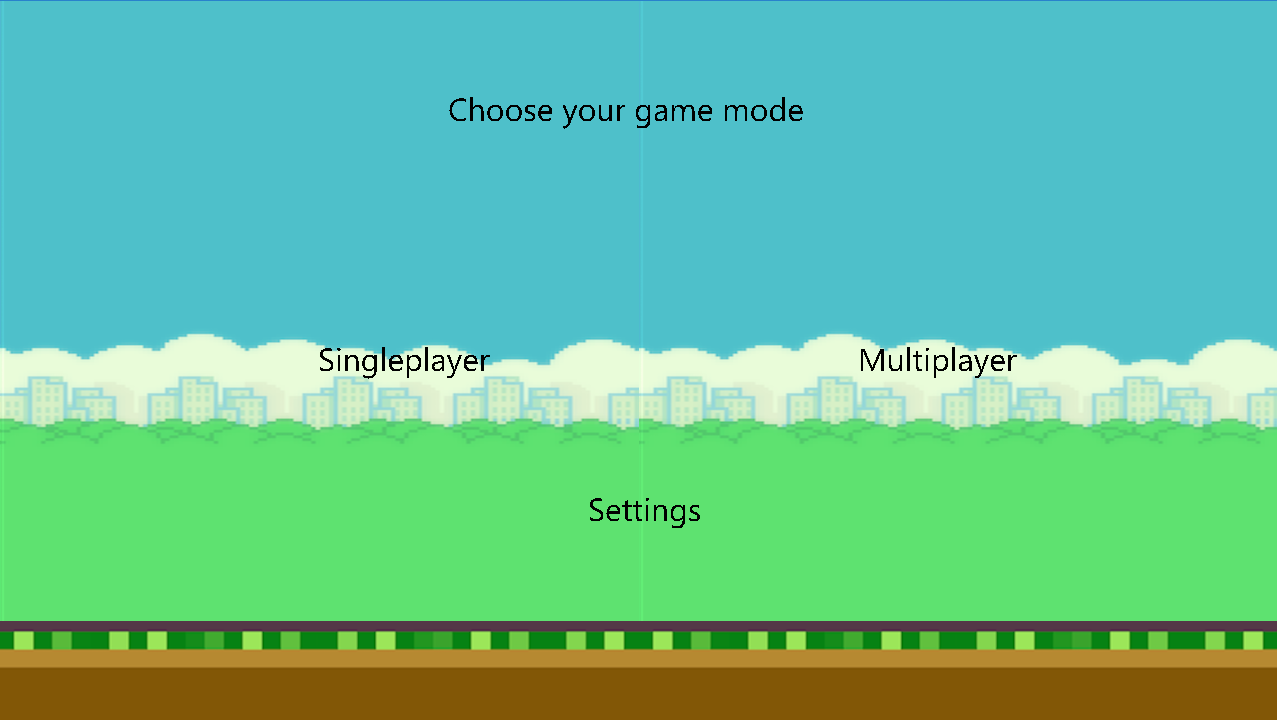
**Flappy Bird AI**

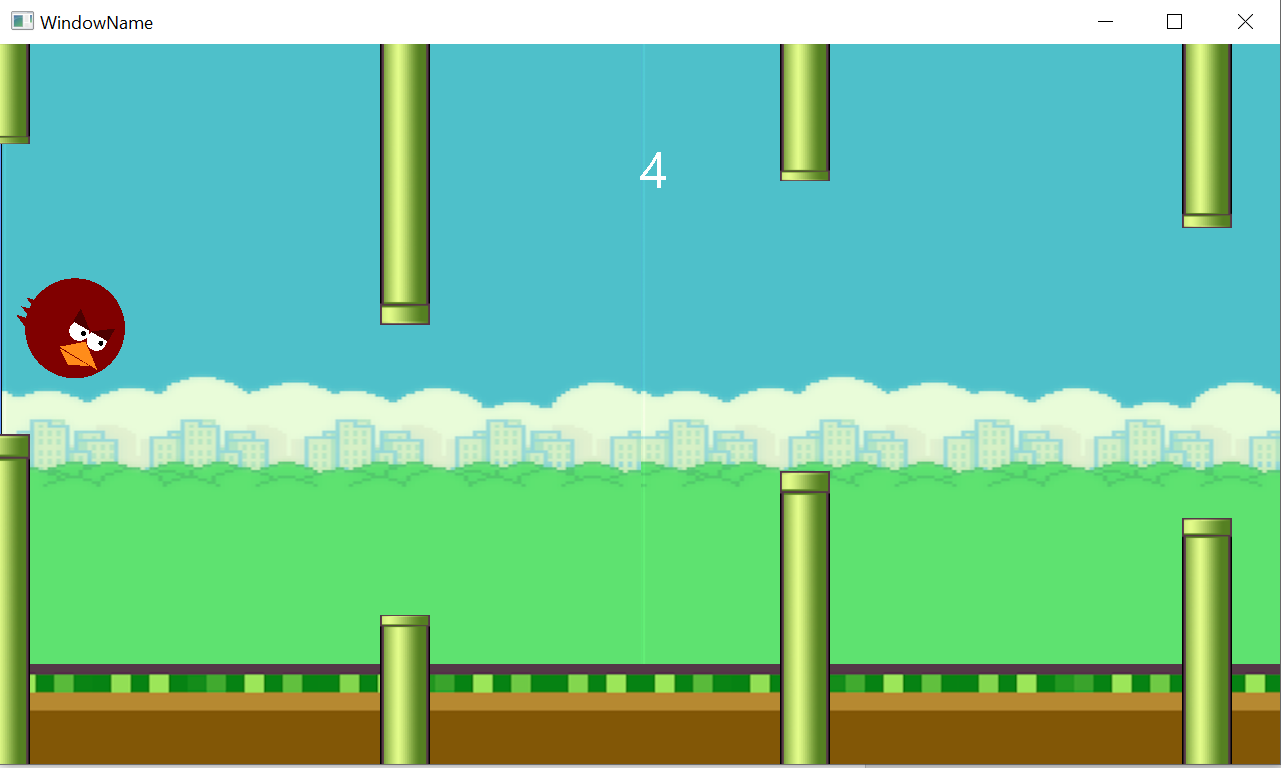
Radu Daniel

Este un joc de tip Flappy Bird cu 2 moduri de joc:

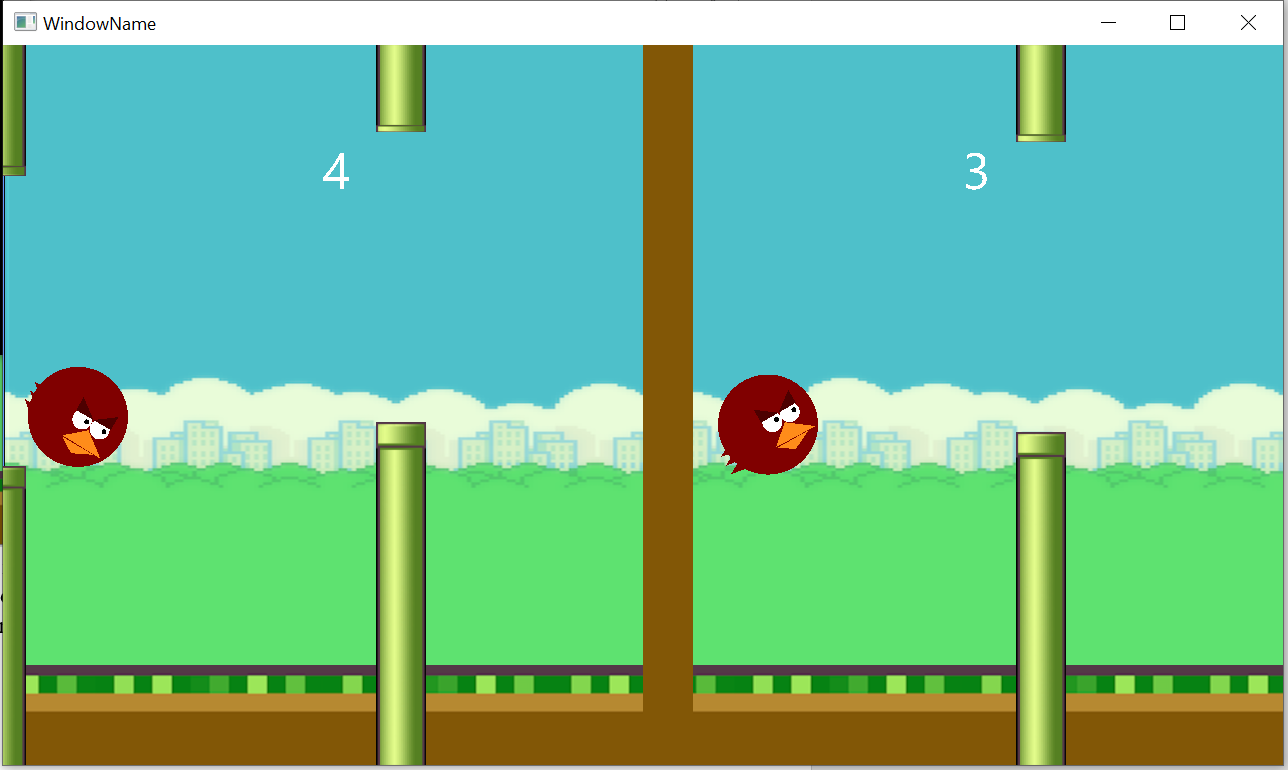


Interfata principala a jocului

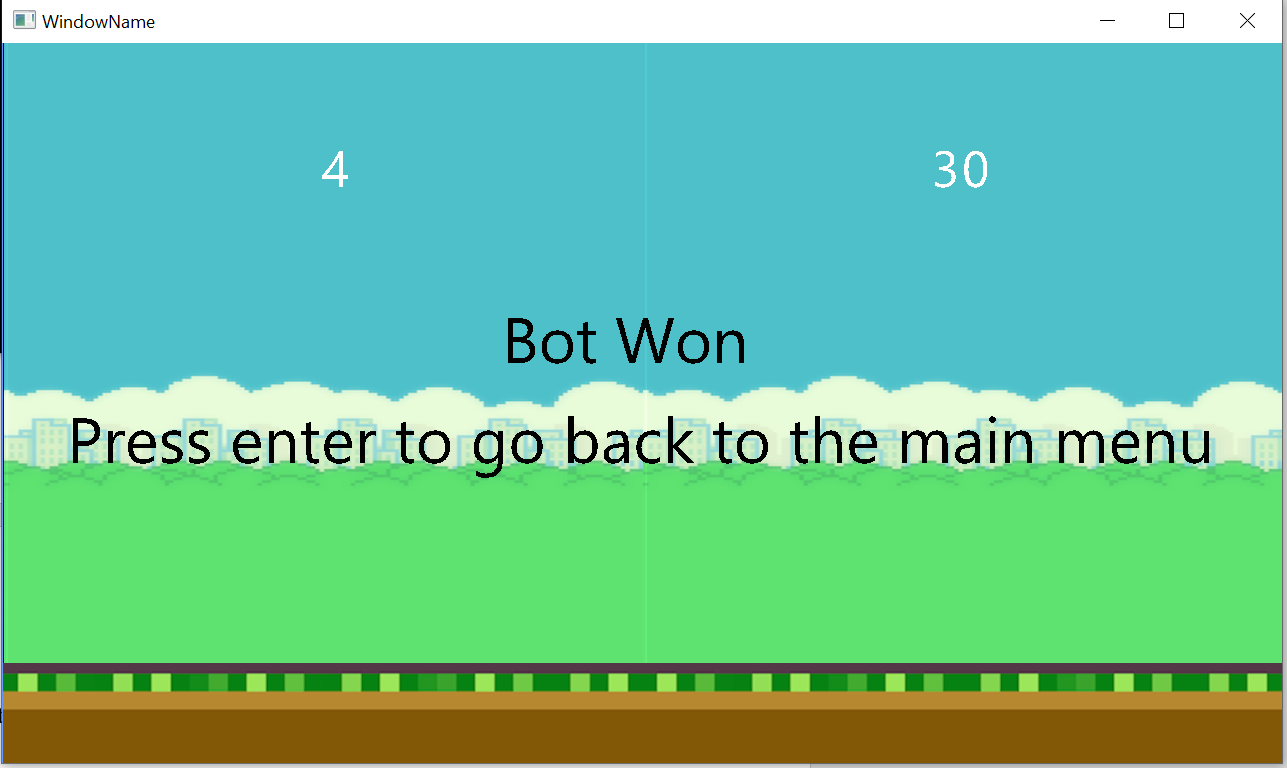
1.Jucatorul poate juca de unul singur. Scopul jucatorului este sa obtina un scor cat mai mare.



