Echipa: Flappy Birds
Membrii: Constantin Filip-Damian (gr. 342), Miu Elena-Adania (gr. 343), Sasu Alexandru-Cristian (gr. 342)

Flappy Bird AI

Ideea proiectului

Proiectul prezintă un mediu 2D, pentru un joc creat în Unity, care imită aplicația cunoscută sub numele de „Flappy Bird”. Astfel, în cadrul unei iterații a jocului, jucătorul parcurge un șir infinit de țevi, iar odată ce a trecut de o pereche de țevi care au un spațiu gol între ele, scorul acestuia crește cu o unitate. În caz că jucătorul se lovește de una dintre țevi, de pământ, sau încearcă să iasă prin tavan, jocul se sfârșește.

Detalii tehnice

Algoritmul de tip Reinforcement Learning utilizat:

Deoarece fiecare iterație a unui joc este diferită de orice altă iterație, (datorită apariției aleatoare a țevilor în cadrul fiecărei încercări) am ales să utilizăm algoritmul Proximal Policy Optimization (PPO). Astfel, polița din cadrul iterației curente va fi actualizată în funcție de datele generate în cadrul poliței (prezintă caracteristica on-policy), neținând cont de datele aferente unor polițe diferite de cea curentă (nu prezintă caracteristica off-policy). Alte motive pentru care am ales algoritmul PPO, le reprezintă utilizarea unui mediu necostisitor de eșantionat, ușurința implementării și utilizării sale, cât și posibilitatea experimentării cu o listă de hiperparametrii robuști.

Elemente specifice Unity utilizate pentru antrenare:

Din cadrul pachetului ML-Agents:

Pentru a antrena agentul, acestuia i-au fost asociate o multitudine de componente, anume: componenta „Behavior Parameters” pentru a defini comportamentul agentului (setul de date asociat, dacă utilizează sau nu un model deja antrenat), componenta „Decision Requester” pentru a-i spune agentului să analizeze ce acțiune trebuie să ia odată la fiecare cadru, cât și componenta „Ray Perception Sensor 2D”, care generează observații referitoare la mediul agentului, legate de spațiul pe care îl are pentru a se deplasa, obstacole, spațiul dintre țevi care îi crește scorul.

Alte elemente:

Echipa: Flappy Birds
Membrii: Constantin Filip-Damian (gr. 342), Miu Elena-Adania (gr. 343), Sasu Alexandru-Cristian (gr. 342)

A fost utilizată și componenta „Circle Collider 2D” pentru a se genera observații legate de mediul agentului (când acesta interacționează sau nu cu obstacole).

Setul de date utilizat:

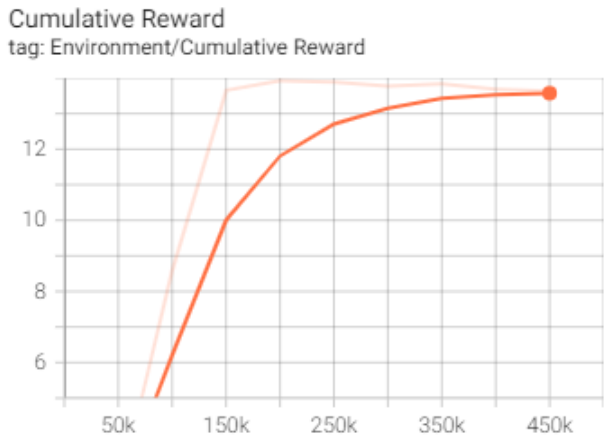
Deoarece în fiecare cadru avem nevoie să știm referitor la acțiunea luată de agent, doar dacă acesta a sărit sau nu, am ales să utilizăm un set de date discret cu 2 elemente, în cadrul căruia: 0 înseamnă că agentul nu a sărit, 1 înseamnă că agentul a sărit.

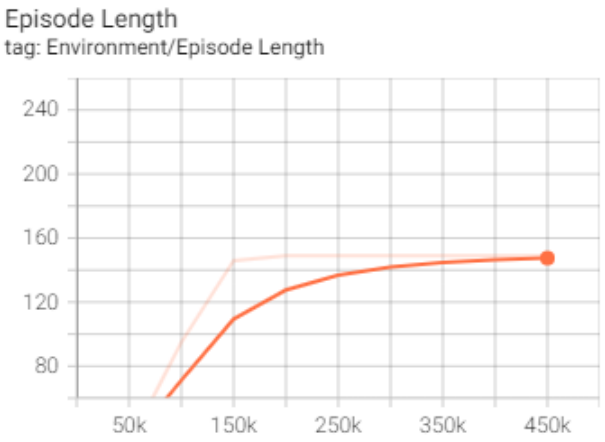
Răsplătirea și penalizarea agentului:

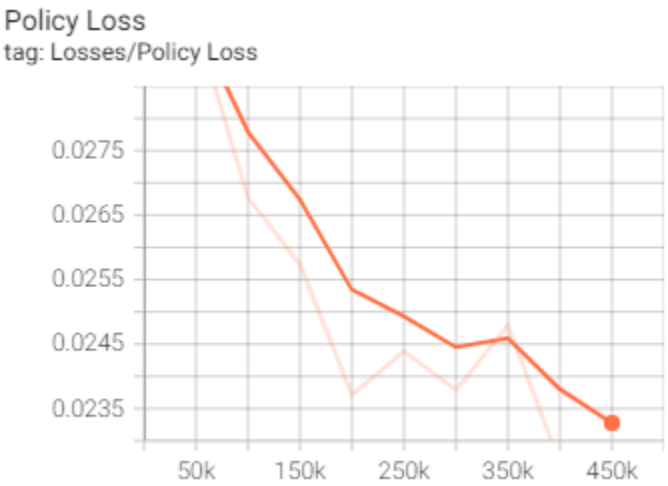
Dacă agentul supraviețuiește un cadru, acesta este răsplătit cu 0.1 puncte, iar dacă se lovește de un obstacol, este penalizat cu 1 punct.

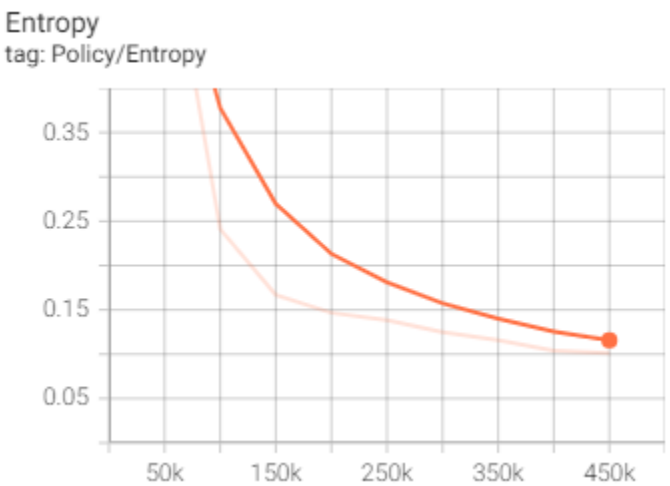
Concluzii și observații în urma antrenării agentului

Încercarea #1 de antrenare:







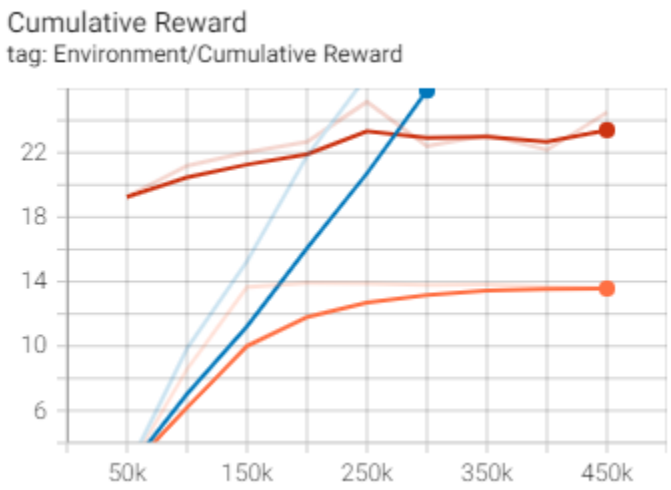


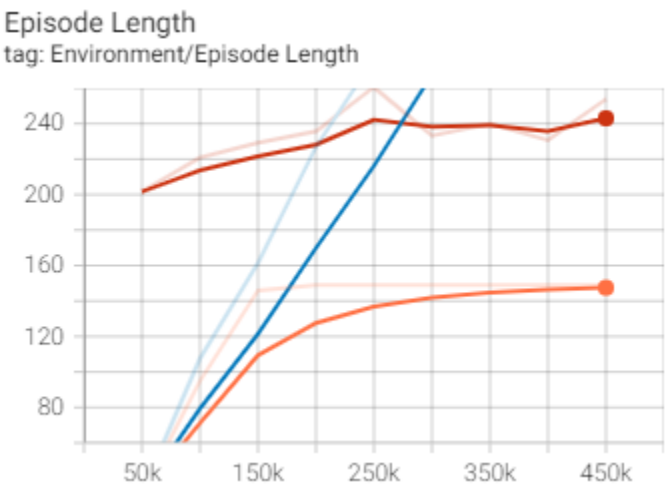
Echipa: Flappy Birds
Membrii: Constantin Filip-Damian (gr. 342), Miu Elena-Adania (gr. 343), Sasu Alexandru-Cristian (gr. 342)

