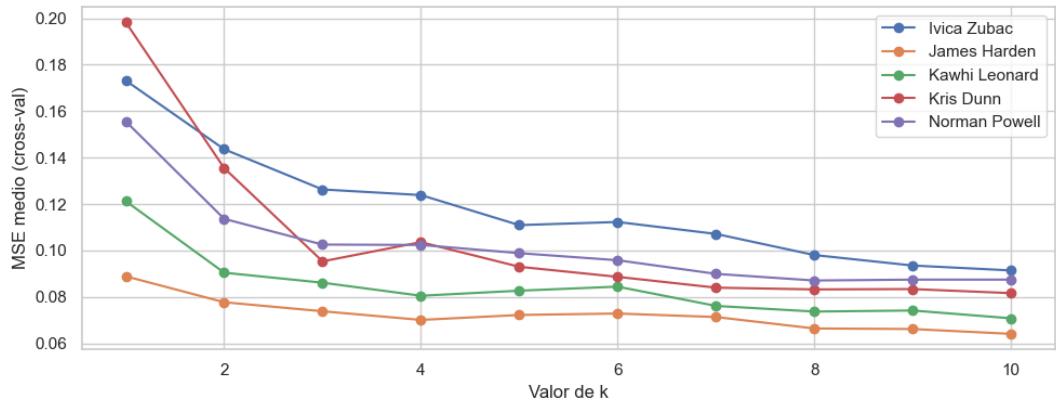
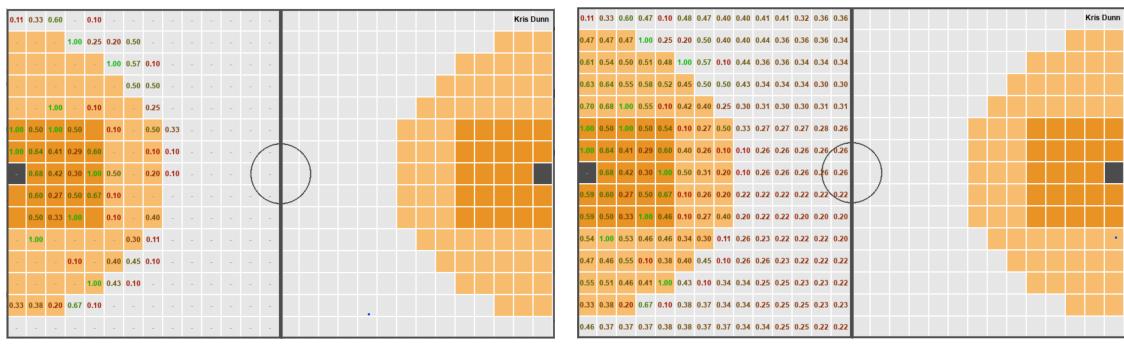


Figura 4.7: Evaluación del valor óptimo de k en el modelo KNN mediante validación cruzada para distintos jugadores

Todo este proceso, desde la conversión inicial de coordenadas hasta la imputación de valores ausentes, se ha implementado en un único script, diseñado para ser fácilmente extensible a otros equipos y ligas, y adaptable a una futura automatización mediante *pipelines*. En una primera etapa, el programa genera un archivo JSON con las probabilidades reales de acierto por celda, calculadas a partir de los tiros mapeados. Posteriormente, se aplica KNN para completar las posiciones sin datos, empleando como variables de entrada la fila, la columna y la distancia euclídea al aro. De este modo, cada casilla del entorno simulado queda asociada a una probabilidad estimada de anotación, garantizando una representación completa y coherente del tablero para los episodios de entrenamiento.

En la [Figura 4.8](#) se presenta un ejemplo visual del mapa de tiro proyectado sobre el tablero de simulación para un jugador concreto, comparando la distribución original de los tiros registrados con el resultado obtenido tras aplicar KNN para imputar las celdas vacías. Esta comparación permite observar cómo el modelo completa las zonas sin datos, generando una representación más densa y continua del campo. Sin embargo, es importante señalar que algunas celdas presentan valores extremos, como probabilidades de acierto del 100 %, lo cual no es estadísticamente realista. Estos valores, aunque reflejen lanzamientos aislados con éxito, pueden introducir sesgos en el entrenamiento del agente, sobreestimando determinadas regiones del tablero. Para mitigar este efecto, resultaría aconsejable aplicar variantes del algoritmo como KNN *smoothing*, una técnica que suaviza la estimación promediando las salidas de los vecinos más cercanos ponderadas por su distancia, reduciendo así la influencia de valores atípicos y mejorando la continuidad espacial de la predicción [28].



(a) Distribución original de tiros mapeados (b) Distribución completa tras aplicar KNN

Figura 4.8: Comparación entre el mapeo original de tiros y el resultado tras aplicar KNN para estimar probabilidades en celdas sin datos

Finalmente, el resto de aspectos del entorno se mantienen sin cambios respecto a versiones anteriores, asegurando la continuidad en la estructura y funcionamiento general del sistema. En definitiva, esta fase consolida un entorno de simulación notablemente más realista al integrar la asignación táctica de roles con datos empíricos procedentes de partidos oficiales de la NBA, elevando así la fidelidad y aplicabilidad del modelo en contextos estratégicos reales.

Capítulo 5

Evaluación experimental

Una vez descritas en detalle las versiones desarrolladas del entorno, en este capítulo se evalúa el comportamiento del agente entrenado, así como el funcionamiento general del entorno. Se analizan las decisiones aprendidas, su coherencia con patrones realistas y el impacto que han tenido las mejoras introducidas.

