Aprenentatge per reforç amb Unity

TAIA Project

Wilber Eduardo Bermeo Quito

14 de novembre de 2021

Índex

1 Problema a resoldre en una frase

Fer que un NPC segueixi una política en concret fent ús de Deep Reinforcement Learning en Unity.

2 Introducció i Motivació

Durant l'estiu del 2021 m'he estat barallant amb Unity (motor de videojocs) perquè vull fer un joc com a treball final de grau.

El joc, resumidament, serà un Metroidvania 2D, en el qual hi haurà una tira de diferents NPCs. Molts d'aquests NPCs seran personatges actius i enemics.

La motivació és clara, aprendre més d'Unity i expandir la riquesa del TFG amb les eines pròpies d'Unity que dona per implementar intel·ligència artificial als videojocs.

3 Dades/Coneixements dels que es disposa

Degut a la naturalesa del problema seleccionat, no es necessiten conjunts de dades per poder treballar.

Conec l'entorn de treball Unity, no professionalment, però si tinc un base.

El que em permetrà encarar el problema és l'eina *ML-Agents Toolkit*. Dona totes les eines necessàries per utilitzar Unity com a motor de simulació perquè els Agents de les escenes aprenguin polítiques segons una modelització del problema a resoldre.

4 Proposta

4.1 Agent

Classe que conté mètodes que poden ser sobre escrits. La seva API contempla mètodes per poder generar observacions del medi, per prendre accions i per assignar recompenses.

La classe Agent conte altres mètodes que poden ser sobre escrits. Cada Agent està relacionat amb un Behavior Parameter class i a un Decision Requester.

4.1.1 Com modelitzar un Agent?

Es necessiten definir tres tipus d'entitats per cada moment en el joc.

- Observacions
- Accions
- Recompenses

4.1.1.1 Observacions

Les observacions poden ser numèriques o visuals. Les observacions numèriques són del punt de vista de l'observador (NPC), les visuals en canvi, són generades per col·lisions de rajos que es poden ajuntada a l'agent i representen el que l'agent està veient en aquell mateix moment.

4.1.1.2 Accions

El que realment l'agent pot fer dintre de l'escena. Les accions poden ser representades per valors continus o discrets.

4.1.1.3 Recompenses

És un escalar que representa que tan bé ho està fent l'agent. Les recompenses (negatives o positives) no tenen per què ser donades a cada moment, es poden donar en certs moments provocats per esdeveniments. És important saber quan i quant es recompensa o es castiga a l'agent pel seu acte. Depenent de la política de recompenses farà que l'aprenentatge d'una certa tasca sigui més ràpida o més lenta.

4.2 Polítiques a aprendre

Els següents dos punts són els problemes a modelitzar perquè l'Agent aprengui la política adequada per compliar-los.

4.2.1 Autonomia en el moviment

L'NPC ha de ser capaç de moure d'esquerra a dreta de manera cíclica. El radi de desplaçament respecte al punt origen ha de ser configurable.

Arribar al màxim rang de desplaçament tant d'esquerra a la dreta ha de donar feedback positiu, però no ha de passar que es quedi en un extrem sense moure, només s'ha de poder atorgar feedback positius i s'ha vingut de l'altre extrem, un cas excepcional seria la primera vegada que apareix en l'entorn del joc, ja que surt del centre del desplaçament.

En cas de col·lisions amb objectes que no siguin targets d'atac dintre del seu rang ha de poder ser capaç de girar i fer el recorregut contrari.

4.2.2 Saber atacar

Un cop l'Agent hagi apres a moure's, la meva idea es que l'NPC detecti entitats a atacar.

