Creació del joc Snake i implementació de IA bàsica.

Víctor Valero Carrasco

Resum—Resum del projecte, màxim 10 línies. ....
....
....
....
....
....
....
....
....
....

Paraules clau—Paraules clau del projecte, màxim 2 línies. ....
....

Abstract—Versió en anglès del resum. ....
....
....
....
....
....
....
....
....
....

Index Terms—Versió en anglès de les paraules clau. ....
....

---

1 INTRODUCCIÓ - CONTEXT DEL TREBALL

AQUEST projecte sorgeix de la meva motivació per aprendre a implementar un videojoc des de zero i adquirir coneixements per a l'execució eficient d'algorismes d'intel·ligència artificial. La selecció del joc Snake com a base per al desenvolupament d'aquest projecte no ha estat casual. Aquesta elecció es fonamenta en diversos factors que van influir en la presa de decisió.

En primer lloc, la popularitat del joc Snake el converteix en una elecció idònia, ja que existeix uan àmplia documentació i recusos disponibles per a la seva implementació. Aquesta abundància d'informació facilita l'aprenentatge i la resolució de problemes que puguin sorgir durant el desenvolupament.

D'altra banda, la simplicitat en la implementació del joc Snake no ha de ser confosa amb una manca de complexitat. Tot i la seva estructura aparentment senzilla, aquest joc ofereix elements distintius que proporcionen una base sòlida per adquirir les nocions bàsiques del desenvolupament de videojocs. Aquesta combinació de simplicitat i complexitat, permetrà explorar i comprendre els fonaments essencial de la creació de jocs.

A més dels objectius centrats en la implementació del joc, aquest projecte també es planteja explorar altres conceptes rellevants en el món dels videojocs i la tecnologia. Amb l'auge dels "Serious Games" i la "Gamification", aquest projecte busca entendre com els videojocs poden transcendir en l'àmbit de l'entreteniment i ser utilitzats amb finalitats educatives, de formació o resolució de problemes tant autònoms com en grups de treball. Aquesta exploració, pot proporcionar una perspectiva més àmplia sobre el paper dels videojocs en la societat actual i les seves aplicacions diverses més enllà de la recreació.

- *E-mail de contacte: victorvalerocarrasco@gmail.com*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Coen Antens (departament)*
- *Curs 2023-24.*

Finalment, l'aplicació de tècniques de "Reinforcement Learning" en el desenvolupament d'una Intel·ligència Artificial per al joc busca portar el projecte a un nivel superior. Això implica la creació d'una entitat virtual capaç d'entendre i millorar les seves accions mitjançant la ínteracció amb l'entorn del joc. Aqusta dimensió d'intel·ligència artificial proporcionarà una experiencia més dinàmica i desafiadora i, per a això, em serviré de la llibreria PyTorch per implementar algorismes d'IA i Reinforcement Learning.

2 Objectius

Amb l'objectiu de concloure de manera satisfactòria el projecte, m'he proposat assolir els següents objectius específics:

- **Objectiu 1:** Formació en les llibreries Pygame i Pytorch
- **Objectiu 2:** Implementació del joc Snake i jugabilitat humana.
- **Objectiu 3:** Realització de tests unitaris per al joc base.
- **Objectiu 4:** Distribució eficient de classes per a la implementació de la IA.
- **Objectiu 5:** Implementació de la IA de manera acurada.
- **Objectiu 6:** Estudi i implementació de Reinforcement Learning.

3 Metodologia

En quant a la metodologia, reconec que la selecció d'una metodologia adequada per a un projecte individual pot presentar certa complexitat. Malgrat això, he optat per implementar SCRUM com a metodologia preferent, ja que ofereix la possibilitat de realitzar un bon seguiment de les tasques. Per dur a terme SCRUM, faré servir l'eina "Jira", la qual em permetrà fer un bon seguiment detallat de les tasques, i a més, tinc a la meva disposició un taulell KANBAN. Així, podré utilitzar simultàniament les metodologies SCRUM i KANBAN per a una planificació efectiva.