5.1. Metodología de evaluación

El objetivo de esta fase es analizar cómo evoluciona el comportamiento del agente a medida que aumenta la complejidad del entorno, desde la versión básica (v1) hasta la más avanzada (v4). Cada versión introduce nuevos elementos, como defensa, roles o datos reales, y se evalúa la capacidad del agente para adaptarse y tomar decisiones más complejas.

Se entrenan cinco modelos por versión, variando únicamente el número de pasos (500, 1000, 10000, 50000 y 100000), con el resto de hiperparámetros constantes. Esto permite observar tanto la progresión del aprendizaje como el rendimiento final. Las métricas utilizadas incluyen la recompensa por episodio, la frecuencia y distribución de acciones, o la tendencia de lanzamientos.

5.2. Resultados por versión

Antes de analizar cada versión de forma individual, las figuras 5.1 y 5.2 ofrecen una visión global del comportamiento del agente en términos de frecuencia de acciones y momento del lanzamiento. Ambas reflejan cómo el agente adapta su toma de decisiones a medida que se incrementa la complejidad del entorno.

En la versión 1, el entorno es simple y sin interacción entre jugadores. Predominan los lanzamientos, especialmente los inmediatos (82,7 %), junto con un desplazamiento preferente hacia la derecha del campo, lo que refleja una estrategia impulsiva orientada a finalizar el episodio rápido desde o cerca de la posición inicial.

En la versión 2, con la incorporación de la acción de pase, se aprecia una drástica reducción de lanzamientos inmediatos (9,8 %) y un aumento significativo en

la planificación: más del 90 % de los tiros se realizan tras algún desplazamiento. Los pases adquieren protagonismo y los movimientos se reparten de forma más equilibrada, lo que indica una estrategia más colaborativa y distribuida.

La versión 3 incorpora defensa activa, lo que provoca un aumento de los lanzamientos tras menos de cinco pasos (55 %), un repunte de tiros inmediatos (34,3 %) y una reducción del número de pases, posiblemente como respuesta al riesgo añadido por la presión defensiva definida (véase [Sección 4.3](#)).

En la versión 4, con roles asignados y datos reales, el comportamiento se vuelve más equilibrado: los tiros tras más de cinco pasos alcanzan el 53,1 %, lo que sugiere una mayor capacidad del agente para adaptarse al contexto y tomar decisiones estratégicamente informadas.

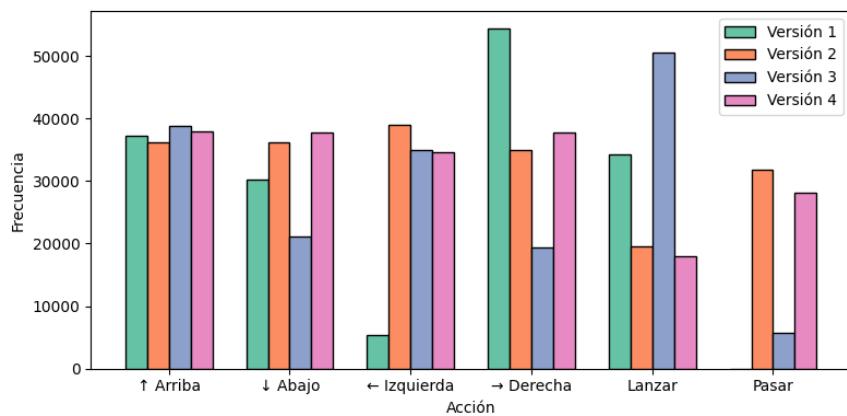


Figura 5.1: Frecuencia total de acciones por versión

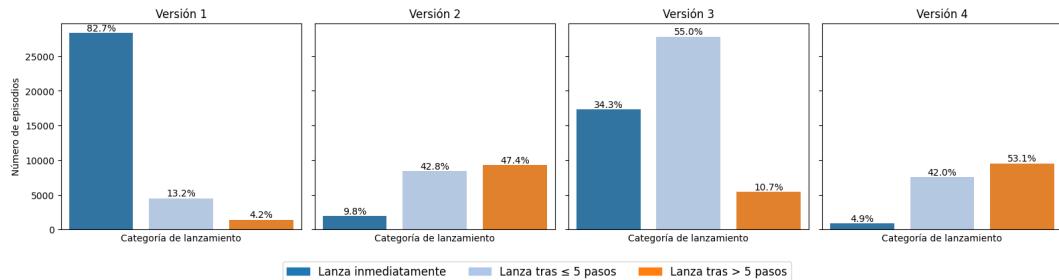


Figura 5.2: Distribución de lanzamiento según acciones del episodio

Si se analiza ahora el resultado de los lanzamientos, independientemente del momento en que se efectúan, la [Figura 5.3](#) refleja una evolución entre versiones. En la versión 1, todos los lanzamientos se consideran exitosos, ya que no se contempla la lógica de fallo o intercepción; se asume que cualquier intento acaba en canasta, lo que convierte esta fase en una referencia artificial. En la versión 2, los aciertos son el 23,7 % y los fallos el 76,3 %, lo que refleja que, aunque el agente empieza a planificar mejor, aún no identifica zonas de tiro con mayor probabilidad de éxito. En la versión 3, con presencia de defensores, los aciertos bajan al 19,5 % y aparecen intercepciones (10,3 %), evidenciando el impacto de la presión defensiva. Por último, en la versión 4, el agente mejora su precisión con un 39 % de aciertos y prácticamente elimina las

intercepciones (0,1 %), lo que indica una toma de decisiones más eficaz e influida por la asignación de roles y el uso de datos reales.