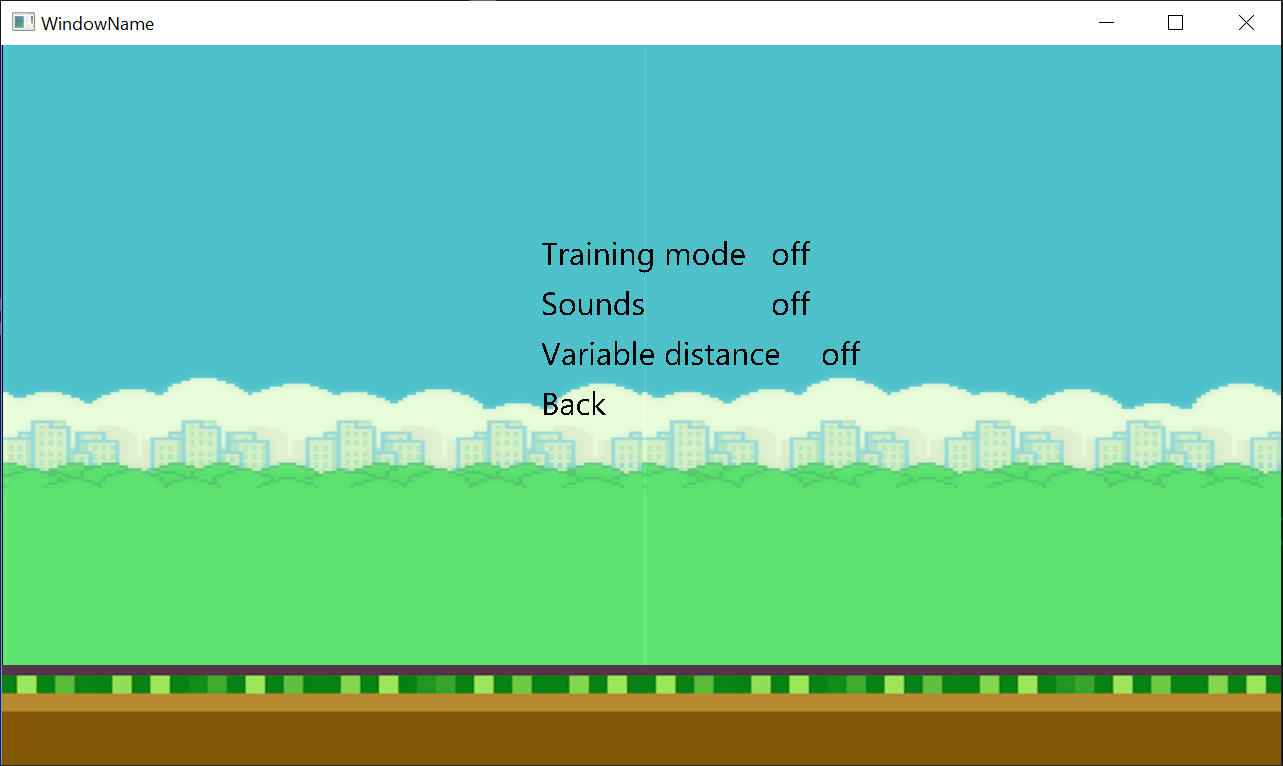
2.Jucatorul poate juca impotriva unui bot, avand mai multe optiuni legate de dificultatea jocului si de nivelul de experienta al botului. Scopul jucatorului este sa obtina un scor mai mare decat botul, pentru a castiga.