Amb l'objectiu de posar en pràctica la metodologia SCRUM, planejo realitzar Sprints de dues setmanes, començant-los el 29 de febrer. Quan arribi al quart Sprint, aquests s'extendran a 3 setmanes. Aquest canvi de dos a tres setmanes es deu al fet que, a mesura que avanço en etapes més avançades del desenvolupament, experimentaré dificultats per a complir amb els terminis, tenint en compte part de documentació, implementació, etc. Per tant, per ajustar-se a les entregues d'informes, els Sprints passaran de dues a tres setmanes. Aquesta elecció, també pretén una sincronització amb les reunions de tutoratge, utilitzant l'últim dia de cada Sprint per aquestes sessions de feedback.

Aquesta estructura em permetrà aprofitar al màxim les reunions amb el tutor i disposar d'una setmana intermèdia per a la discussió dels progressos o possibles retrassos experimentats en el desenvolupament del projecte.

4 Planificació

Per a poder portar a terme el projecte, també em faré servir de l'eina "Jira", així mantindré centralitzada una eina on podré controlar tot el que està relacionat amb el meu projecte.

Per a establir una planificació precisa del projecte, he introduït conceptes coneguts com a "Epic", que representen les tasques de major envergadura. Cada Epic es descompon en subtasques més específiques, permetent-nos un seguiment detallat i establir un cronograma que faciliti l'estimació del temps necessari per a la seva realització.

Les tasques Epic a realitzar són les següents:

1. Entrega informe previ
2. Crear joc amb jugabilitat humana
3. Testing del joc amb jugabilitat humana.
4. Estudi d'algorismes d'IA.
5. Implementació d'algorismes d'IA.
6. Testing d'algorismes d'IA.
7. Entrega informe I
8. Estudi d'algorismes RL (Reinforcement Learning).
9. Implementació d'algorimes RL.
10. Testing d'algorismes RL.
11. Entrega informe II.
12. Conclusions i tancament del projecte.
13. Presentacions finals.

El detall del cronograma associat a aquestes tasques es troba disponible a la secció d'annexos d'aquest informe.

4.1 Actualització Informe Progrés I

Aquesta subsecció serà eliminada quan realitzi el informe final, però ens servirà per poder veure com va evolucionant el projecte.

Segons la planificació, hauria d'haver realtizat ja el joc Snake amb jugabilitat humana, el testing i el estudi d'algorismes de IA i implementar-lo, escollint així el de Reinforcement Learning. De moment, he realitzat el joc amb agent humà, testing i ja he començat a implementar l'algorisme de Reinforcement Learning, el qual dona errors en l'aplicació de l'agent, per tant estic en un punt correcte del desenvolupament.

5 Requeriments

Per a la realització del projecte, he fet un recaptació de requeriments fent servir la metodologia MOSCOW per prioritzar-los. Una metodologia on dividim els requeriments en Must (Imprescindibles), Should (Importants) i Could (Desitjables). A continuació, veurem les llistes amb els requisits, separats en funcionals i no funcionals i els seus

respectius codis on entenem RF com a Requisit Funcional i RNF com a Requisit No Funcional, seguit de un nombre que farà de codi identificador. Per tant el conjunt seria RF_00 o RNF_00 per identificar els diferents Requisits.

5.1 Requeriments funcionals (RF)

Must (Imprescindibles):

- RF_01: El joc ha de tenir una interfície que permeti als jugadors interactuar amb la serp fent servir el teclat.
- RF_02: La serp ha de poder moure's en les quatre direccions principals: amunt, avall, esquerra i dreta.
- RF_03: S'han de generar aliments aleatoris dins del límit del joc.
- RF_04: El joc ha de mostrar la puntuació del jugador en pantalla, actualitzant-se cada vegada que la serp s'alimenti.
- RF_05: Quan la serp col·lisioni amb un aliment, ha de créixer i augmentar la puntuació del jugador.
- RF_06: Quan la serp col·lisioni amb el final de la pantalla, o amb el seu propi cos, morirà i el jugador perdrà la partida.