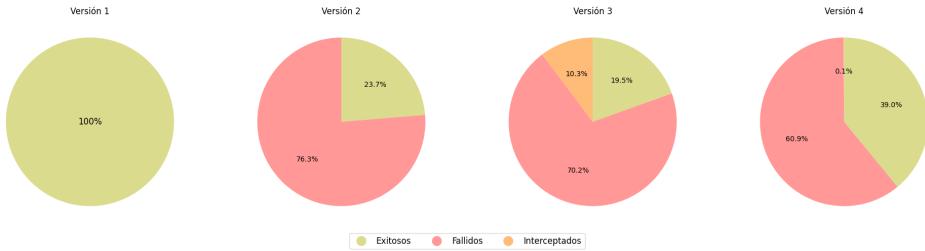
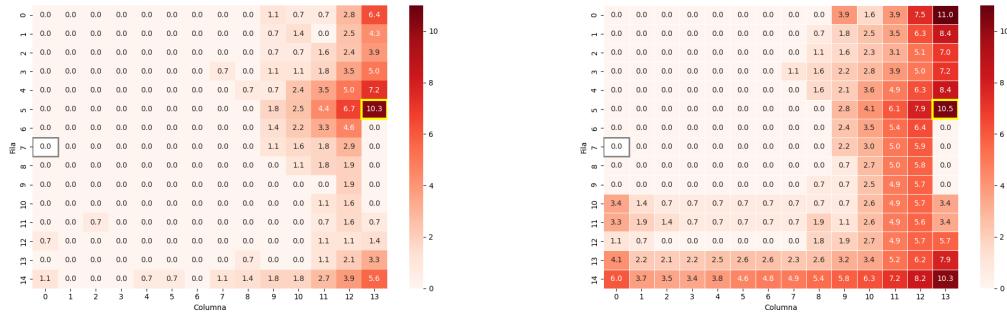


Figura 5.3: Distribución del resultado de los lanzamientos por versión

En conjunto, esta evolución progresiva evidencia cómo el agente deja atrás comportamientos automáticos para adoptar estrategias más complejas, realistas y adaptadas al entorno. A continuación, se analiza cada versión en detalle.

Pasando al **análisis concreto de la versión 1**, la recompensa obtenida por episodio confirma el enfoque impulsivo del agente: prácticamente todos los episodios finalizan con una recompensa de +3. La mediana de duración es de un solo paso y la media apenas alcanza los 4.7, lo que indica que el agente ha aprendido a lanzar de forma inmediata desde una posición cercana al inicio para maximizar la recompensa. La [Figura 5.4](#) ilustra claramente este patrón. En la [Figura 5.4a](#), correspondiente a la distribución de lanzamientos por celda, destaca la casilla (5, 13) o sus proximidades, con una frecuencia superior al resto, por lo que la mayoría de los tiros se producen cerca de la posición inicial del jugador (resaltada con borde amarillo). Por otro lado, la [Figura 5.4b](#) muestra mayor concentración de las posiciones del jugador con balón en las columnas más alejadas del aro (marcado con borde gris), especialmente entre las columnas 11 y 13. Todo ello sugiere que el agente apenas se desplaza antes de lanzar, ya que es capaz de obtener la recompensa máxima desde su posición inicial.



(a) Posiciones lanzamiento

(b) Posiciones jugador

Figura 5.4: Distribución espacial de eventos durante los episodios en la versión 1

Este comportamiento es óptimo en un entorno sin restricciones, pero extremadamente básico. La [Tabla 5.1](#) muestra que esta estrategia se mantiene constante incluso tras 1000000 iteraciones. Aunque los tiros inmediatos disminuyen ligeramente con el entrenamiento a partir de las 50000, no hay mejoras significativas en rendimiento ni en variedad de acciones.

Iteraciones	Recompensa media	Recompensa máxima	% tiros inmediatos
500	3,00	3,00	92,9 %
1000	3,00	3,00	94,8 %
10000	3,00	3,00	94,3 %
50000	3,00	3,00	78,2 %
100000	3,00	3,00	77,0 %

Tabla 5.1: Resumen de métricas por iteraciones en la versión 1

En resumen, esta fase demuestra que el agente converge rápidamente hacia una política acorde pero simple: lanzar desde el inicio sin explorar. Esto sirve como referencia para evaluar la evolución del comportamiento en entornos más complejos.

Centrándonos en la **evaluación de la versión 2**, los mapas de calor presentados en la [Figura 5.5](#) reflejan una clara evolución hacia un modelo más colaborativo. A diferencia de la versión 1, donde la estrategia aprendida era extremadamente básica y se limitaba a lanzar desde o cerca de la posición inicial ([Figura 5.4](#)), ahora se ve cierta diversidad en las zonas desde las que se ejecutan los lanzamientos y movimientos. Aunque la celda (5,13), correspondiente a la posición inicial, sigue siendo la más utilizada, la frecuencia de lanzamientos desde ella ha disminuido, lo que indica una tendencia a empezar a explorar otras opciones. La distribución espacial revela un uso algo más equilibrado del terreno de juego, con acciones que parecen orientadas a construir jugadas en dirección a la canasta. Este cambio es coherente con las modificaciones introducidas en esta versión (véase [Sección 4.2](#)), donde se incorporó una probabilidad de acierto por celda; sin embargo, parece que esto aún no es lo suficientemente determinante como para inducir un comportamiento aún más táctico. Por tanto, aunque aún predomina una cierta toma de decisiones impulsiva, empiezan a manifestarse movimientos que exploran zonas más próximas al aro.