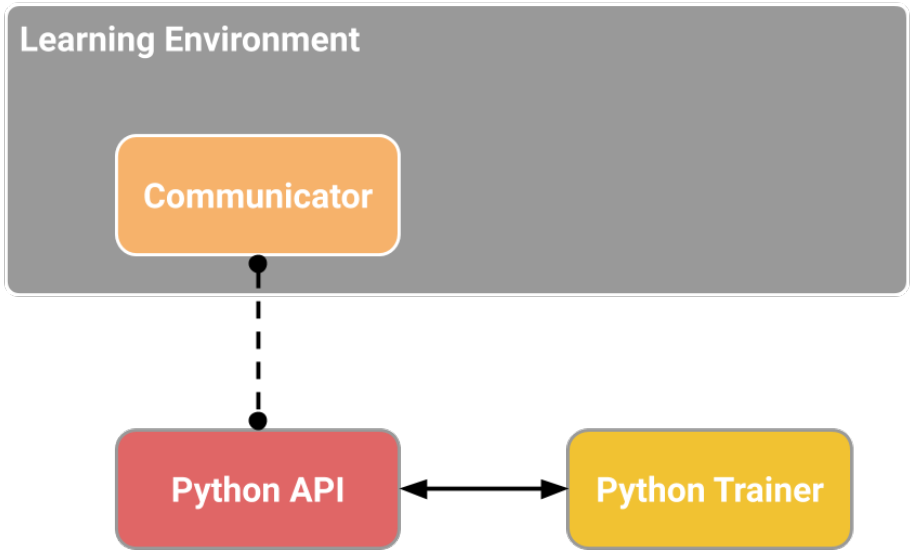
Aquestes entitats únicament seran detectables si estan dintre d'un rang esfèric que tindrà com radi el rang de desplaçament del NPC. L'objectiu en aquest cas es que, deixi de fer el moviment cíclic d'esquerra detrac i que salti sobre l'objectiu fent col·lisió, les col·lisions encertades donaran feedback positiu.

4.3 Arquitectura, Algorismes, Propostes existents

L'arquitectura del problema que abordo bé donat per les eines que utilitzo per resoldre-ho, en aquest cas *ML-Agents Toolkit*. A part de les eines de *ML-Agents Toolkit* faré servir *Tensorboard* per veure les estadístiques d'aprenentatge.

A continuació explicaré com és l'arquitectura de ML-Agents Toolkit. ML-Agents Toolkit té els següents quatre blocs com a estructura principal.

- Espai d'aprenentatge (Escena d'Unity)
- Comuniador (Broquer que connecta Unity amb l'API de Python)
- Python API
- Python Trainers



4.3.1 Espai d'aprenentatge

Aquí està l'escena d'Unity i els personatges. Els personatges que per requeriment del problema s'han definit com a Agents, podran observar, prendre decisions i tenir recompenses.

4.3.2 Agent

Aquí està l'escena d'Unity i els personatges. Els personatges que per requeriment del problema s'han definit com a Agents, podran observar, prendre decisions i tenir recompenses.

- Initialize
- OnEpisodiBegin
- OnActionReceived
- CollectObservations
- Heuristic

4.3.2.1 Initialize

Aquest mètode es crida un cop únicament en el cicle de vida de l'aprenentatge. Està pensat per fer caching d'objectes de l'escena o inicialitzar valors.

4.3.2.2 OnEpisodiBegin

Aquest mètode es crida cada cop que un episodi d'aprenentatge ha acabat, forçadament o perquè el màxim nombre de passos de l'episodi s'han completat.

4.3.2.3 OnActionReceived

Aquest mètode porta un paràmetre d'entrada, anomenat ActionBuffer, l'ActionBuffer, és un objecte que té dos arrays, un per poder representar valors continus i altres per representar valors discrets.

Aquest mètode interactua tant amb el mètode *heuristic* i amb el motor d'aprenentatge automàtic.

4.3.2.4 CollectObservations

Aquest mètode s'executa cada x temps, aquest temps d'execució es configura amb l'eina d'Unity.

El mètode té com a paràmetre una estructura de dades semblants com a paràmetre d'entrada que el mètode OnActionReceived. En aquest cas no està pensat per consumir les dades sinó per modificar aquesta referència i per donar informació al motor de Reinforcement Learning quan estem en mode de *Learning*.

4.3.2.5 Heuristic

Aquest mètode permet interactuar amb el jugador. Fixat que té la mateixa estructura de dades com a paràmetre que el mètode ActionBuffers.

Els valors que s'emplenin en els buffers d'aquesta estructura de dades aniran a parar al mètode *OnActionReceived*.