(jucatorul joaca in partea stanga, botul in partea dreapta)

Dupa ce jucatorul si botul se ciocnesc de un obstacol, va aparea un ecran care arata scorul final si cine a castigat runda.

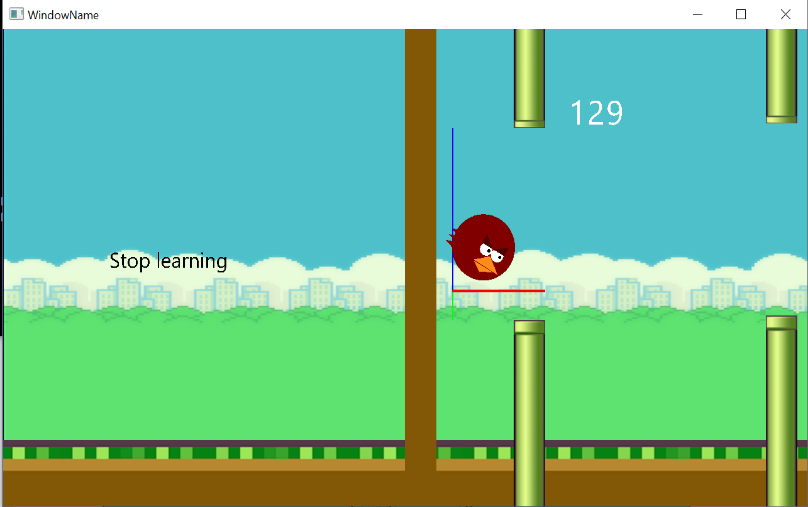


**Setari posibile:**



1. Training mode:

* In acest mod, botul va invata sa joace. Imediat dupa finalizarea unui episod(atunci cand are loc o coliziune si botul moare) jocul va fi repornit.
* Playerul nu poate juca impotriva botului
* Distantele folosite in procesul de invatare, ca parte a starii curente, vor fi desenate pe ecran(rosu pentru distanta pe orizontala, albastru pentru distanta pe verticala pana la obstacolul de sus si verde pentru distanta pe verticala pana la obstacolul de jos).
* Procesul de invatare poate sa fie oprit in orice moment prin apasarea butonului “**Stop learning**”.



2.Sounds: permite activarea sau dezactivarea suntelor jocului: sunete pentru aripi, sunete de coliziune si sunetul folosit atunci cand pasarea trece cu succes de un obstacol. Functia folosita pentru redarea sunetelor este din WinAPI si, din pacate, creeaza lag uneori, afectand procesul de decizie al botului. In procesul de invatare, este ideal ca sunetele sa fie oprite.

3.Variable distance:

Obstacolele sunt generate in mod dinamic, respectand cateva reguli:

* Obstacolele vin la interval de timp constant
* Inaltimile lor sunt determinate in mod random
* Cand “variable distance” este “off” distanta dintre cele 2 obstacole este constanta. Obstacolele au in continuare intaltimi random, dar distanta dintre conducta de sus si cea de jos este mereu aceeasi.
* Cand “variable distance” este “on” distanta dintre cele 2 conducte este si ea variabila, sporind dificultatea jocului.

**Procesul de invatare**: se foloseste algoritmul Q learning:

* Starea este compusa dintr-un numar de distante, explicate mai jos, pentru cele 2 cazuri posibile.
* Starea initiala Q0 este data de valorile initiale ale distantelor: (999, 0, 0)
* Exista 2 actiuni posibile pe care agentul le poate face: sa sara sau sa nu faca nimic.
* Recompensele posibile sunt **-1000** pentru cazul in care moare si **0** altfel. O recompensa de 0 pentru cazul in care actiunea aleasa nu rezulta intr-o coliziune determina agentul sa prioritizeze supravietuirea pe termen lung.
* Factorul de atenuare gamma este 0.9. Un factor de 1.0 rezulta in valorile pentru cele 2 actiuni posibile sa devina aproximativ egale dupa un timp suficient de lung, rezultand in decizii gresite.
* Rata de invatare este 0.7, pentru a obtine un balans intre ceea ce agentul a invatat deja si noile informatii.