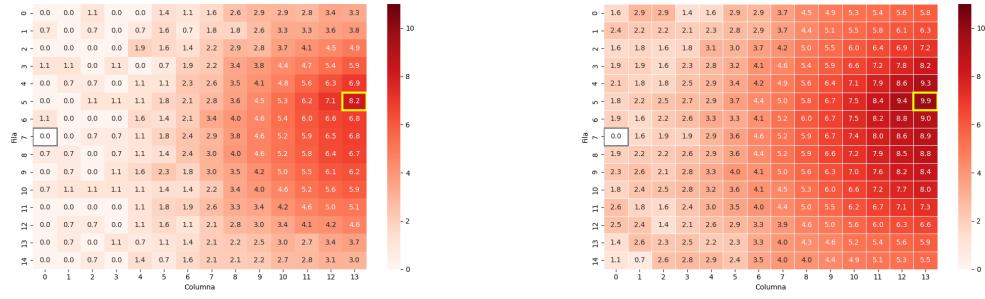


Figura 5.5: Distribución espacial de eventos durante los episodios en la versión 2

Desde una perspectiva cuantitativa, los resultados de la [Tabla 5.2](#) confirman este cambio de comportamiento. La recompensa media por episodio aumenta progresivamente con el número de iteraciones, pasando de 0,12 con 500 iteraciones a 0,18 con 100000. Esta media es inferior a la observada en la versión 1 debido a la introducción de probabilidades de acierto dependientes de la celda de lanzamiento, lo que introduce mayor realismo y dificultad en el entorno. Además, se observa

una notable reducción en los lanzamientos inmediatos: mientras que en la versión anterior más del 75 % de los tiros se realizaban en el primer paso, en esta versión dicha cifra cae hasta un 8,2 % tras 100000 iteraciones. Esto evidencia que el agente ha dejado atrás comportamientos impulsivos y tiende a construir jugadas más elaboradas para maximizar la probabilidad de éxito.

También destaca el aumento en la duración de los episodios, con una mediana de 6 pasos y una media de 8,46. Este incremento refleja que el agente ha aprendido a mantener la posesión, a desplazarse estratégicamente y a coordinar acciones entre jugadores antes de lanzar. En conjunto, estos indicadores evidencian un avance en la complejidad táctica del comportamiento aprendido en esta segunda versión.

Iteraciones	Recompensa media	Recompensa máxima	% tiros inmediatos
500	0,12	1,17	14,2 %
1000	0,14	1,05	16,8 %
10000	0,15	1,32	13,7 %
50000	0,18	1,50	10,9 %
100000	0,18	1,64	8,2 %

Tabla 5.2: Resumen de métricas por iteraciones en la versión 2

Tras introducir la presión defensiva, el **comportamiento del agente en la versión 3** cambia de forma notable respecto a las fases anteriores. La distribución espacial de eventos mostrada en la [Figura 5.6](#) evidencia un retroceso táctico derivado de la complejidad añadida por la presencia de defensores activos y la incorporación de probabilidades de acierto condicionadas por su proximidad (véase [Sección 4.3](#)), lo que afecta tanto a la planificación como a la ejecución ofensiva.

Los mapas de calor muestran que el agente restringe sus opciones de avance para evitar zonas de mayor riesgo, concentrando su actividad en áreas alejadas del aro. En la [Figura 5.6a](#), se observa que los lanzamientos vuelven a realizarse mayoritariamente desde posiciones cercanas al punto de inicio, un patrón similar al de la versión 1. Este comportamiento sugiere que el agente lanza de forma anticipada para evitar enfrentarse a la defensa. Esta hipótesis se refuerza al observar que el jugador con balón ([Figura 5.6c](#)) permanece principalmente en la parte trasera del medio campo, reduciendo la profundidad de su avance. La presión defensiva parece forzar al agente a resolver las jugadas con rapidez con tiros más precipitados.

Los datos de la [Tabla 5.3](#) reflejan el impacto de esta presión defensiva. La recompensa media por episodio es negativa en las primeras fases del entrenamiento y apenas logra alcanzar un valor positivo de 0,02 tras 100000 iteraciones. Este rendimiento contrasta con versiones anteriores y se debe, en parte, a una penalización excesiva por intercepciones o a una reducción demasiado brusca de la probabilidad de acierto en presencia de defensores. Además, el porcentaje de tiros inmediatos aumenta considerablemente: del 15,9 % al 35,5 % a medida que avanzan las iteraciones. Este incremento confirma que el agente opta por lanzar cuanto antes para evitar la pérdida de balón. Esta tendencia también se refleja en la duración de los episodios, cuya mediana desciende a 2 pasos y la media a 3,22, lo que representa un fuerte descenso respecto a la versión 2.

