Snake: Implementació d'agents Intel·ligents

Cristian Pérez Díaz

Resum– Aquest article presenta un estudi sobre la implementació del joc Snake utilitzant Pygame. S'han implementat i avaluat dos agents basats en heurístiques: Greedy Best-First Search (GBFS) i A*. També s'ha investigat l'aprenentatge per reforç (RL) amb l'algoritme Proximal Policy Optimization (PPO) de Stable Baselines 3. Els agents s'han avaluat segons el seu rendiment en la puntuació.
L'estudi explora l'ús del RL en el joc Snake, adaptant l'entorn de joc a la llibreria Gymnasium i utilitzant l'algoritme PPO. Encara que l'agent RL no supera els agents heurístics, s'analitza i es discuteix el seu rendiment, destacant les dificultats i limitacions específiques en aquest escenari de joc. Aquesta recerca contribueix a comprendre diferents mètodes per abordar el joc Snake, posant de manifest els punts forts i les limitacions dels enfocaments heurístics i les dificultats associades a l'ús del RL en aquest context.

Paraules clau– Algorismes d'IA, Aprenentatge per reforçament, OpenAI, Gymnasium, Greedy Best-First Search, A*, Pygame, joc Snake, Stable Baselines 3.

Abstract– This article presents a study on the implementation of the Snake game using Pygame. Two heuristic-based agents, Greedy Best-First Search (GBFS) and A*, have been implemented and evaluated. The study also investigates reinforcement learning (RL) with the Proximal Policy Optimization (PPO) algorithm from Stable Baselines 3. The agents have been evaluated based on their performance in scoring. The study explores the use of RL in the Snake game by adapting the game environment to the Gymnasium library and utilizing the PPO algorithm. Although the RL agent does not outperform the heuristic-based agents, its performance is analyzed and discussed, highlighting the specific difficulties and limitations in this game scenario. This research contributes to understanding different methods for approaching the Snake game, revealing the strengths and limitations of heuristic approaches, as well as the challenges associated with RL in this context.

Keywords– AI algorithms, Reinforcement learning, OpenAI, Gymnasium, Greedy Best-First Search, A*, Pygame, Snake game, Stable Baselines 3.

1 Introducció - Context del treball

Aquest projecte ve motivat pel meu interès a aprendre a desenvolupar un videojoc i poder entendre i aplicar algorismes d'intel·ligència artificial capaços de jugar al videojoc. Les opcions entre els diferents videojocs són molt àmplies, però m'he decidit pel famós joc de l'Snake [1] ja que crec que em permetrà aprendre les nocions bàsiques de desenvolupar un videojoc, i serà molt fàcil de veure el progrés dels algorismes d'intel·ligència artificial aplicats.
La llibreria decidida per desenvolupar el videojoc és pygame [2], una llibreria de Python que proporciona una plataforma optimitzada i preparada per crear videojocs en Python.

El joc de l'Snake a implementar, és un videojoc individual que consisteix en controlar els moviments d'una serp dins d'una finestra delimitada mitjançant les fletxes del teclat de l'ordinador. L'objectiu principal és anar menjant les pomes que surten aleatòriament en pantalla, mentre que el cos de la serp va creixent a mesura que va menjant, i alhora evitant tocar els extrems de la pantalla i també evitant tocar el seu propi cos per no morir. La puntuació de la partida és definida per la quantitat de pomes menjades sense morir.
La idea és que un cop el videojoc estigui desenvolupat per tal que una persona pugui jugar, desenvolupar i aplicar dos algorismes d'intel·ligència artificial. El primer algorisme a fer és l'A* [3], l'he escollit ja que crec que és un bon algorisme per tal de trobar camins òptims a un punt i que en aquest cas seria la poma que ha de menjar la serp en cada moment. El segon algorisme a desenvolupar seria amb aprenentatge amb reforçament.

• E-mail de contacte: cristianperezd0@gmail.com
• Menció realitzada: Enginyeria del Software
• Treball tutoritzat per: Maria Vanrell Martorell (Departament de Ciències de la Computació)
• Curs 2022/23

2 OBJECTIUS

Per tal d'aconseguir el que he proposat a l'apartat anterior, els objectius són els següents:

- **Objectiu 1:** Entendre les bases i l'entorn de la llibreria Pygame.
- **Objectiu 2:** Implementar el joc de l'Snake per un agent humà amb Pygame.
- **Objectiu 3:** Implementar l'algorisme clàssic A* per l'Snake creat.
- **Objectiu 4:** Implementar un algorisme basat en aprenentatge per reforçament per l'Snake creat.
- **Objectiu 5:** Anàlisi i comparació dels dos algorismes en el joc de l'Snake.

3 ESTAT DE L'ART

En aquest apartat, es revisarà l'estat actual de la tecnologia i les eines relacionades amb el desenvolupament i implementació d'algorismes de resolució del joc Snake. El joc Snake es classifica dins dels problemes de cerca de camins, que involucra trobar una ruta òptima per arribar a un destí determinat. A més, s'han desenvolupat altres videojocs similars que comparteixen aquesta característica, com ara Pac-Man, on el jugador ha de guiar un personatge a través d'un laberint per menjar punts mentre evita els enemics.