4.3.3 Behaviour

Script que esta pensat per parametritzar i relacionar l'Agent amb la configuració externa a Unity.

Un Behavior pot tenir els següents comportaments:

- Learning

- Heuristic
- Inference

4.3.3.1 Learning

Quan el *Behavior* està en mode *Learning* vol dir que utilitza el model generat per la sobreescriptura dels mètodes de la clase *Agent* per aprendre. En aquest mode Unity es connecta amb el procés de *mlagents-learn* per generar la xarxa neuronal que representa l'aprenentatge del model.

4.3.3.2 Heuristic

En aquest mode no hi ha procés d'aprenentatge, es fa servir la funció *heuristic* de la classe *Agent* per interactuar amb l'Agent.

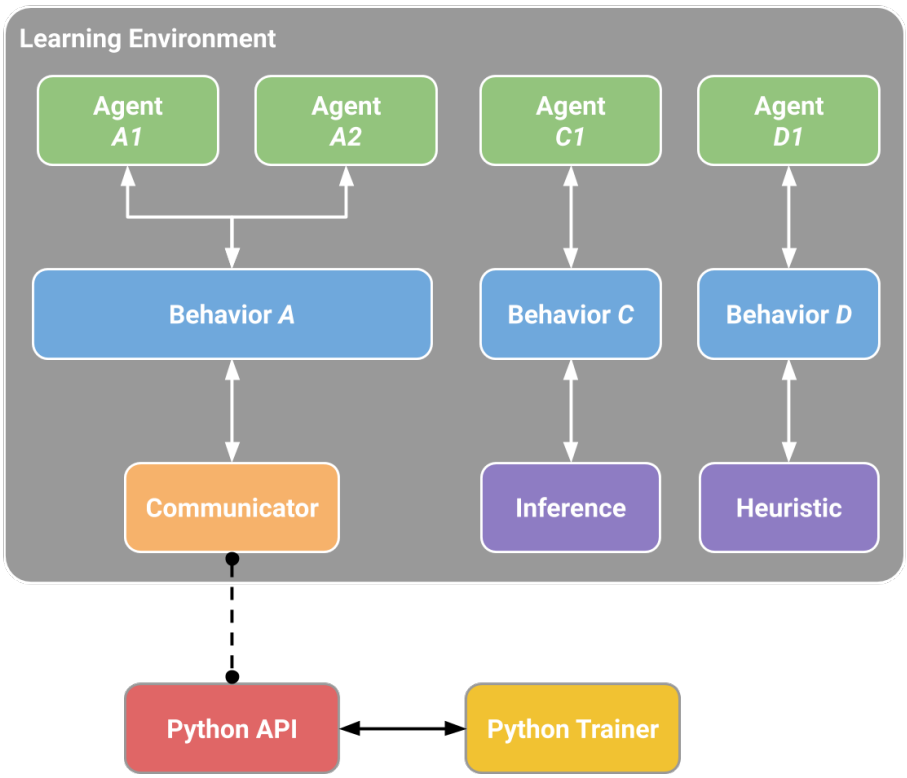
4.3.3.3 Inference

Quan el *Behavior* està en mode *Inference* implica que hi ha una xarxa neuronal amb una política apresa, i l'objecte que té aquesta xarxa neuronal es comportarà segons el que s'hi hagi après.

4.3.4 Requester

És un Script que serveix per forçar que l'Agent prengui decisions, sense aquest Script com a Component en el nostre Agent, l'Agent mai prendrà decisions. Aquí es pot configurar el període de decisions.

En la següent imatge hi ha les tres diferents arquitectures que podem tenir segons quin comportament tingui el component *Behavior* del objecte Agent.



4.3.5 Tensorboard

Quan l'entrenament s'està executant, ML-Agents Toolkit guarda estadístiques dintre del directori resolts en un directori amb nom únic, donat a l'hora d'executar el procés mlagents-learn.

Per observar el procés d'entrenament quan està corrent el procés o quan no, ens apropem a la carpeta de resultats i executem: tensorboard –logdir */results${training id}.

Això obrirà un servidor per servir el contingut estàtic en un dels ports de la teva màquina.