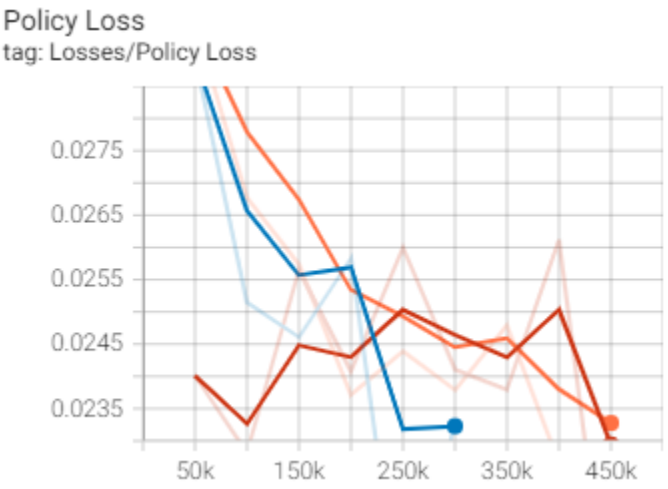
Deoarece agentul nu a mai fost antrenat deloc, acesta a început prin a se deplasa imediat după start-ul fiecărui episod, încontinuu, spre tavanul scenei, încheiând episodul într-un timp foarte scurt. După câteva minute de antrenare, a învățat să se stabilizeze în centrul ecranului, reușind să parcurgă distanța până la prima pereche de țevi, dar nefiind în stare să treacă prin spațiul îngust dintre ele, pierzând jocul prin a se deplasa rapid spre tavan atunci când ajungea aproape de țevi.

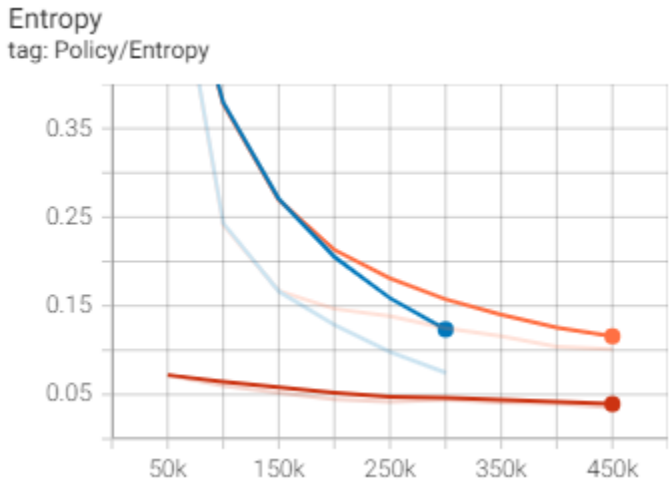
Încercările #2 și #3 de antrenare:

- Încercarea #2 folosește modelul antrenat în cadrul încercării #1, iar încercarea #3, cel antrenat în cadrul încercării #2









Încercarea #2 – culoarea albastră

Încercarea #3 – culoarea roșie

În cea de-a doua încercare, am preluat abordarea utilizată de strategia de antrenare „Curriculum Learning”, dar într-un format manual, neutilizând strategia propriu-zisă. Astfel, am mărit semnificativ distanța dintre

țevile unei perechi, ceea ce i-a permis agentului să parcurgă mediul cu o mult mai mare ușurință, reușind să adune o mulțime de puncte.

În cea de-a treia încercare, am renunțat la strategia abordată în încercarea precedentă, antrenând agentul în mediul nemodificat. Astfel, datorită antrenării anterioare, dar și a defectului căpătat în prima încercare, agentul a început să depășească un număr limitat de perechi de țevi (maximum ~20), întâmpinând în continuare problema generată de prima încercare.

Alte încercări intermediare de antrenare (bazate pe încercări precedente lor):

Am scăzut rata apariției țevilor, (de la 1 secundă la 1.5 secunde), păstrând distanța inițială dintre țevile unei perechi, ceea ce a ajutat agentul să învețe când să sară și când nu.

Am păstrat rata inițială de apariție a țevilor (1 secundă), dar am micșorat spațiul dintre țevile unei perechi, și am mărit suprafața de pe axa Y (atât în sus cât și în jos) în cadrul căreia se pot genera spații între țevile perechilor. Prin aceste încercări am dorit să învățăm agentul când să sară și când nu atunci când se află între țevile unei perechi.

Am introdus și configurat parametrul de curiozitate în fișierul de configurare al modelului, ceea ce nu a adus îmbunătățiri vizibile, din pricina mediului simplu și redus în dimensiune.

Am modificat caracteristicile componentei „Ray Perception Sensor 2D”, astfel încât agentul să poată observa mai bine împrejurările acestuia atunci când se află în spațiul dintre țevile unei perechi.

Concluzii în urma antrenării agentului:

Agentul a învățat să parcurgă mediul într-un mod eficient, reușind să adune sute de puncte într-un singur episod.