En l'actualitat, s'utilitzen diversos algoritmes per resoldre problemes de cerca de camins com el joc Snake. Entre els algoritmes 'clàssics' de cerca informada destaquen el GBFS (Greedy Best-First Search) i l'A* (A-star). Aquests algoritmes utilitzen una heurística per avaluar les possibles rutes i trobar camins òptims en un espai de cerca. El GBFS utilitza una heurística per crear l'arbre, mentre que l'A* utilitza l'heurística juntament amb un cost acumulat per cada pas.
Una altra tendència en el desenvolupament d'algorismes de resolució de jocs com l'Snake és l'ús d'algoritmes de reinforcement learning (aprenentatge per reforç). Aquests algoritmes permeten que un agent (en aquest cas, la serp del joc) aprengui de forma autònoma a prendre decisions per aconseguir els objectius del joc.

Dins de l'àmbit del reinforcement learning, destaca l'ús de l'OpenAI Gym (actualment Gymnasium) [4], una biblioteca mantinguda per la Farama Foundation [5]. L'OpenAI Gym ofereix una plataforma flexible per al desenvolupament d'entorns de videojocs i l'aplicació d'algoritmes de reinforcement learning. Permet als desenvolupadors definir nous entorns de simulació i establir regles per a l'interacció entre l'agent i l'entorn. A més, proporciona una àmplia gamma d'eines i entorns predefinits per a l'aprenentatge automàtic, facilitant així la implementació de solucions de reinforcement learning en diferents videojocs com l'Snake en aquest cas.
També, una eina destacada dins del reinforcement learning és la biblioteca Python Stable-Baselines3 [6]. És una biblioteca open-source que implementa algoritmes de reinforcement learning com ara A2C (Advantage Actor-Critic), PPO (Proximal Policy Optimization), SAC (Soft Actor-Critic) i DQN (Deep Q-Network), entre d'altres. A més, ofereix compatibilitat per utilitzar aquests algoritmes amb els entorns de Gymnasium (OpenAI Gym).
Aquesta informació serveix com a base per a l'elecció i justificació de les solucions utilitzades en el treball i proporciona una visió general de les tecnologies actuals relacionades amb el desenvolupament del joc Snake.

4 SNAKE

En aquest apartat explicaré el funcionament, regles i algorismes del joc Snake que serviran com a base per entendre i tenir una visió clara del projecte.

4.1 Regles del Joc

Les regles del joc són les següents:

- **Inici del joc:** El joc comença amb una serp de longitud prefixada en el mig d'una àrea de joc limitada. També es genera de forma aleatòria una poma dins d'aquesta àrea de joc.
- **Moviment de la serp:** El jugador o agent controla la direcció de la serp a partir de 4 direccions, que normalment s'assignen a les fletxes del teclat. La serp es mou de forma contínua a partir de la darrera direcció proporcionada per al jugador o agent.
- **Col·lisió amb els extrems i amb la cua:** Si el cap de la serp xoca amb la seva pròpia cua o amb els extrems de l'àrea del joc, el joc acabarà.
- **Puntuació:** La puntuació del joc és definida per al nombre de pomes menjades, on a partir de cada poma consumida s'augmenta la longitud de la cua en una unitat.
- **Objectiu del joc:** L'objectiu principal del joc és menjar el nombre màxim de pomes possibles sense morir.

4.2 Agent Basat en Cerca

Durant el desenvolupament del projecte, es va prendre la decisió de començar implementant l'algorisme GBFS (Greedy Best-First Search) com a primera opció d'algorisme de cerca, tot i que l'objectiu principal era implementar l'algorisme A*.
L'algorisme GBFS [7] té com a objectiu trobar una solució que eviti que la serp col·lidi amb la seva pròpia cua i amb les parets del taulell, així com trobar la posició del següent menjar. Utilitza una heurística basada en la distància de manhattan per determinar quins nodes explorar en primer lloc, prioritzant aquells que semblen més prometedors.
Posteriorment, es va decidir implementar també l'algorisme A* com a alternativa. La principal diferència entre GBFS i A* és la manera en què es calcula el cost d'un node. Mentre GBFS es basa únicament en la heurística per seleccionar el millor node següent, A* té en compte tant la heurística com el cost acumulat fins a aquest punt. Això permet a A* buscar una solució òptima en termes de cost, tenint en compte la distància fins al menjar i el nombre de passos realitzats [8].

4.3 Agent basat en Reinforcement Learning

En el desenvolupament d'aquest projecte, s'ha investigat l'aplicació de tècniques d'aprenentatge per reforçament per entrenar l'agent de la serp. L'aprenentatge per reforçament és una branca de l'aprenentatge automàtic que se centra en com un agent pot aprendre a prendre decisions òptimes mitjançant la interacció amb un entorn. Es basa en el concepte de recompenses i penalitzacions per guiar el comportament de l'agent cap a una meta desitjada [9].

