FMI Unibuc, an 3, sem 1

Boss fight using Unity ML

Proiect Reinforcement Learning

*CodlinGoing:*

Nedelcu Radu-Andrei

Anghelache Vald-Alexandru

Stroie Mira Maria

Paula Iuga

**Cuprins**

[**Scopul lucrarii** 2](#_Toc89280929)

[**Introducere Reinforcement Learning** 2](#_Toc89280930)

[**Reinforcement Learning folosind ML agents** 2](#_Toc89280931)

[**Competitive Agents sau Self-play (**algoritmul**)** 4](#_Toc89280932)

## **Scopul lucrarii**

Dorim sa antrenam un adversar de joc in Unity pentru player folosing RL.

## **Introducere Reinforcement Learning**

Observatii

Rewards

Decizie

Actiuni

Reinforcement Learning poate fi reprezentat de un ciclu:

* Observatii - agentul observa (aduna informatii din meniu).
* Decizie – agentul ia o decizie bazata pe datele pe care le-a adunat deja
* Actiune – in urma deciziei agentul face o actiune/isi schimba starea
* Reward sau rasplata – agentul este raspalit in urma actiunii facute apoi ciclul se reia

## **Reinforcement Learning folosind ML agents**

Unity ML – how it works:

* ML Agents Toolkit – ML Agents python package -runs the algorithm
* Scene – Learning Env.
* Ml agent c# package – for definding data

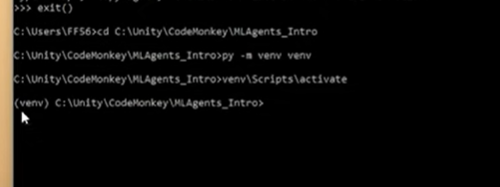
*RL : Observa -> ia decizie -> face o actiune -> primeste reward -> observa*

!ML merge doar cu nr

**ENV**

In venv

Agents-learn –run-id-Test2 => apoi play in unity pt training



**OBS**

sensor.AddObservation(this.transform.position); *// 3 floats pt pozitia lui*sensor.AddObservation(enemyTransform.position); *// 3 floats pt pozitia adversarului*sensor.AddObservation(playerStats.currentHP); *//1 val pt hp-ul curent*

* **Space Size = 7**
* **Stacked Vector 2 = ia decizia anterioara + o obsersatie noua pentru a face o noua decizie**

*CollectObservations*

**ACTIONS:**

Continous action => float

Action Space 1 => se misca pe x cu Vector.right\*getAxis()

*OnActionReceived*

**REWARD:**

Fuctiile din MlAgents:

* *SetReward*
* *AddReward*

Functia Reward descrie cum ar trebui sa se comporte agentul/ce vrem sa obtina in timpul jocului. Vrem ca agentul antrenat sa poata fi un adversat portivit pentru un player:

* + sa il atace
  + sa isi mareasca health points luand mere (un reward game object)
  + sa isi mentina health points ridicat ferindu-se de atacurile inamicului
  + sa foloseasca environmentul (platforme, teren, collision objects) pentru a se deplasa/feri de inamic

Training:

Prefab pt invironment + duplicates se antreneaza mai multe simultan

EpisodeEnd()

OnEpisodeBegin() = da restart la episod ca sa se poata antrena din nou

Dupa endTraining => .onx se copiaza modelul neuronal in proiect => sa face drag in Model de la Behaviour Parameters

Hiperparametrii:

(Numele Brainului apoi parametrii) in config.yml

Pt a rula cu noul brain *mlagents-learn config/<nume\_config>.yml*

Vizualizare cu TensorBoard: pentru improve la Hiperparametrii



## **Competitive Agents sau Self-play (algoritmul)**

2 inamici se bat => astfel vom pute obtine un inamic ia pentru playerul cu care se joaca userul.

Fiecare inamic are o copie a policy-ului curent, doar unul invata celalat are un fixed policy.