Should (Importants):

- RF_07: El joc ha de permetre reiniciar la partida quan la serp col·lisioni amb els límits del joc o amb el seu propi cos.
- RF_08: La IA quan jugui ha d'aconseguir una puntuació mínima de 20 punts.

Could (Desitjables):

- RF_09: Incorporar efectes de so com a reacció a les accions del jugador, com ara la recollida d'aliments o col·lisió de la serp.
- RF_10: Incorporar un apartat visual atractiu al jugador.

5.1 Requeriments no funcionals

Must (Imprescindibles):

- RNF_01: El joc ha de ser lleuger i mantenir un bon rendiment, fins i tot en equips amb recursos limitats.
- RNF_02: El joc ha de ser compatible amb la majoria d'equips disponibles a l'actualitat.
- RNF_03: El joc ha de ser clar i fàcil d'entendre per a totes les persones.

Should (Importants):

- RNF_04: El codi ha de ser testejat, almenys cada línia de codi.

Could (Desitjables):

- RNF_05: Incorporar un disseny visual atractiu i coherent per millorar l'experiència de l'usuari i augmentar la immersió en el joc.

6 DESENVOLUPAMENT

A continuació, s'explica la implementació del joc Snake, la implementació basada en el jugador humà, la implementació basada en Reinforcement Learning i la realització de testing s'ha dut a terme per poder assegurar un bon funcionament del projecte.

6.1 Joc Snake amb interacció humana

En aquesta secció, descriuré tot el que s'ha desenvolupat de la part de creació del joc Snake, entenent així les classes creades.

- Classe Direction: És una classe que ens fa el mapeig a través d'un enum, on assignem la direcció UP al valor 1, la direcció DOWN al valor 2, la direcció LEFT al valor 3 i la direcció RIGHT al valor 4.
- Classe Snake: És la classe responsable de gestionar el comportament relacionat amb la serp al joc. Guarda informació sobre la longitud, direcció, cos de la serp, mida del bloc, color, límit, puntuació i temps d'inici.
- Classe Food: És responsable de gestionar la generació i el comportament dels aliments al joc. Guarda informació sobre la mida, color, coordenades i límit del joc.
- Classe MainHuman: Aquesta classe gestiona el flux principal del joc per interacció humana. Ens serveix per poder tenir un punt d'entrada centralitzat a aquesta funció del joc, on es separarà del punt d'interacció amb IA.

Fig X: Joc Snake amb acció humana

6.2 Testing del joc Snake amb interacció humana.

Una vegada ja hem creat el joc bàsic, s'ha de començar a fer proves sobre aquesta part. Per tant, el que hem fet és crear diferentes classes que ens ajudaran a cobrir tot el codi.

- Classe TestSnakeFood: Classe que farà test a totes les funcions que hi ha a la classe del joc Food.
- Classe TestSnakeGame: Classe que farà test a totes les funcions que hi ha a la classe del joc Snake.
- Classe MainTest: És responsable de l'execució de proves unitàries per a les classes TestSnakeFood i TestSnakeGame. És el entorn centralitzat de proves que ens servirà per crear un arxiu de cobertura.

La classe MainHuman, no fa falta ser testejada ja que és el només el control de flux del joc. Per tant, un correcte funcionament de les classes per separat, ens garanteix un correcte funcionament de MainHuman.

Sobre el arxiu de cobertura, gràcies a una llibreria anomenada Coverage.py, hem pogut crear un arxiu HTML que ens dirà un aproximat de la cobertura que passen els nostres tests

Module	statements	missing	excluded	coverage
V:\TFG_Snake_Victor\food.py	19	0	0	100%
V:\TFG_Snake_Victor\game.py	74	1	0	99%
maintest.py	17	1	0	94%
testfood.py	33	0	0	100%
testgame.py	96	0	0	100%
Total	239	2	0	99%

Fig X: Cobertura del joc amb agent humà

Com podeu observar, aquesta cobertura posa que falten 2 linies per provar. Fent una revisió dels tests podem observar que si que s'executen aquelles línies, per tant podem veure que hi ha un petit marge d'error.