Si entrem en més detall, l'aprenentatge per reforçament es basa en un procés de presa de decisions en el qual un agent interactua amb el seu entorn. Agafarem la descripció matemàtica que es fa a [10] com a base per definir-lo aquí. En aquest procés, es defineixen els següents elements:

- Un conjunt d'estats de l'entorn i de l'agent, denotat com S.
- Un conjunt d'accions disponibles per a l'agent, denotat com A.
- La probabilitat de transició $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ indica la probabilitat de passar de l'estat s a l'estat s' en prendre l'acció a en el temps t.
- La recompensa immediata $R_a(s, s')$ s'obté després de la transició de l'estat s a l'estat s' en prendre l'acció a.

En cada pas discret de temps t, l'agent interactua amb el seu entorn. Rep l'estat actual s_t i la recompensa r_t. A partir d'això, selecciona una acció a_t d'un conjunt d'accions disponibles i l'envia a l'entorn. Com a resultat, l'entorn passa a un nou estat s_{t+1} i proporciona una recompensa r_{t+1} associada a la transició (s_t, a_t, s_{t+1}).
L'objectiu de l'agent d'aprenentatge per reforç és aprendre una política $\pi : A \times S \rightarrow [0, 1]$, on $\pi(a, s) = \Pr(a_t = a \mid s_t = s)$, que maximitzi la recompensa acumulativa esperada.
En altres paraules, l'agent cerca trobar una estratègia òptima per seleccionar accions en funció dels estats actuals, maximitzant la suma de les recompenses a llarg termini.

Primer de tot, es va prendre la decisió d'adaptar el joc Snake ja creat a un entorn de Gymnasium. Això va permetre treballar amb el meu joc d'una manera estandarditzada, tal com ho fa la gran majoria de la comunitat d'aprenentatge per reforçament en els videojocs. A més, això em va permetre trobar més referències i exemples de qualitat que em van ser de gran d'utilitat.
Seguidament es va decidir utilitzar un algorisme d'aprenentatge per reforçament ja implementat a partir dels que ofereix la biblioteca Stable-Baselines3. Aquesta elecció es va basar en el fet que les implementacions d'aquests algorismes eren compatibles amb els entorns de Gymnasium, la qual cosa va permetre estalviar una gran quantitat de temps.

Dins de les tècniques d'aprenentatge per reforçament més conegudes és el mètode Q-learning. Aquest algoritme utilitza una taula de valors (Q-values) per representar la utilitat esperada de cada acció en un estat donat. Mitjançant l'exploració i l'actualització iterativa dels Q-values, l'agent aprèn a prendre les accions que maximitzen les recompenses a llarg termini [11].
En el context de l'aprenentatge profund, el Deep Q-learning (DQN) combina l'enfocament del Q-learning amb xarxes neuronals profundes. En lloc d'utilitzar una taula de valors, s'entrena una xarxa neuronal per aproximar els Q-values. Això permet abordar problemes més complexos i de major dimensionalitat. El DQN ha demostrat ser efectiu en una àmplia gamma d'aplicacions, inclosos els videojocs[12].

Tanmateix, al final del desenvolupament, es va decidir utilitzar l'algoritme Proximal Policy Optimization (PPO) per entrenar l'agent en l'entorn de la serp adaptat a través de Gymnasium. La elecció de PPO es va basar en la seva facilitat d'ús i en la capacitat que va proporcionar per calibrar les recompenses i observacions de l'entorn.
L'objectiu de l'algoritme PPO és entrenar un agent d'aprenentatge per reforç perquè aprengui una política òptima en un entorn determinat [13]. Una política és una estratègia que determina quina acció prendre en cada estat per maximitzar la recompensa acumulada.
L'enfocament principal de PPO és millorar i actualitzar la política de manera iterativa utilitzant mètodes basats en gradient. L'algoritme cerca trobar l'equilibri adequat entre l'aprenentatge i l'estabilitat de la política, evitant canvis dràstics que puguin conduir a un mal rendiment.
PPO utilitza una tècnica anomenada optimització de polítiques pròximes, que es basa en la idea que les actualitzacions de la política han de ser limitades per evitar canvis massa grans. Això s'aconsegueix mitjançant la maximització d'una funció objectiu que té en compte la diferència entre la política actual i la política actualitzada.
En resum, PPO és un algoritme d'aprenentatge per reforç que busca trobar una política òptima de manera iterativa, utilitzant mètodes basats en gradient i optimització de polítiques pròximes. L'objectiu és millorar la política de manera estable i evitar canvis dràstics que puguin afectar negativament el rendiment de l'agent en l'entorn.

5 METODOLOGIA

El desenvolupament del videojoc es farà en Python, ja que proporciona una gran quantitat de llibreries per fer aquesta tasca, i en aquest cas es decideix utilitzar Pygame.
Aquesta llibreria proporciona una àmplia documentació i tutorials, que seran molt útils per poder entendre i desenvolupar un codi de qualitat.
Respecte a la metodologia de desenvolupament del projecte, s'ha de tenir en compte que és un projecte individual i que, per tant, seguir una metodologia concreta no tindria gaire sentit, en conseqüència, adaptaré la metodologia Scrum que he fet servir anteriorment en altres projectes acadèmics grupals per a poder seguir la mateixa filosofia de forma individual, excloent els aspectes més grupals. Faré ús l'eina Azure per definir els Sprints i les tasques corresponents en cada sprint, ja que proporciona un taulell per visualitzar i fer un seguiment del progrés de les tasques.
Els sprints tindran una durada aproximada d'un mes, ja que el seu inici i final coincideixen amb les dates d'entrega dels informes. En cas que les tasques d'un sprint no s'aconsegueixin complir, es traslladarien al següent sprint per tal de finalitzar-les.

