**Nyan Bird**

Proiect evaluarea Performantelor

Enachi Vasile

Muraru Alexandru

Ilioi Alexandru

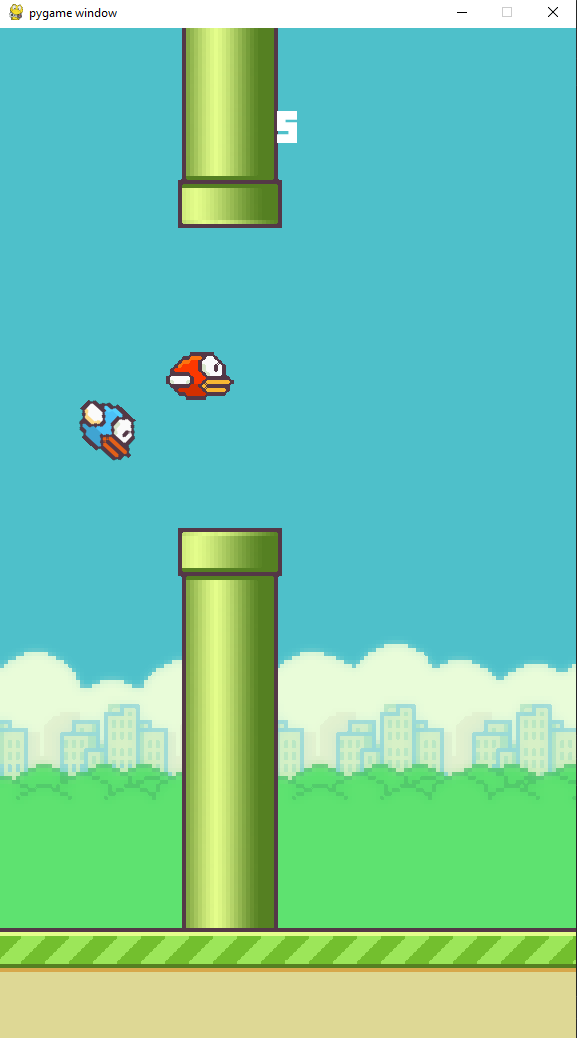
Grupa 1409B

**Enuntarea temei:**

Aplicatia noastra consta intr-o copie a jocului Flappy Bird construita cu ajutorul PyGame in care utilizatorul are posibilitatea de a concura cu un AI.

Jocul este din genul infinite runner, unde teoretic scorul poate merge pana la infinit si scopul este acela de a obtine un scor cat mai mare posibil. Jucatorul controleaza o pasare care trebuie sa evite toate obstacolele pentru a nu muri, acestea fiind: pipe-uri (din care unul din partea de jos si altul de sus), solul si tavanul. Pasarea este afectata de gravitatie si singura actiune pe care o poate executa utilizatorul este sa dea click pentru a o face sa sara in sus in incercarea de a ramane in viata.

Scopul jocului creat de noi este de a infrange calculatorul la acest joc.