Explicaré alguna de les gràfiques que *Tensorboard* genera, ja que molt segurament les veurem més endavant en cada un de les versions de l'Agent.

4.3.5.1 Cumulative Reward

Gràfica que representa la mitja de el que guanya l'Agent per episodi. Un bon procés d'aprenentatge gènera una gràfica ascendent que en algun moment s'estanca, ja que la seva mitja per episodi és manté.

4.3.5.2 Episodi Length

Representa la durada mitjana per episodi. Aquesta gràfica depèn de la implementació del model, ja que segons la característica del problema es pot forçar el reinici d'episodi, o potser el problema és maximitzar o minimitzar el temps de vida de l'Agent.

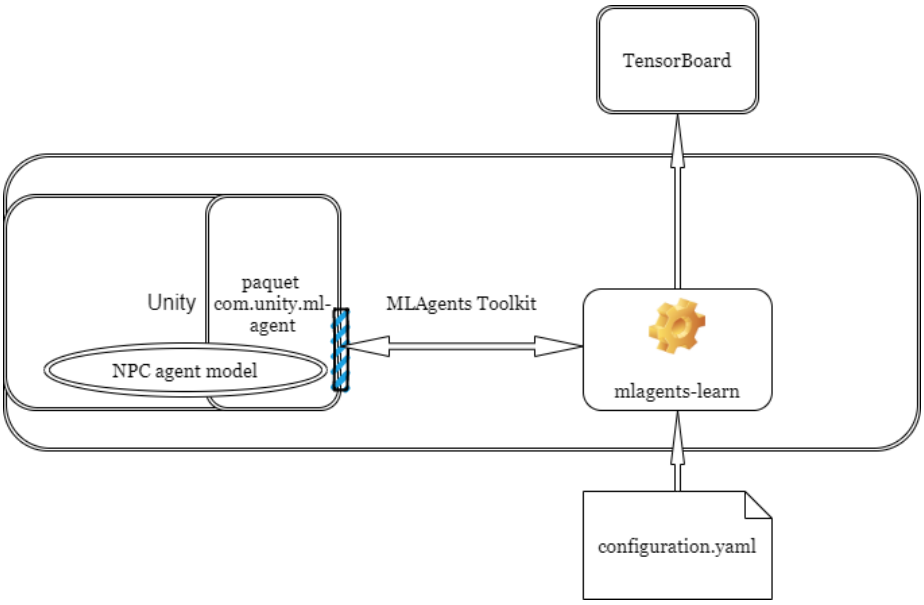
4.3.5.3 Policy Loss

Representa quan la política de l'Agent canvia, un model correcte tendirà a fer una gràfica descendent.

4.3.5.4 Entropy

Representa que tan aleatoris són els moviments de l'Agent, un model correcte expressa una gràfica d'Entropia descendent, ja que significa que realment està aprenent una política i no està fent tants d'intents per provar coses noves.

Arquitectura final



5 Experiments

Aquest projecte conté un seguit d'experiments basats en fases. Les fases venen donades per l'apartat *4.2 Polítiques a aprendre*.

5.1 Representació de l'objecte NPC dintre d'Unity

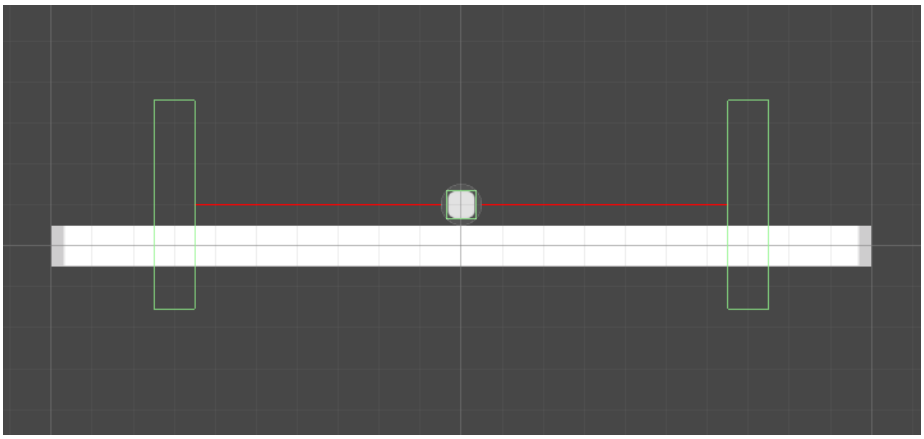
L'objecte NPC està format tres objectes fills.