6.3 Implementació del Reinforcement Learning

Seguidament, he implementat el aspecte del Reinforcement Learning. Com ara estem a un informe de progrés, explicaré per sobre els diferents enfocs que he tingut fins arribar a l'últim i com encara no està acabat no entraré en detalls de resultats i d'explicació de tecnicismes ja que pot variar una mica amb la implementació final.

6.3.1 Primera implementació LinearQNet sense QTrainer

La primera implementació, vaig intentar fer-ho simple ja que pensava en deixar-ho simplificat i vaig intentar aplicar LinearQNet sense una classe QTrainer, per tant tota la lògica dels passos d'entrenament es feien en un main_AI que em servia com a "Hub" d'entrenament on estaven gairebé totes les funcions relacionades amb les activitats d'entrenament. La vaig descartar ràpid perquè veia que el model tot i fer uns 3000-4000 episodis d'entrenament no aprenia res i no podia fer res per poder fer-lo aprendre. Ja en aquesta part, implemento unes noves clases d'agent i de model que a la segona implementació ens servirà per poder tenir una base.

6.3.2 Segona implementació LinearQNet amb QTrainer

Aquesta implementació és la final i la que tenim actualment. És una implementació que es basa en Q-Learning, implementant amb una xarxa neuronal (Deep Q-Learning). I segueix la següent lògica:

1. Inicialització: Es crea un agent amb una xarxa neuronal que inicialment pren accions de manera aleatòria (exploració).
2. Bucle de jocs:
    a. Per a cada joc, es reinicia l'estat del joc.
    b. L'agent observa l'estat actual del joc.
    c. Es pren una acció basada en una política ε-greedy.
    d. Es rep una recompensa i es passa a un nou estat.
    e. Es guarda la transició (estat actual, acció i recompensa, nou estat) a la memòria de l'agent.
    f. Es realitza un entrenament a curt termini amb aquesta transició.
    g. Si el joc ha acabat, es realitza un entrenament a llarg termini amb una mostra aleatòria de transicions de la memòria.
3. Actualització de la política: Es redueix gradualment el valor d'ε per afavorir l'explotació en comptes de l'exploració a mesura que l'agent aprèn.
4. Entrenament del model: Es realitza un entrenament del model ab les dades emmagatzemades.

La política ε-greedy és una política on tenim una probabilitat ε de prendre una acció aleatoria (exploració) i una probabilitat 1-ε es pren l'acció que maximitza el valor Q previst (explotació).

A continuació, explicarem les classes que hem implementat i què fa cada una d'elles:

- Classe Agent: És responsable de gestionar l'aprenentatge de la serp. S'encarrega de gestionar la memòria d'experiències passades, decidir quina acció prendre en funció de l'actual del joc, entrenar el model de xarxa neuronal per predir els valors de Q i recordar i entrenar tant a curt termini com a llarg termini.
- Classe Model (Linear_QNet i QTrainer): Linear_QNet, defineix una xarxa neuronal simple amb dues capes ocultes i batch normalization per a l'entrada i sortida dels valors Q. S'encarrega de: Definir l'arquitectura de la xarxa, guardar i carregar el model entrenat. QTrainer s'encarrega de definir l'optimitzador i la funció de pèrdua i implementar el pas d'entrenament que ajusta els pesos del model basant-se en les prediccions i els objectius calculats a partir de les recompenses.
- Classe SnakeGameAI: Aquesta classe, s'encarrega de fer servir la classe Snake inicial amb agent humà i adaptar-la a les necessitats de les classes de l'algorisme de Reinforcement Learning. Per tant, és una classe que s'encarrega de fer de pont entre el joc i la IA.

6.4 Validació de l'algorisme

Com he comentat prèviament, la part de la validació encara està pendent de ser implementada, ja que el principal problema que estic tenint amb la serp és sobre la memòria a llarg termini, ja que a curt termini està sent un èxit.