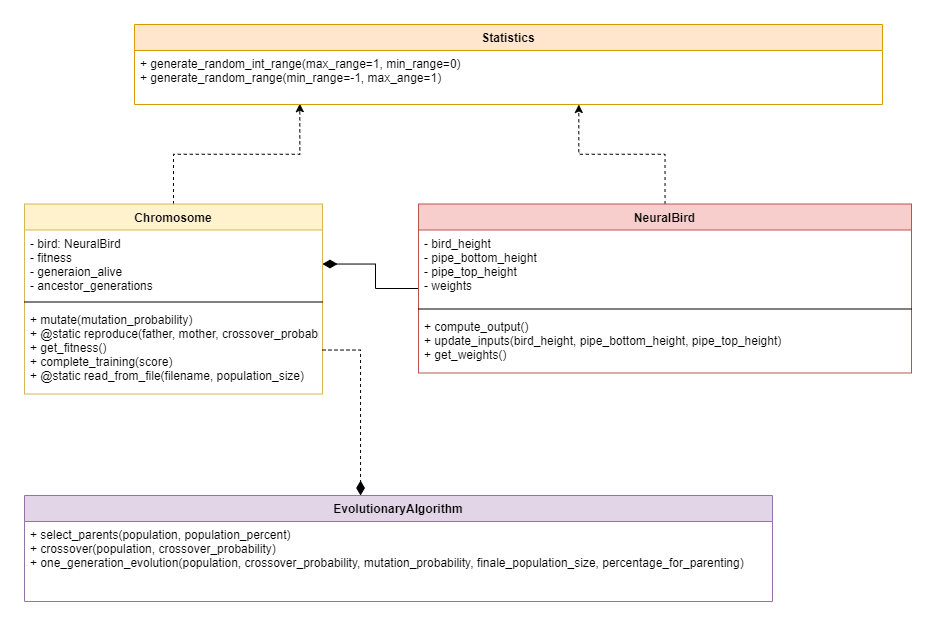
**Interfata grafica:**

Jocul are doua ferestre, prima avand rolul de a afisa scorul maxim obtinut de jucator, cum se joaca jocul si faptul ca poate incepe cand apasa pe ecran. A doua fereastra reprezinta jocul efectiv si tranzitia catre fereastra initiala se face atunci cand fie calculatorul fie jucatorul moare.



**Arhitectura**

Pentru algoritmul evolutiv



**Functionalitate joc**

Pentru a da iluzia de miscare, lasam pasarile in aceeasi pozitie, si doar mutam pipe-urile din dreapta ecranului spre stanga, pana cand acestea ies complet din vederea jucatorului dupa care le mutam inapoi in partea dreapta a ecranului pentru a crea iluzia ca au aparut altele noi. Singurul lucru care se modifica la pasari este pozitia pe verticala.

**Complexitati**

**In-game:**

Toate cazurile(best, medium, worst case) au aceleasi complexitati de spatiu si timp.

Pentru a crea pipe-uri noi folosim o memorie suplimentara de O(1) si timp O(1) pentru cazul ideal, mediu si cel mai rau.

Pentru a muta pipe-urile si a le desena avem O(1) memorie si timp O(n) unde n reprezinta numarul de pipe-uri active din scena. Cu toate acestea, din cauza modulu in care functioneaza crearea si stergerea pipe-urilor, n nu depaseste niciodata valoarea 4.

Pentru a testa coliziunea intre o pasare si obstacolele din scena avem O(1) memorie si timp O(n), unde n este numarul de pipe-uri active din scena deoarece numarul de pasari este constant, fiind mereu doua.

Pentru a da update la scor si high-score avem O(1) memorie si O(1) timp.

Pentru ca AI-ul sa ia decizia daca sa sara sau nu, am folosit O(1) memorie si O(1) timp.

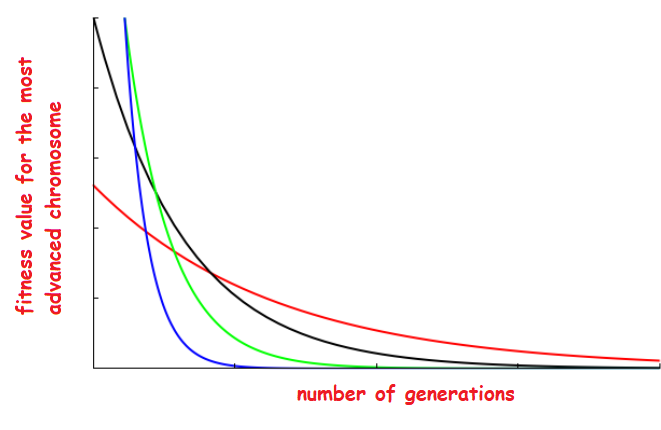
**Training AI**

Pentru a antrena AI-ul am recreat jocul si am rulat in paralel un numar considerabil de pasari. Ca metoda de antrenare am folosit un algoritm genetic iar ca model, o retea neuronala. Complexitatea functiei de evaluare a output-ului(daca trebuie sau nu sa sara pasarea) este O(1) ca timp si spatiu.

Pentru cazul mediul si nefavorabil este dificil de a determina complexitatea întrucât soluția este una cu caracter stohastic.