- Limit esquerra
- Limit dret
- Agent
  - Raig esquerra
  - Raig dret

Els objectes de límits, són objectes buits i invisibles que tenen un comportament de trigger de esdeveniments en cas de col·lisions. Aquests objectes són essencials per poder delimitar el rang de moviment de l'Agent.

L'objecte Agent conte el Script de modelatge que hereta de la classe *Agent* a més dels Scripts *Behavior Parameters* i *DesitionRequester*.

Finalment tenim els rajos, que interaccionen amb elements de l'escena. El funcionament d'aquests rajos s'explica en l'apartat *4.1.1.1 Observacions*.



5.2 Autonomia en el moviment

5.2.1 Versió (I)

Els mètodes que s'hi han sobreescrit de la classe *Agent* són els següents:

- Initialize
- OnEpisodiBegin
- OnActionReceived
- CollectObservations
- Heuristic

Altres mètodes de classe no propis de la classe *Agent* i que són interessants per poder explicar la modelització del problema:

- OnTriggerEnter2D
- OnTriggerExit2D

5.2.1.1 Initialize

Rutina que s'executa només un cop quan es dona play al motor d'Unity.

Agafo el component de l'objecte que em permet treballar amb les físiques de Unity. El component és el *RigidBody* que ha d'haver estat afegit a l'objecte com una component des de l'interficie d'Unity. El component *RigidBody* ha de tenir el comportament de *Dynamic* perquè sigui afectat per la gravetat de configurada en l'escena.

Per altra banda configuro el nombre de passos per episodi a zero, en cas que l'estat de la variable *trainingMode* sigui fals. Aquesta configuració em permet fer proves en mode *Heuristic* o en mode *Inferencia* sense passar pel cicle de vida *OnEpisidoBegin*.

La variable *MaxStep* és una variable pròpia de la classe *Agent* i representa el nombre de passos màxims que es fan durant l'Agent està aprenent una política. Quan s'ha fet aquests nombres de passos, immediatament pel cicle de vida de l'Agent, força l'execució de la rutina *OnEpisodiBegin*.

```
1  public override void Initialize()
2  {
3    _rigidbody2D = GetComponent<Rigidbody2D>();
4    // infinite steps for session
5    if (!trainningMode) MaxStep = 0;
6  }
7
```

5.2.1.2 OnEpisodeBegin

El següent mètode s'executa a l'inici de cada episodi. És important de restablir els estats inicials d'algunes propietats de l'Agent, degut a què l'Agent manté la inèrcia del episodi anterior.

En la rutina es treu l'inèrcia que tenia l'Agent en l'episodi anterior, se'l reposiciona al centre i s'escull una meta aleatòriament. A aquestes altures les metes a escollir són els límits.

```
1  public override void OnEpisodeBegin()
2  {
3    // reseting movement inercy
4    _rigidbody2D.velocity = Vector2.zero;
5    //reseting positions
6    transform.position = transform.parent.position;
7    // changin randomnes
8    Random.InitState(System.DateTime.Now.Millisecond);
9    // finding the moving target
10   FindMovingTarget();
11 }
12
```

5.2.1.3 OnActionReceived

La lògica dintre d'aquests mètode determina el comportament de l'Agent. Les accions que s'esperen és una continua, un valor entre menys u i més u. Segons aquesta acció la velocitat de l'Agent serà donada.

```
1  // called when action is received from either {player,
     neural network}
2  // each buffer position refers to an action, I decide what
      it means for each positions
3  // inside that structure has continuous and discrete
     actions
4  // index 0: -1 means move to the left, +1 means move to
     the right
5  // the cool thing about the neural network, is that it
     figures it all automatic
6  public override void OnActionReceived(ActionBuffers
     actions)
7  {
8    Vector2 movement = new Vector2(actions.ContinuousActions
     [0] * movementForce, 0);
9    _rigidbody2D.velocity = movement;
10 }
11
```

5.2.1.4 CollectObservations

En aquest mètode li he de donar al procés d'aprenentatge aquells paràmetres que jo crec que són importants per tenir en compte a l'hora d'aprendre la política.