6 PLANIFICACIÓ

Per tal de poder dur a terme el projecte, seguiré la següent planificació, per veure amb més detall consultar annex:

1. Definició del projecte
    1.1. Revisió d'informació
    1.2. Definició de l'abast
2. Preparar informe inicial
- Entrega Informe Inicial
3. Revisió llibreria Pygame
4. Disseny del sistema inicial
5. Creació repositori
6. Creació Snake amb Control Humà
7. Preparar Informe de Progrés 1
- Entrega Informe de Progrés 1
8. Estudiar Algorisme A*
9. Implementar Algorisme A*
10. Estudiar Algorisme de Reinforcement Learning
11. Implementar Algorisme Reinforcement Learning
12. Preparar Informe de Progrés 2
- Entrega Informe de Progrés 2
13. Anàlisi i comparació dels algorismes
14. Preparar Informe Final
15. Preparar Presentació
16. Preparar Dossier

Tasca	Data Inicial	Estimació
1. Definició del projecte	06/02/2023	21 dies
1.1. Revisió d'informació	06/02/2023	16 dies
1.2. Definició de l'abast	27/02/2023	6 dies
2. Preparar informe inicial	06/03/2023	6 dies
Entrega informe inicial - 12/03/2023		
3. Revisió llibreria Pygame	13/03/2023	11 dies
4. Disseny del sistema inicial	13/03/2023	11 dies
5. Creació repositori	13/03/2023	2 dies
6. Creació Snake amb Control Humà	27/03/2023	16 dies
7. Preparar Informe de Progrés 1	10/04/2023	11 dies
8. Estudiar Algorisme A*	10/04/2023	11 dies
Entrega informe de Progrés 1 - 23/04/2023		
9. Implementar Algorisme A*	25/04/2023	16 dies
10. Estudiar Algorisme de Reinforcement Learning	24/04/2023	16 dies
11. Implementar Algorisme Reinforcement Learning	15/05/2023	11 dies
12. Preparar Informe de Progrés 2	15/05/2023	11 dies
Entrega informe de Progrés 2 - 28/05/2023		
13. Anàlisi i comparació dels resultats	30/05/2023	7 dies
14. Preparar Informe Final	29/05/2023	16 dies
Entrega Informe Final - 18/06/2023		
15. Preparar Presentació	19/06/2023	10 dies
16. Preparar Dossier	19/06/2023	11 dies
Entrega Proposta de Presentació - 30/06/2023		
Entrega Dossier del TFG - 02/07/2023		

TAULA 1: PLANIFICACIÓ DETALLADA

7 DESENVOLUPAMENT

A continuació, s'explica la implementació inicial del joc Snake, les implementacions dels algorismes de cerca i l'agent basat en reinforcement learning:

7.1 Base del joc i Agent Huma

En aquesta secció desciuré les classes creades per tal de desenvolupar el joc:

- Classe Snake: Aquesta classe s'encarrega de controlar la serp en el joc Snake. Conté mètodes que permeten moure la serp en diferents direccions, comprovar col·lisions i dibuixar la serp en la pantalla del joc. La classe té tres atributs principals: la direcció en què es mou la serp, el cap de la serp i el seu cos, que és una llista de segments que conformen la serp.
- Classe Food: Aquesta classe s'encarrega de generar i controlar el menjar en el joc Snake. Conté mètodes que permeten generar el menjar aleatòriament en una posició buida a la pantalla del joc, comprovar si el menjar

ha estat tocat per la serp i dibuixar el menjar a la pantalla. L'atribut principal d'aquesta classe és la posició del menjar a la pantalla.

- Classe Game: Aquesta classe és l'encarregada de controlar la lògica del joc Snake. Conté mètodes que permeten dibuixar la pantalla del joc, actualitzar la posició de la serp i el menjar, detectar col·lisions i actualitzar la puntuació del joc. Els atributs principals d'aquesta classe són l'amplada i alçada de la pantalla, la pantalla del joc en si, el rellotge utilitzat per controlar la velocitat del joc, la puntuació del jugador, la serp i el menjar.

Un cop creat l'esquelet bàsic que permet a un jugador humà jugar, s'ha de començar a reestructurar el codi per permetre als agents d'intel·ligència artificial jugar al videojoc. Caldrà crear un mètode per generar una matriu amb la informació del tauler per permetre als agents accedir-hi de manera senzilla.