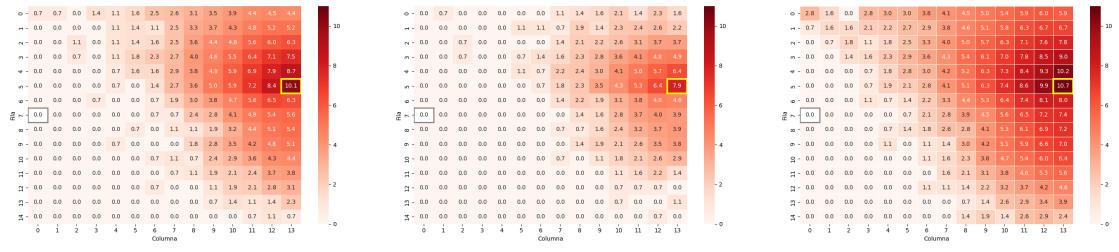


Figura 5.6: Distribución espacial de eventos durante los episodios en la versión 3

Iteraciones	Recompensa media	Recompensa máxima	% tiros inmediatos
500	-0,07	1,06	15,9 %
1000	-0,11	1,05	16,9 %
10000	-0,05	1,05	25,9 %
50000	-0,01	1,31	31,9 %
100000	0,02	1,34	35,5 %

Tabla 5.3: Resumen de métricas por iteraciones en la versión 3

En resumen, la presión defensiva condiciona fuertemente la toma de decisiones del agente, que adopta estrategias más conservadoras y apresuradas. Aunque logra reducir las intercepciones, lo hace a costa de renunciar a jugadas más elaboradas, lo que reduce su rendimiento y limita el uso efectivo del terreno de juego.

Finalmente, el **agente en la versión 4** demuestra un comportamiento más avanzado, reflejo de un entorno que incorpora roles tácticos diferenciados, lógica de tiro basada en datos reales y una mayor complejidad contextual. La distribución espacial de eventos mostrada en la [Figura 5.7](#) refleja un uso más equilibrado y eficiente del campo ofensivo. Los movimientos del jugador con balón se van aproximando al aro, lo que favorece la generación de líneas de pase efectivas. Además, los lanzamientos muestran una reducción de los tiros desde zonas traseras del campo respecto a versiones anteriores, que indica que el agente ha superado la limitación de lanzar desde posiciones fijas y considera las probabilidades de éxito asignadas a cada celda y a cada rol de jugador. Las intercepciones son mínimas y más dispersas, lo que sugiere una mejora en decisiones bajo presión. No obstante, este progreso también depende de la presión defensiva implementada, tal como se analizó en la versión previa. Si esta no fuese tan intensa, es probable que las diferencias con respecto a los mapas de calor de la versión 2 ([Figura 5.5](#)) fueran más pronunciadas.

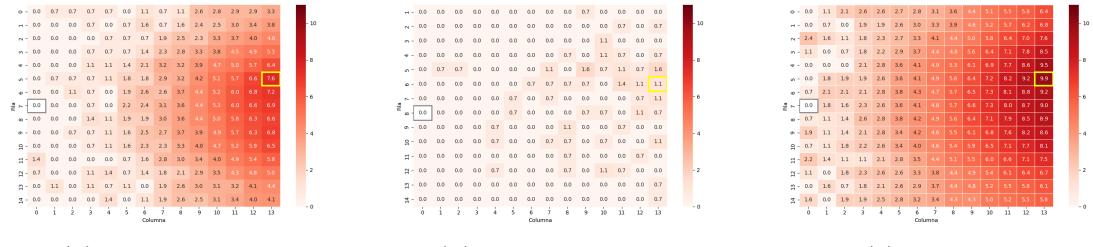


Figura 5.7: Distribución espacial de eventos durante los episodios en la versión 4

Los resultados presentados en la [Tabla 5.4](#) muestran un rendimiento superior respecto a todas las versiones anteriores. La recompensa media por episodio aumenta progresivamente con el entrenamiento. El porcentaje de lanzamientos inmediatos cae de forma contundente, del 18,2 % al 2,8 %, reflejando una clara preferencia por jugadas más elaboradas y contextualizadas.

Iteraciones	Recompensa media	Recompensa máxima	% tiros inmediatos
500	0,39	1,56	18,2 %
1000	0,31	1,56	13,0 %
10000	0,36	1,80	14,9 %
50000	0,49	3,00	3,9 %
100000	0,51	3,00	2,8 %

Tabla 5.4: Resumen de métricas por iteraciones en la versión 4

En cuanto a la duración de los episodios, se observa un incremento respecto a la versión 3: la mediana se sitúa en 7 pasos y la media en 9,23. Este aumento confirma que el agente aprende a mantener la posesión, colaborar entre jugadores con roles diferenciados y seleccionar el momento más oportuno para lanzar.

La [Figura 5.8](#) compara los porcentajes de acierto y fallo de los jugadores. Aunque los ejes verticales no son directamente comparables, la izquierda representa estadísticas reales y la derecha intentos acumulados en la simulación, se identifican patrones relevantes sobre cómo el agente ha interpretado el perfil de cada jugador.