Aquí vaig trobar interessant d'informar el procés d'aprenentatge, quina és la horientació del Agent respecte el punt objectiu. La distància de l'Agent respecte el punt objectiu, la posició del punt objectiu i finalment la velocitat de moviment de l'Agent.

```
1  // Should include all variables relevant for following
2  // to take the agent the optimally informed desition.
3  // No extraneous information here please
4  public override void CollectObservations(VectorSensor
     sensor)
5  {
6    if (!currentTarget) return;
7
8    Vector2 currentPos = new Vector2(transform.position.x,
     0);
9    Vector2 targetPos = new Vector2(currentTarget.position.x
     , 0);
10
11   Vector2 toTarget = targetPos - currentPos;
12   // 2 observations (horientation)
13   sensor.AddObservation(toTarget.normalized);
14   // 1 observation (distance)
15   sensor.AddObservation(Vector2.Distance(targetPos,
     currentPos));
16   // 2 observations for current target position
17   sensor.AddObservation(targetPos);
18   // 2 observations for movement velocity
19   sensor.AddObservation(_rigidbody2D.velocity);
20   // Note: curiosamente si normalizo la velocidad, le
     cuesta mucho aprenderx
21 }
22
```

5.2.1.5 Heuristic

En aquesta versió únicament l'Agent ha de poder interpretar valors entrats per teclat que representin moviments laterals.

Es modifica la referència dels *ActionBuffers* de tal manera que en el buffer de variables contínues és passa el valor entrat per teclat.

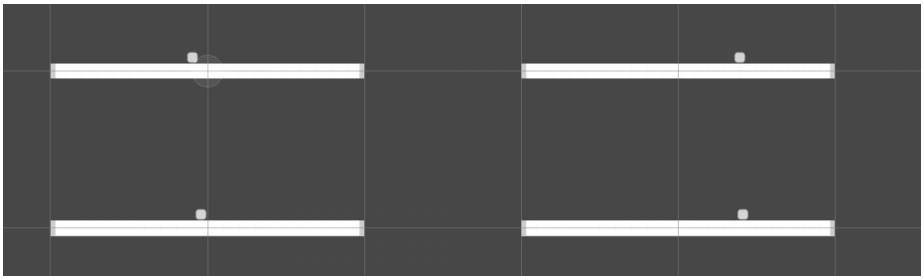
```
1  // this method allows me to interact with the game
2  // when the ml agents is not set to trainning
3  public override void Heuristic(in ActionBuffers actionsOut
     )
4  {
5    ActionSegment<float> continuosActions = actionsOut.
     ContinuousActions;
6    var force = Input.GetAxis("Horizontal");
7    continuosActions[0] = force;
8  }
9
```

5.3 Metodologia i Resultats

5.3.1 Autonomia en el moviment, versió (I)

L'objectiu de fer aprendre la política de moure's cíclicament ha sigut tot un èxit.

Perquè el temps d'aprenentatge es fes més curt, el que vaig fer va ser duplicar l'Agent quatre vegades.