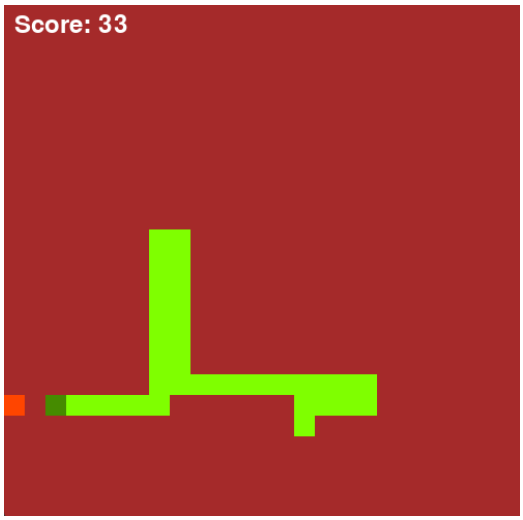

Fig. 1: Joc Snake implementat

La gestió de l'agent humà es fa mitjançant les 4 fletxes del teclat per decidir els moviments de la serp. La detecció d'aquestes accions de l'agent es realitza dins del bucle del joc a partir de la classe game.

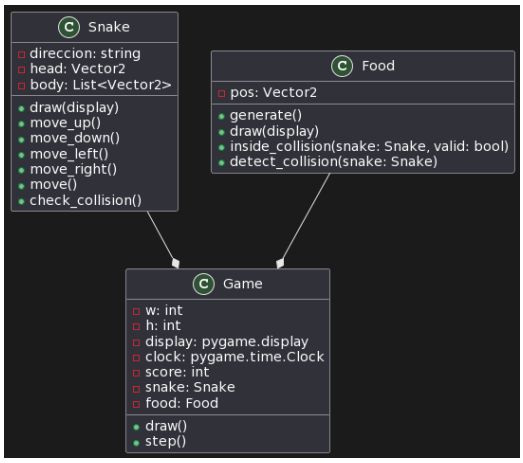

Fig. 2: Diagrama de Classes Snake

7.2 Agent Basat en Cerca

Per poder implementar els dos algoritmes de cerca (GBFS i A*), es van crear diverses classes esencials:

- **Node**: Aquesta classe representa un node en l'espai de cerca i conté la informació necessària per a la cerca. Els atributs específics de la classe no són mencionats per brevetat, però inclouen el taulell actual, la serp, el menjar, l'heurística i el cost acumulat. Aquesta classe permet la construcció de l'arbre de cerca mitjançant referències als nodes pare i fills.
- **GBFSNode**: És una subclasse de **Node** i s'utilitza específicament per a l'algorisme GBFS. A més dels atributs heretats de la classe **Node**, implementa el mètode de comparació 'lt',que ordena els nodes en funció de l'heurística. Aquest mètode permet ordenar els nodes en funció del seu valor heurístic, prioritzant aquells que semblen més propers a l'objectiu.
- **AStarNode**: És una subclasse de **Node** i s'utilitza específicament per a l'algorisme A*. La diferència principal és la implementació del mètode de comparació 'lt', que ordena els nodes en funció del cost acumulat (cost + heurística). Això assegura que els nodes més prometedors tinguin prioritat en la cerca.
- **GBFS**: Aquesta classe és responsable de l'execució de l'algorisme GBFS. Conté el node inicial, el node objectiu, l'estat actual, la solució trobada i un comptador per controlar el progrés de la cerca. La cerca s'efectua utilitzant una cua de prioritat, on els nodes s'ordenen en funció de l'heurística, prioritzant aquells que semblen més propers al menjar.
- **AStar**: És similar a **GBFS**, però s'utilitza per a l'algorisme A*. Conté el node inicial, el node objectiu, l'estat actual, la solució trobada i un comptador. La cerca s'efectua utilitzant una cua de prioritat, on els nodes s'ordenen en funció del cost acumulat (cost + heurística), prioritzant així els nodes amb menor cost total fins al moment.

Aquestes classes, juntament amb les seves funcionalitats específiques, controlen el funcionament de l'algorisme de cerca en el context del videojoc de la serp. Amb l'ús d'aquests algorismes, es pot explorar i trobar rutes òptimes o prometedores per a la serp, evitant col·lisions amb obstacles i aconseguint els objectius establerts.

7.3 Aprenentatge per Reforçament

Per poder aplicar tècniques d'aprenentatge per reforçament al nostre joc Snake, primerament es va adaptar el joc creat a l'entorn Gymnasium. Aquesta adaptació va requerir la creació d'una nova classe anomenada SnakeEnv, que hereta de la classe base proporcionada per Gymnasium.
Dins de la classe SnakeEnv, es van definir diversos mètodes essencials per a l'interacció entre l'agent d'aprenentatge per reforçament i l'entorn del joc:

- **Init**: Aquest mètode s'encarrega d'inicialitzar la classe SnakeEnv. En aquest mètode es defineix el nombre d'accions possibles per l'agent i el tipus d'espai d'observació.

- **Step**: Aquest mètode inclou la funcionalitat bàsica d'un joc, equivalent a un frame. Verifica la direcció presa per l'agent, mou la serp i comprova si hi ha hagut col·lisions o si s'ha menjat el menjar. A més, es defineixen les recompenses i observacions que utilitzarà l'agent d'aprenentatge per reforçament.
- **Reset**: Aquest mètode permet reiniciar el joc a l'estat inicial.
- **Render**: Aquest mètode s'utilitza per visualitzar el joc, de manera que es pot observar en temps real el funcionament de l'algoritme en acció.