**Stari**:

1.Pentru cazul in care distanta dintre conducta de sus si cea de jos este constanta:

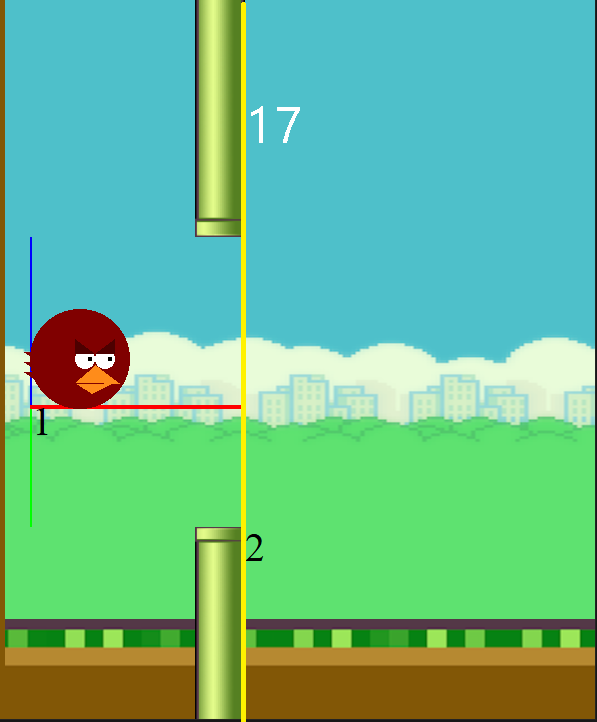
Starea este un tuplu (horizontal\_distance, lower\_vertical\_distance) unde **horizontal\_distance** reprezinta distanta pe orizontala pana la urmatoarea pereche de obstacole(linia rosie in imagine), iar **lower\_vertical\_distance** reprezinta distanta pe verticala pana la conducta de jos(linia verde in imagine).

Tinand cont ca distanta dintre cele 2 obstacole este constanta, tuplul de mai sus este suficient pentru ca agentul sa poate lua o decizie.

2.Pentru cazul in care distanta dintre cele 2 conducte este variabila:

In stare se mai adauga un element: (horizontal\_distance, lower\_vertical\_distance, upper\_vertical\_distance), unde **upper\_vertical\_distance**(linia albastra in imagine) este distanta pe verticala pana la conducta de sus. Acest lucru este necesar deoarece un obstacol de sus mai inalt inseamna ca agentul o sa fie nevoit sa sara mai devreme pentru a putea trece printre obstacole, fara sa il loveasca(deoarece spatiul dintre cele 2 obstacole o sa fie mai mic).

**Calcularea distantelor:**



Distantele se calculeaza raportat la punctul din coltul stanga-jos al agentului(punctul ‘1’) si marginea din partea dreapta a obstacolului urmator(punctul ‘2’). Atat timp cat agentul se afla inaintea obstacolului sau intre cele 2 conducte, toate distantele vor fi raportate la acest obstacol.

Doar dupa ce punctu ‘1’ trece de linia galbena agentul va considera urmatorul obstacol. Altfel, daca agentul incepe sa calculeze starea curenta raportat la urmatorul obstacol, cat timp se afla intre 2 conducte, acestea vor deveni transparente pentru agent si exista riscul ca un salt valid sa rezulte intr-o colizune cu conducta din partea de sus.

**Actualizarea valorilor din matricea Q:**

Actualizarea Q(s,a) in functie de Q(s’, a’) nu se face la fiecare tranzitie din starea s in starea s’ pe actiunea a’, toate perechile (s, s’, a) sunt retiune intr-o lista, iar atunci cand agentul moare se fac toate actualizarile, simultan, incepand cu ultima pereche de stari. Astfel, informatia se propaga mai repede catre starile anterioare.