2.1 Exemple de subsecció

....

2.2 Exemple de subsecció

....

3 CONCLUSIÓ

....

AGRAÏMENTS

....

BIBLIOGRAFIA

[1] Loeber, P. Snake AI Pytorch. Recuperat 27 de febrer de 2024, GitHub. https://github.com/patrickloeber/snake-ai-pytorch

[2] amnindersingh1414. Snake Game in Python – Using Pygame module. Recuperat 27 de febrer de 2024, Geeks for Geeks. https://www.geeksforgeeks.org/snake-game-in-python-using-pygame-module/

[3] Wikipedia contributors. Snake (vídeo game). Recuperat 28 de febrer de 2024, Wikipedia. https://en.wikipedia.org/wiki/Snake_(video_game)

[4] Kim, Y. Pygame v2.1.4 documentation. Recuperat 28 de febrer de 2024, Pygame. https://www.pygame.org/docs/

APÈNDIX

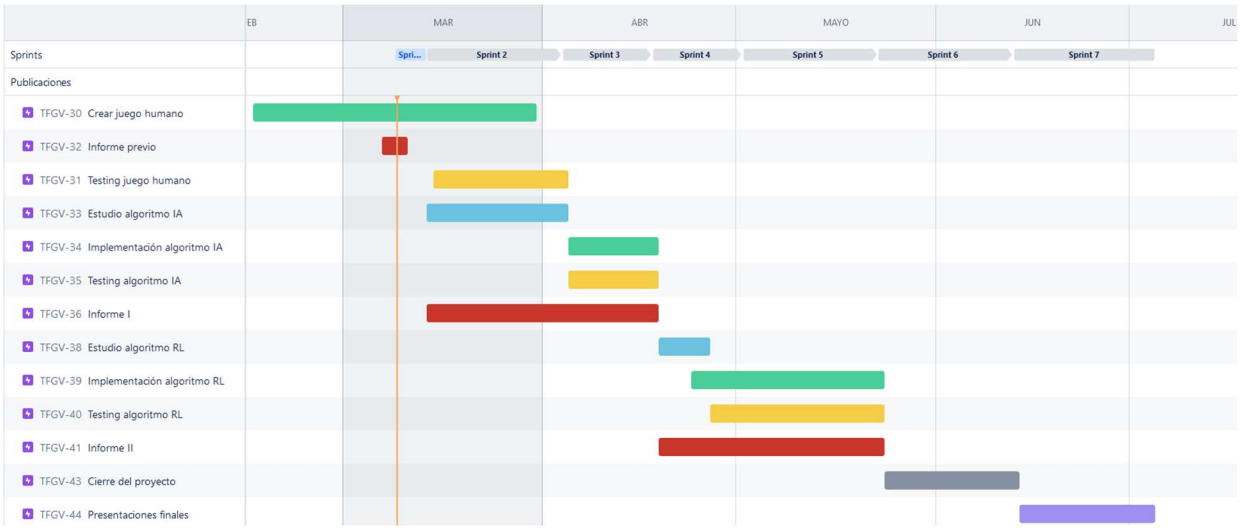
A1. SECCIÓ D'APÈNDIX

....

A2. SECCIÓ D'APÈNDIX

....

ANNEXOS


EB
MAR
ABR
MAYO
JUN
JUL
Sprints
Spri...
Sprint 2
Sprint 3
Sprint 4
Sprint 5
Sprint 6
Sprint 7
Publicaciones
TFGV-30 Crear juego humano
TFGV-32 Informe previo
TFGV-31 Testing juego humano
TFGV-33 Estudio algoritmo IA
TFGV-34 Implementación algoritmo IA
TFGV-35 Testing algoritmo IA
TFGV-36 Informe I
TFGV-38 Estudio algoritmo RL
TFGV-39 Implementación algoritmo RL
TFGV-40 Testing algoritmo RL
TFGV-41 Informe II
TFGV-43 Cierre del proyecto
TFGV-44 Presentaciones finales