Posteriorment, es va definir el sistema de recompenses per al joc Snake. Inicialment, l'agent rebia una recompensa constant de 10 al menjar una peça de menjar i una recompensa de -10 al morir. No obstant això, es va observar que entrenant el model d'aquesta manera, l'agent només aprenia a menjar una única unitat de menjar i a girar per evitar morir mai. Això va portar a la decisió de modificar el sistema de recompenses per acumular una recompensa positiva cada vegada que es menjava una peça de menjar, incentivant així l'agent a menjar més d'una unitat de menjar.

Pel que fa a l'observació rebuda per part de l'agent, es va prendre la decisió de proporcionar una observació simple ja que el mateix agent no seria capaç d'entendre directament la matriu del tauler del joc. En lloc d'això, es va passar a l'agent la direcció relativa de la peça de menjar respecte al cap de la serp, la direcció actual en què es mou la serp i si hi ha perill o no en cada direcció del cap de la serp.
Aquesta simplificació de l'observació permet que l'agent pugui prendre decisions basades en informació rellevant i no es vegi sobrecarregat amb detalls innecessaris del tauler del joc.

8 RESULTATS

En aquest apartat es mostraran i compararan els resultats dels algoritmes utilitzats en el joc Snake.

8.1 Resultats de GBFS i A*

Durant l'avaluació dels algoritmes GBFS i A* en la resolució del joc de la serp, es van realitzar 50 simulacions per analitzar el seu rendiment. Es va establir un límit màxim de 2000 nodes expandits per cerca.

Es van registrar les puntuacions obtingudes en cada simulació per comparar el rendiment de GBFS i A*. Els resultats es mostren en la següent taula:

Algoritme	Mitjana	Rang (Mínim - Màxim)
GBFS	42.94	23 - 72
A*	78.34	36 - 136

TAULA 2: RESULTATS DE GBFS Y A* EN LES SIMULACIONS

A la taula es pot observar que GBFS va obtenir una puntuació mitjana de 42.94, amb un rang de 23 com a mínim i 72 com a màxim. D'altra banda, A* va aconseguir una puntuació mitjana de 78.34, amb un rang de 36 com a mínim i 136 com a màxim.
Comparant els números, es pot observar una clara millora en el rendiment de l'A* en comparació amb el GBFS. La puntuació mitjana de l'A* (78.34) és considerablement superior a la del GBFS (42.94), la qual cosa indica un millor rendiment en la cerca de solucions.
Per explicar aquesta diferència de resultats, cal tenir en compte les causes de la mort de la serp en aquests casos. En el GBFS, la serp pot morir en ocasions per perdre's buscant un camí i arribar al nombre màxim de nodes explorats. A més, també hi ha situacions en les quals la mort és inevitable a causa de l'atzar en l'aparició del menjar, portant el joc a un estat sense solució.
En canvi, en l'A* s'evita arribar al nombre màxim de nodes explorats gràcies al cost afegit i l'eliminació de camins redundants. Això permet explorar nous camins que el GBFS no arribaria a explorar. No obstant això, l'A* també pot enfrontar situacions en les quals, a causa de l'atzar en l'aparició del menjar, arribi a un estat del joc sense solució i acabi amb la mort de la serp.
En resum, l'A* mostra un millor rendiment que el GBFS en termes de puntuació mitjana i capacitat per explorar nous camins. No obstant això, totes dues estratègies poden trobar-se amb situacions de mort inevitables a causa de l'atzar en l'aparició del menjar.

8.2 Resultats de Reinfocement Learning

Durant l'avaluació de l'algorisme PPO, es van realitzar 50 simulacions utilitzant les dues versions entrenades: la primera versió amb les recompenses inicials i la segona versió amb les recompenses millorades.

Es van registrar les puntuacions obtingudes en cada simulació per comparar el rendiment de les dues versions. Els resultats es mostren a continuació en la següent taula:

Versió	Mitjana	Rang (Mínim - Màxim)
1	11.54	1 - 33
2	20.9	3 - 42

TAULA 3: RESULTATS DE PPO EN LES DUES VERSIONS

A la taula es pot observar com amb la segona versió s'obté aproximadament el doble de puntuació de mitjana, gràcies a les recompenses millorades. Això demostra que les modificacions en les recompenses han tingut un impacte positiu en el rendiment de l'algorisme PPO.