Inițial, valoarea fitness a celui mai dezvoltat individ converge rapid. Ulterior, cu cât atinge valoarea de prag, aceasta converge mai greu. Acest lucru poate fi explicat prin faptul că, cu cât genele ating o valoare mai mică, cu atât scade probabilitatea ca în urma încrucișării sau a mutației să fie generat un individ cu modulul genelor .

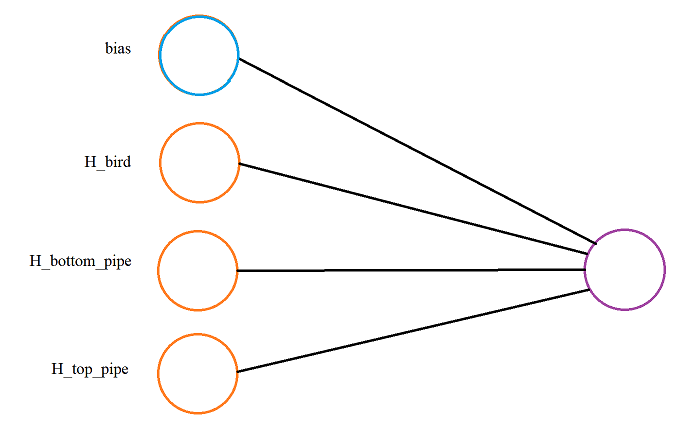
Viteza de convergență a soluției este determinată în special de probabilitatea de mutației, probabilitatea de încrucișare și metodele alese pentru operațiile respective.



**Modelul AI**

Pentru AI am folosit o retea neuronala cu 4 neuroni in stratul de input(3 de input si unul bias) si un singur neuron de output care descrie daca trebuie apasat sau nu pe ecran pentru a face pasarea sa sara.

Structura retelei este ilustrata in urmatoarea figura.



Bias este neuronul de bias.

H\_bird reprezinta inaltimea curenta a pasarii in joc.

H\_bottom\_pipe reprezinta inaltimea la care e situat urmatorul pipe de jos.

H\_top\_pipe reprezinta inaltimea la care e situat urmatorul pipe de sus.

Pentru antrenare am utilizat un algoritm genetic, cu sansa de incrucisare 90%, sansa de mutatie 20%, numar maxim de generatii 200 000. Ca functie de selectie am luat un segment aleator format din 50% din indivizi dupa care i-am incrucisat pana ajungem la cel mult dimensiune a populatiei fiilor egala cu cea initiala. Efectuam mutatiile si pastram doar cei mai buni indivizi. Ca functie de fitness am folosit distanta pe care o parcurge o pasare pana cand se loveste de un obstacol.

**Testarea**

Am utilizat testarea experimentala pentru AI, ruland de mai multe ori jocul. Parametrii celui mai bun individ sunt salvate in fisierul training.json in format JSON pentru a putea relua antrenarea la un punct ulterior. In timpul antrenarii AI-ul a atins un scor de 13 726 dupa care am oprit functionarea pentru ca rula deja de 8 ore.

Pentru a testa daca jocul se comporta asa cum ar trebui, am folosit din nou testarea experimentala jucand de mai multe ori. Pe parcursul acestui tip de testare am descoperit ca jocul nu rula la fel pe toate calculatoarele fiind dependent de framerate, problema ce a fost solutionata ulterior.

Integration testing a fost tot experimental si a decurs fara probleme cand am adaugat AI-ul la jocul initial.

Pentru testarea faptului ca generatorul functioneaza si da rezultatul corect si faptul ca weight-urile asociate neuronilor de intrare sunt valori in intervalul [-1,1] am folosit urmatoarele functii de testare.



**Membri**

**Enachi Vasile:** implementarea propriu-zisa a algorimtului genetic.

**Muraru Alexandru:** implementarea propriu-zisa a aplicatiei si integrarea AI-ului.

**Ilioi Alexandru:** proiectarea aplicatiei, testarea si realizarea documentatiei.

**Bibliografie**

Implementarea jocului <https://www.youtube.com/watch?v=UZg49z76cLw>

Pygame <https://www.pygame.org/docs/>

Algoritmi genetici <https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3>

Neural networks: <https://www.youtube.com/watch?v=OGHA-elMrxI>