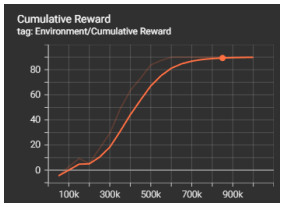
Fixem-nos en els següents logs que *mlagents-learn* va soltant en el procés d'aprenentatge.

```
[INFO] Resuming from results\ppo\NPC.
[INFO] Resuming training from step 896.
[INFO] NPC. Step: 50000. Time Elapsed: 149.945 s. Mean Reward: -4.500. Std of Reward: 13.955. Training.
[INFO] NPC. Step: 100000. Time Elapsed: 230.711 s. Mean Reward: 2.791. Std of Reward: 8.714. Training.
[INFO] NPC. Step: 150000. Time Elapsed: 303.819 s. Mean Reward: 9.286. Std of Reward: 5.517. Training.
[INFO] NPC. Step: 200000. Time Elapsed: 368.859 s. Mean Reward: 5.349. Std of Reward: 12.454. Training.
[INFO] NPC. Step: 250000. Time Elapsed: 447.106 s. Mean Reward: 17.317. Std of Reward: 13.619. Training.
[INFO] NPC. Step: 300000. Time Elapsed: 524.144 s. Mean Reward: 29.286. Std of Reward: 18.695. Training.
[INFO] NPC. Step: 350000. Time Elapsed: 593.135 s. Mean Reward: 48.537. Std of Reward: 13.715. Training.
[INFO] NPC. Step: 400000. Time Elapsed: 673.271 s. Mean Reward: 63.488. Std of Reward: 11.790. Training.
[INFO] NPC. Step: 450000. Time Elapsed: 754.590 s. Mean Reward: 73.171. Std of Reward: 9.223. Training.
[INFO] NPC. Step: 500000. Time Elapsed: 834.312 s. Mean Reward: 83.810. Std of Reward: 6.884. Training.
[INFO] Exported results\ppo\NPC\NPC-499932.onnx
[INFO] NPC. Step: 550000. Time Elapsed: 914.108 s. Mean Reward: 87.317. Std of Reward: 4.431. Training.
[INFO] NPC. Step: 600000. Time Elapsed: 993.981 s. Mean Reward: 90.000. Std of Reward: 0.000. Training.
[INFO] NPC. Step: 650000. Time Elapsed: 1074.479 s. Mean Reward: 90.000. Std of Reward: 0.000. Training.
[INFO] NPC. Step: 700000. Time Elapsed: 1152.584 s. Mean Reward: 90.000. Std of Reward: 0.000. Training.
[INFO] NPC. Step: 750000. Time Elapsed: 1233.012 s. Mean Reward: 90.000. Std of Reward: 0.000. Training.
[INFO] NPC. Step: 800000. Time Elapsed: 1312.612 s. Mean Reward: 90.000. Std of Reward: 0.000. Training.
[INFO] NPC. Step: 850000. Time Elapsed: 1394.562 s. Mean Reward: 90.000. Std of Reward: 0.000. Training.
[INFO] NPC. Step: 900000. Time Elapsed: 1546.801 s. Mean Reward: 90.000. Std of Reward: 0.000. Training.
[INFO] NPC. Step: 950000. Time Elapsed: 1628.662 s. Mean Reward: 90.000. Std of Reward: 0.000. Training.
[INFO] NPC. Step: 1000000. Time Elapsed: 1706.933 s. Mean Reward: 90.000. Std of Reward: 0.000. Training.
```

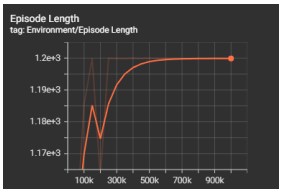
Cada una de les entrades del log representen episodis. Les primeres entrades donen una mitja *Mean Reward* que les últimes i la desviació estàndard és bastant més ample que les últimes.

Finalment arriba el moment on l'aprenentatge de l'Agent no canvia. Això vol dir que ha après una política i immediatament, *mlagents-learn* quan veu que la mitja per cada episodi és la mitja tanca el procés d'aprenentatge.

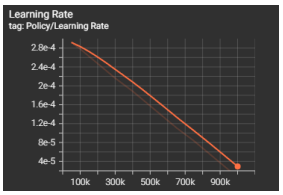
A continuació veurem algunes gràfiques que *Tensorboard* ens ha generat respecte a aquest procés d'aprenentatge.



(a) Comulative reward

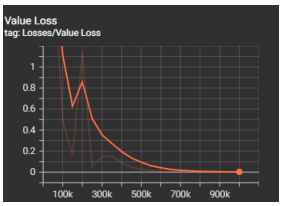


(b) Episodi length

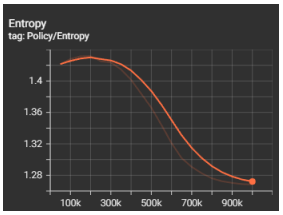


(c) Learning rate

(d) Policity loss



(e) Value Loss



(f) Entropy

6 Conclusions i línies futures

7 Referències