8.3 Comparativa final

En la comparativa final, s'observa que els agents basats en cerca, com GBFS i A*, han mostrat resultats superiors en termes de puntuació en el joc Snake. GBFS va obtenir una puntuació mitjana de 42.94, mentre que A* va aconseguir una puntuació mitjana de 78.34.
D'altra banda, l'agent de Reinforcement Learning amb PPO ha mostrat resultats menys satisfactoris en comparació amb els agents basats en cerca. En les 50 simulacions realitza-

des, la primera versió de l'agent de RL va obtenir una puntuació mitjana de 11.54, mentre que la segona versió, amb recompenses millorades, va assolir una puntuació mitjana de 20.9.
És important tenir en compte que l'entrenament d'un agent de Reinforcement Learning comporta certes dificultats. Primer, cal dissenyar adequadament les recompenses per a l'agent, la qual cosa pot ser un desafiament en si mateix. A més, és necessari implementar les observacions que l'agent utilitzarà per aprendre i prendre decisions.
És rellevant destacar que, malgrat els agents basats en cerca, com GBFS i A*, han mostrat millors resultats en el joc Snake, l'entrenament d'un agent de Reinforcement Learning implica reptes addicionals en termes de disseny de recompenses i observacions.
Amb més investigació, documentació i temps dedicats, és possible millorar els resultats de l'agent de Reinforcement Learning en futurs treballs. Amb un enfocament més profund en el disseny de recompenses adequades i la selecció d'observacions rellevants, és probable que s'obtinguin millores significatives en el rendiment dels agents de RL.

9 CONCLUSIONS

Durant el desenvolupament del joc Snake utilitzant Pygame, he après els conceptes clau necessaris per crear un joc amb aquesta biblioteca. L'experiència m'ha permès familiaritzar-me amb les funcions i característiques essencials de Pygame per al desenvolupament de jocs interactius.

Un aspecte fonamental que he après és la importància d'estructurar el codi adequadament des del principi. Tenir una estructura clara i ben organitzada facilita el progrés i l'ampliació del codi, evitant problemes a mesura que avancem en el desenvolupament. Aquest enfocament ha estat clau per mantenir el codi ordenat i afegir noves funcionalitats a mesura que el projecte ha evolucionat.

Un altre aspecte important és la necessitat d'una planificació adequada i un marc de referència clar. Establir objectius i metes específiques, així com un calendari de treball, permet seguir un pla i aprofitar eficientment el temps disponible. Això ha contribuït a mantenir un progrés constant en el desenvolupament del joc i a adaptar-me a les necessitats del projecte.

També vull destacar la importància de les llibreries open-source en l'àmbit del reinforcement learning. Aquestes llibreries proporcionen una base comuna per a tothom que vulgui aventurar-se en aquest camp, independentment de la seva experiència prèvia. Són eines valuoses que amplien l'accés a les tècniques de reinforcement learning i faciliten el desenvolupament de nous algorismes i aplicacions.
Gràcies a les llibreries open-source com l'OpenAI Gym i la biblioteca Stable-Baselines3, els desenvolupadors tenen a la seva disposició una amplia gamma d'eines, algorismes implementats i exemples de codi. Això permet accelerar el procés d'aprenentatge i reduir la barrera d'entrada per a aquells que volen explorar el món del reinforcement learning.

Per últim, durant el desenvolupament d'aquest projecte, he tingut l'oportunitat d'aprofundir en la implementació dels algoritmes GBFS i A* i entendre millor com funcionen. Implementar aquests algoritmes m'ha proporcionat una visió més clara dels aspectes clau de la cerca i la presa de decisions en entorns complexos. A més, en el context de l'aprenentatge per reforç, he après la importància fonamental de la definició i el disseny adequats de les recompenses, així com de les observacions que rep l'agent. Aquests aspectes són essencials per garantir un aprenentatge efectiu i un comportament òptim de l'agent en tasques d'intel·ligència artificial.

REFERÈNCIES

[1] Wikipedia contributors. Snake (video game genre). Recuperat 8 de març de 2023, Wikipedia. https://en.wikipedia.org/wiki/Snake_(video_game_genre)

[2] Kim, Y. (s. f.). Pygame v2.1.4 documentation. Pygame. Recuperat 8 de març de 2023, de https://www.pygame.org/docs/

[3] Ably, T., Pang, H., Tiliksew, B., Moore, K., Khlm, J. (s. f.-b). A* Search. Brilliant. Recuperat 8 de març de 2023, de https://brilliant.org/wiki/a-star-search/

[4] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W. (2016). Openai gym. arXiv preprint arXiv:1606.01540.

[5] Farama Foundation. (2022, 25 octubre). Announcing The Farama Foundation. Farama. Recuperat 18 de juny de 2023, de https://farama.org/Announcing-The-Farama-Foundation

[6] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. The Journal of Machine Learning Research, 22(1), 12348-12355.

[7] Vanrell, Maria. Apunts d'Intel·ligència Artifical. Grau d'Enginyeria Informàtica UAB, 2022.