In cazul in care agentul moare: ultimele 2 stari sunt penalizate cu un reward = -1000, deoarece putem considera ca reprezinta o secventa de actiuni gresite ce au dus la coliziune. In plus, daca agentul moare prin coliziune cu conducta de sus, se penalizeaza in mod suplimentar inca o stare deoarece o astfel de coliziune este in general rezultatul unor salturi gresite.

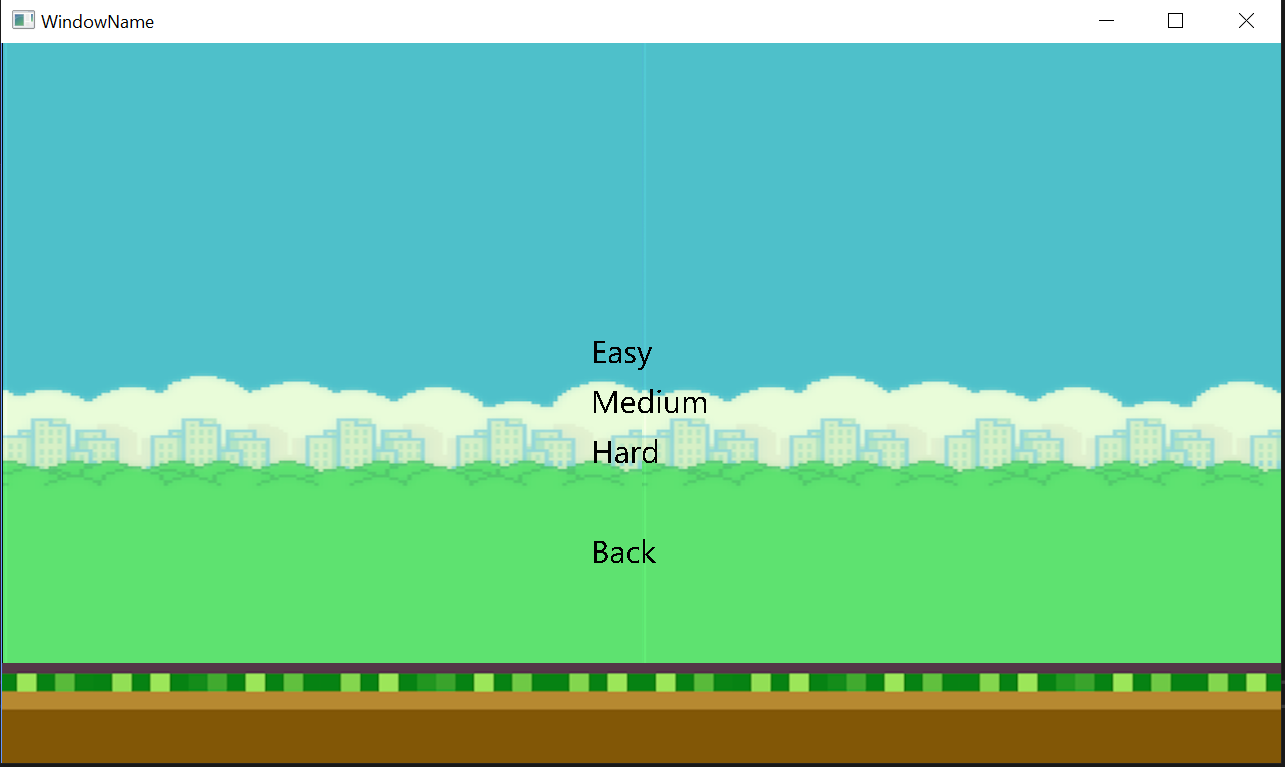
**Discretizarea spatiului:**

Distantele sunt numere reale, pot sa fie negative si pot sa ia un numar foarte mare de valori.