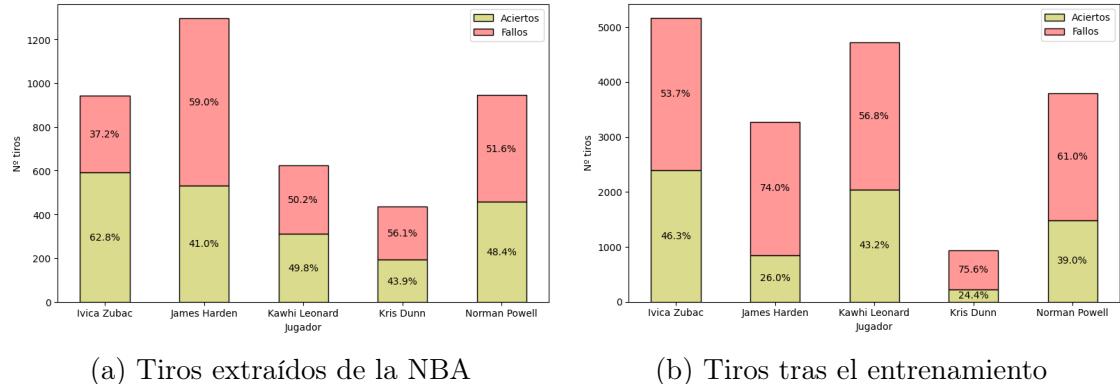


Figura 5.8: Porcentaje de aciertos y fallos en los tiros de cada jugador

Zubac, pivot con el mayor acierto real, es el jugador que más lanza tras el entrenamiento, pese a reducir su precisión al 46,3 %, lo que indica que el agente lo considera fiable. James Harden, el jugador con más volumen de tiro en los datos reales, pierde protagonismo debido a su bajo rendimiento simulado (26 %), lo que sugiere que el agente penaliza la ineficiencia. Leonard, con porcentajes equilibrados en la realidad, gana más peso ofensivo del esperado. Kris Dunn conserva su bajo volumen de lanzamientos en ambos contextos, reflejando una interpretación coherente de su papel como base. Norman Powell muestra una relación próxima entre realidad y simulación, señalando una adecuada adaptación del agente a su perfil.

En conjunto, el agente interioriza parcialmente los roles y eficiencias de cada jugador, ajustando el reparto de tiros. Pese a algunas desviaciones, la versión 4

muestra mayor nivel de sofisticación, con decisiones más estratégicas y un comportamiento ofensivo más realista gracias a la integración de roles y datos reales.

Para concluir esta evaluación experimental, se presenta a continuación un episodio correspondiente a la primera y a la última versión implementada. En el caso de la primera versión, se incluyen todos los pasos del episodio, mientras que en la última se muestran únicamente algunos pasos, debido a limitaciones de espacio. El objetivo es ilustrar visualmente las diferencias en el comportamiento del agente previamente analizadas y evidenciar la evolución lograda, partiendo en ambos casos desde la misma posición inicial de los jugadores (como se observa en [Figura 3.2](#)).