[8] Gajjar, N. (2018). Automated Snake Game Solvers using AI Search Algorithms. GitHub. Recuperat 17 d'abril de 2023, de https://github.com/neelgajjar/Snake-game-AI-Solver

[9] Simonini, T. (s.f.). An introduction to reinforcement learning. freeCodeCamp. Recuperat 26 de maig de 2023, de https://www.freecodecamp.org/news/an-introduction-to-reinforcement-learning-4339519de419

[10] Wikipedia contributors. Reinforcement Learning.Recuperat 30 de juny de 2023, Wikipedia. https://en.wikipedia.org/wiki/Reinforcement_learning

[11] Simonini, T. (s. f.). Diving deeper into Reinforcement Learning with Q-Learning. freeCodeCamp. Recuperat 26 de maig de 2023,

de https://www.freecodecamp.org/news/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe/

[12] Simonini, T. (s. f.). An introduction to Deep Q-Learning: let's play Doom. freeCodeCamp. Recuperado 26 de maig de 2023, de https://www.freecodecamp.org/news/an-introduction-to-deep-q-learning-lets-play-doom-54d02d8017d8/

[13] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[14] Arnaus Comellas, Martí; Vanrell i Martorell, Maria Isabel, dir. Jugador d'un joc de cartes interactiu. 2023. (958 Enginyeria Informàtica) https://ddd.uab.cat/record/272822 [Consulta: 12 març 2023].

[15] Culhane, T. (s. f.). Reinforcement Learning on Hearts. Stanford University. http://cs230.stanford.edu/projects_spring_2021/reports/9.pdf

[16] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. arXiv (Cornell University). http://cs.nyu.edu/ koray/publis/mnih-atari-2013.pdf

[17] Pons, P. (s. f.). SOLVING CONNECT 4: HOW TO BUILD A PERFECT AI. blog.gamesolver. Recuperat 26 de febrer de 2023, de http://blog.gamesolver.org/

ANNEXOS

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1 Definició del project	**21 días?**	**lun 06/02/23**	**dom 05/03/2**	
1.1 Revisió d'informació	16 días	lun 06/02/23	dom 26/02/23	
1.2 Definició de l'ab	6 días	lun 27/02/23	dom 05/03/2	2
2 Preparar informe inicial	6 días	lun 06/03/23	dom 12/03/23	1
3 Revisió llibreria Pygame	11 días	lun 13/03/23	dom 26/03/23	
4 Disseny del sistema inicial	11 días	lun 13/03/23	dom 26/03/23	
5 Creació repositori	2 días	lun 13/03/23	mar 14/03/2	
6 Creació Snake amb Control Humà	16 días	lun 27/03/23	dom 16/04/23	5;6;7
7 Preparar Informe de Progrés 1	11 días	lun 10/04/23	dom 23/04/23	
8 Estudiar Algorisme A estrella	11 días	lun 10/04/23	lun 24/04/23	
9 Implementar Algorisme A estrella	16 días	mar 25/04/23	mar 16/05/23	8;10
10 Estudiar Algorisme de Reinforcement Learning	16 días	lun 24/04/23	dom 14/05/23	
11 Implementar Algorisme Reinforcement Learning	11 días	lun 15/05/23	lun 29/05/23	8;12
12 Preparar Informe de Progrés 2	11 días	lun 15/05/23	dom 28/05/23	
13 Anàlisi i comparació dels	7 días	mar 30/05/23	mié 07/06/23	11;13
14 Preparar Informe Final	16 días	lun 29/05/23	dom 18/06/23	
15 Preparar Presentac	10 días	lun 19/06/23	vie 30/06/23	
16 Preparar Dossier	11 días	lun 19/06/23	dom 02/07/2	

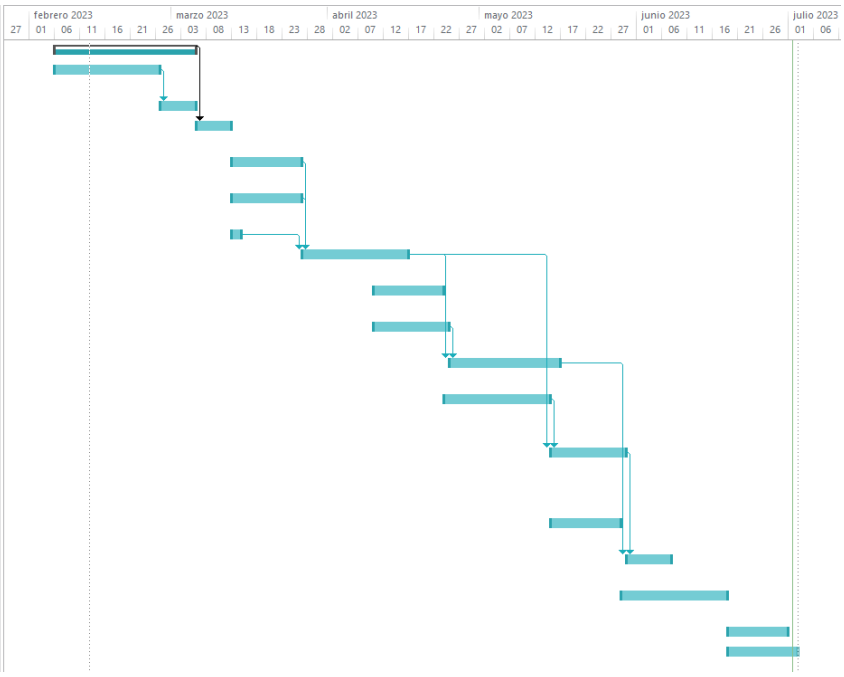

Fig. 3: Diagrama de gantt