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

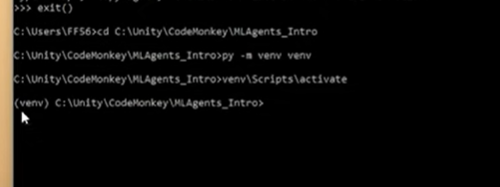
*RL : Observa -> ia decizie -> face o actiune -> primeste reward -> observa*

!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:**

*SetReward*

*Add Reward*

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**

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.