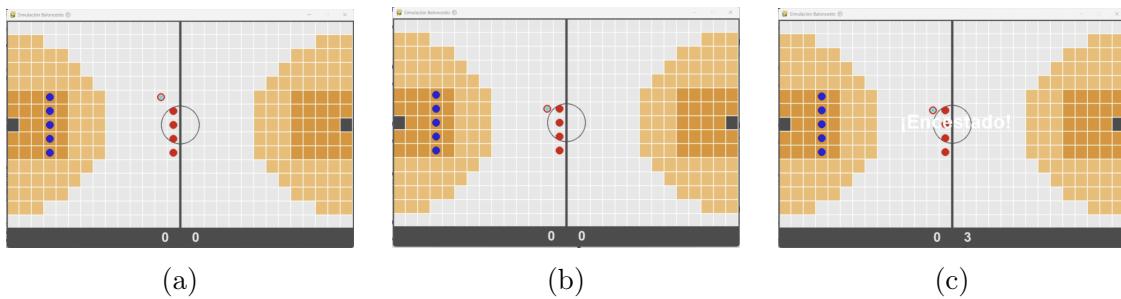


Figura 5.9: Representación de un episodio completo de la versión 1

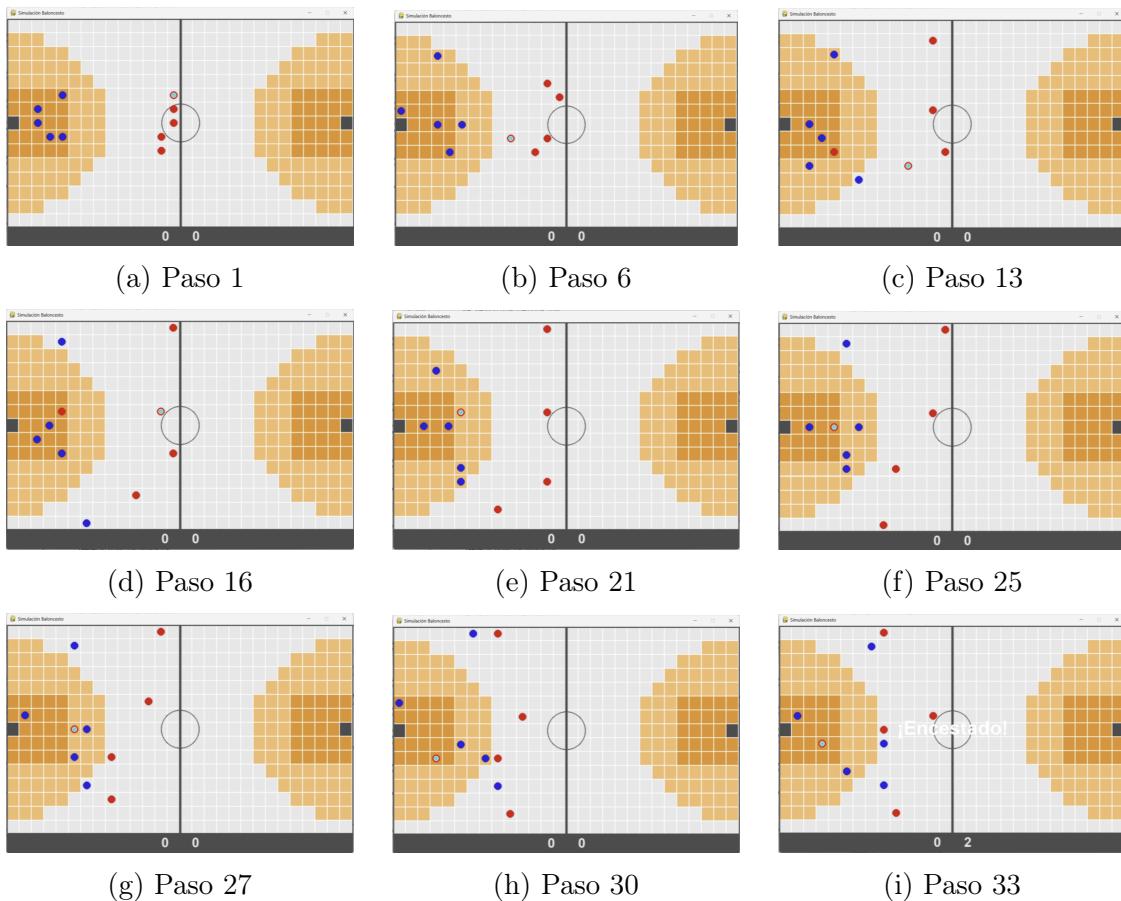


Figura 5.10: Representación de fragmentos de un episodio de la versión 4

Capítulo 6

Conclusiones y posibles ampliaciones

Este trabajo ha demostrado cómo el Aprendizaje Reforzado [5] puede emplearse para modelar y mejorar el comportamiento táctico en un entorno simulado de baloncesto. A través del desarrollo progresivo de cuatro versiones del entorno, se ha evidenciado una evolución clara en la toma de decisiones del agente, pasando de comportamientos impulsivos a estrategias más elaboradas y contextualizadas.

En la versión inicial, el agente aprendió rápidamente una política simplificada basada en lanzar desde la posición de inicio para maximizar la recompensa. Sin embargo, al incorporar nuevas capacidades como el pase, la presión defensiva y los roles diferenciados, el agente fue ajustando su comportamiento hacia esquemas más realistas y eficientes. La última versión, con integración de datos reales de jugadores de la NBA [25], muestra un notable aumento en la complejidad y eficacia de las decisiones tomadas, evidenciado por la mejora de las recompensas medias, la duración de los episodios y la reducción de lanzamientos inmediatos.

El entorno desarrollado, de estructura modular y visualización desacoplada, ha facilitado tanto la implementación como el análisis del comportamiento aprendido. No obstante, se han identificado limitaciones: el KNN simple [27] para completar celdas vacías en el mapeo de probabilidades de tiro podría mejorarse con técnicas de suavizado (KNN *smoothing*) [28], y la penalización por presión defensiva definida tal vez sea excesivamente brusca, afectando al rendimiento del agente.

Este trabajo abre diversas vías de ampliación. A nivel visual, se podrían sustituir el diseño de líneas rectas por curvas más representativas del campo real. A nivel técnico, convendría limitar los episodios a solo 24 pasos (uno por segundo), para tener una simulación más realista y coherente con las reglas básicas de baloncesto. En cuanto al entrenamiento, una mejora sería implementar un enfoque multiagente, con un agente por jugador, fomentando la coordinación y recompensas compartidas. También podría integrarse defensa con aprendizaje propio, generando una dinámica competitiva entre atacantes y defensores. Finalmente, escalar el entorno para simular partidos completos, con reinicios tras canasta, faltas o posesiones.

En definitiva, este trabajo establece una base sólida para seguir explorando la aplicación de IA en entornos deportivos, demostrando que es posible modelar decisiones complejas en entornos simulados accesibles, y que estas simulaciones pueden evolucionar hacia representaciones cada vez más cercanas al juego real.

Apéndice A

Entregables del proyecto

Todo el contenido desarrollado a lo largo del proyecto, incluyendo el código fuente, los entornos de simulación, los resultados obtenidos y la documentación técnica correspondiente a cada versión definida, se encuentra disponible en el siguiente repositorio:

https://github.com/letyml/proyecto_basket_rl

Este repositorio constituye el principal entregable técnico del trabajo y permite reproducir y analizar en detalle cada una de las versiones implementadas, así como evaluar el comportamiento del agente en los distintos escenarios propuestos.