* Pentru a permite distante negative folosim map-uri in care cheie este reprezentata de cele 2/3 distante concatenate(dist1-dist2) in loc de o matrice.
* Pentru a reduce timpul de invatare distantele sunt transformate in numere intregi si scalate. Pentru cazul (1) in care distanta dintre cele 2 conducte este constanta, scalarea se face cu un factor de 4. Pentru cazul (2) scalarea se face cu un factor de 10, deoarece prin adaugarea celei de-a 3-a dimensiune, spatiul starilor devine foarte mare si timpul de invatare este extrem de lung.

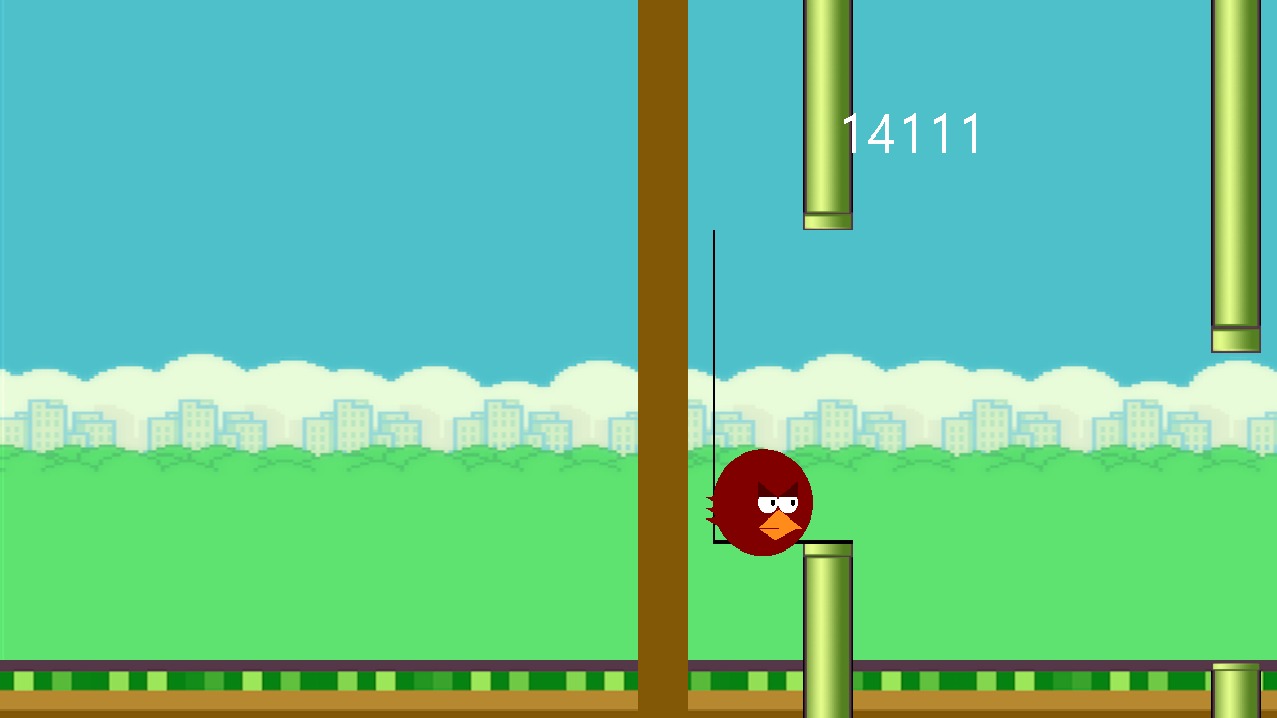
**Dificultati:**

Pentru cazul in care distantele dintre conducte sunt constante, botul are 3 niveluri de dificultate: easy, medium, hard. Fiecare nivel este obtinut prin antrenarea botului pe o perioada mai lunga sau mai scurt de timp.



**Rezultate:**

1.Cazul cu distante constante: am reusit sa obtinem un bot foarte eficient, care reuseste sa atinga scoruri de peste 10.000.



2.Cazul cu distante variabile: datorita dimensiunii mult mai mari a spatiului de stari, nu am avut timp suficient sa antrenam botul pentru a atinge scoruri atat de mari, insa reuseste sa obtina, in medie, scoruri de 50. Cu mai mult timp de antrenare, ar trebui sa ajunga la scoruri mult mai mari.