Bibliografía

- [1] W. Qiang y Z. Zhongli, “Modelo de aprendizaje de refuerzo, algoritmos y su aplicación,” en *2011 Conferencia Internacional sobre Ciencia Mecatrónica, Ingeniería Eléctrica y Computación (MEC)*, 2011, págs. 1143-1146. DOI: <https://doi.org/10.1109/MEC.2011.6025669>.
- [2] GeeksforGeeks, *Reinforcement learning*, s.f. dirección: <https://www.geeksforgeeks.org/what-is-reinforcement-learning/> (visitado 19-03-2025).
- [3] FIBA, *Official basketball rules 2024 (v1.0a)*, 2024. dirección: <https://assets.fiba.basketball/image/upload/documents-corporate-fiba-official-rules-2024-v10a.pdf> (visitado 19-03-2025).
- [4] A. Hammoudeh, “A Concise Introduction to Reinforcement Learning,” *ResearchGate*, 2018. DOI: <https://doi.org/10.13140/RG.2.2.31027.53285>.
- [5] J. Achiam, *Spinning Up in Deep Reinforcement Learning*, 2018. dirección: https://spinningup.openai.com/en/latest/spinningup/rl_intro.html (visitado 17-06-2025).
- [6] GeeksforGeeks, *Deep Q-Learning*, s.f. dirección: <https://www.geeksforgeeks.org/deep-q-learning/> (visitado 24-06-2025).
- [7] GeeksforGeeks, *A Brief Introduction to Proximal Policy Optimization*, s.f. dirección: <https://www.geeksforgeeks.org/machine-learning/a-brief-introduction-to-proximal-policy-optimization/> (visitado 24-06-2025).
- [8] D. Silver, A. Huang, C. J. Maddison et al., “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, n.º 7587, págs. 484-489, 2016. DOI: <https://doi.org/10.1038/nature16961>.
- [9] D. Silver, T. Hubert, J. Schrittwieser et al., “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm,” *Science*, vol. 362, n.º 6419, págs. 1140-1144, 2017. DOI: <https://doi.org/10.1126/science.aar6404>.
- [10] G. Tesauro, “Temporal Difference Learning and TD-Gammon,” *Communications of the ACM*, vol. 38, n.º 3, págs. 58-68, 1995. DOI: <https://doi.org/10.1145/203330.203343>.
- [11] H. Jia, Y. Hu, Y. Chen et al., “Fever Basketball: A Complex, Flexible, and Asynchronized Sports Game Environment for Multi-agent Reinforcement Learning,” *ArXiv*, 2020. DOI: <https://doi.org/10.48550/arXiv.2012.03204>.

- [12] C. Yanai, A. Solomon, G. Katz et al., “Q-Ball: Modeling Basketball Games Using Deep Reinforcement Learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, n.º 8, págs. 8806-8813, 2022. DOI: <https://doi.org/10.1609/aaai.v36i8.20861>.
- [13] C. Truong, C. Ruffino, A. Crognier et al., “Error-based and reinforcement learning in basketball free throw shooting,” *Scientific Reports*, vol. 13, pág. 499, 2023. DOI: <https://doi.org/10.1038/s41598-022-26568-2>.
- [14] IBM, *IBM Watson: Using AI and Machine Learning for Business*, 2020. dirección: <https://www.ibm.com/watson> (visitado 06-04-2025).
- [15] Microsoft, *Microsoft Azure AI: Reinforcement Learning for Business Optimization*, 2021. dirección: <https://azure.microsoft.com/en-us/solutions/ai/> (visitado 06-04-2025).
- [16] OpenAI, *OpenAI Five: Reinforcement Learning for Competitive Games*, 2018. dirección: <https://openai.com/research/openai-five> (visitado 06-04-2025).
- [17] Netflix Technology Blog, *Reinforcement Learning for Budget Constrained Recommendations*, dic. de 2021. dirección: <https://netflixtechblog.com/reinforcement-learning-for-budget-constrained-recommendations-6cbc5263a32a> (visitado 06-04-2025).
- [18] InfoQ, *Netflix’s New Algorithm Offers Optimal Recommendation Lists for Users with Finite Time*, sep. de 2022. dirección: <https://www.infoq.com/news/2022/09/Netflix-optimal-recommendation/> (visitado 06-04-2025).
- [19] Waymo, *Imitation is not enough: Robustifying imitation with reinforcement learning*, s.f. dirección: <https://waymo.com/research/imitation-is-not-enough-robustifying-imitation-with-reinforcement-learning/>.
- [20] Farama Foundation, *Gymnasium: A modern standard for reinforcement learning environments*, 2025. dirección: <https://gymnasium.farama.org/> (visitado 09-06-2025).
- [21] A. Raffin, A. Hill, A. Gleave y pthers, “Stable-Baselines3: Reliable Reinforcement Learning Implementations,” *Journal of Machine Learning Research*, vol. 22, n.º 268, págs. 1-8, 2021. dirección: <http://jmlr.org/papers/v22/20-1364.html>.
- [22] GeeksforGeeks, *Pygame Tutorial*, s.f. dirección: <https://www.geeksforgeeks.org/python/pygame-tutorial/> (visitado 24-06-2025).
- [23] MLflow, *MLflow Documentation*, Documentación oficial del sistema de gestión del ciclo de vida del aprendizaje automático, 2025. dirección: <https://mlflow.org/docs/latest/ml/> (visitado 24-06-2025).
- [24] Stable-Baselines3, *DQN - Deep Q-Networks*, Documentación oficial de Stable-Baselines3, s.f. dirección: <https://stable-baselines3.readthedocs.io/en/master/modules/dqn.html> (visitado 24-06-2025).
- [25] P. Swar, Randall, T. Kelley and others, *nba_api: An API Client Package to access the NBA Stats API*, Repositorio oficial de acceso a datos estadísticos y jugadas de la NBA, 2024. dirección: https://github.com/swar/nba_api (visitado 09-06-2025).

- [26] Microsoft, *Data Wrangler*, Extensión oficial para limpieza y exploración de datos en Visual Studio Code, 2024. dirección: <https://marketplace.visualstudio.com/items?itemName=ms-toolsai.datawrangler> (visitado 24-06-2025).
- [27] IBM, *¿Qué es el algoritmo k-NN (k-Nearest Neighbors)?* s.f. dirección: <https://www.ibm.com/es-es/think/topics/knn> (visitado 24-06-2025).
- [28] Statistics Easily, *What is k-Nearest Neighbor Smoothing?* s.f. dirección: <https://statisticseasily.com/glossario/what-is-k-nearest-neighbor-smoothing/> (visitado 17-07-2025).