Aprenentatge per reforç amb Unity

TAIA Project

Wilber Eduardo Bermeo Quito

11 de novembre de 2021

Índex

1 Problema a resoldre en una frase

Fer que un NPC segueixi una política en concret fent us 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 resumint-ho, 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 es 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

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

Coneixo l'entorn de treball Unity, no a nivell professional, però si tinc un base.

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

4 Proposta

4.1 Agent

Classe que conte 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 esta relacionat amb un Behavior Parameter class i a un Decision Requester.

4.1.1 Como 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 i/o visuals. Les observacions numèriques són del punt de vista del observador (NPC), les visuals en canvi, són generades per col·lisións de rajos que es poden ajuntada al agent i representen el que l'agent esta veient en aquell mateix moment.

4.1.1.2 Accions

Lo 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

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

4.2 Politiques a aprendre

Els següents dos punts son els problemes a modelitzar perquè l'Agent aprengui la politica adecuada per compliar-los.

4.2.1 Autonomia en el moviment

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

Arribar al màxim rang de desplaçament tant d'esquerra a la dreta ha de donar feetback positiu, però no ha de passar que s'és quedi en un extrem sense moure, només s'ha de poder atorgar feetback positiu si 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 hasgi apres a moure's, la meva idea es que l'NPC detecti entitats a atacar.

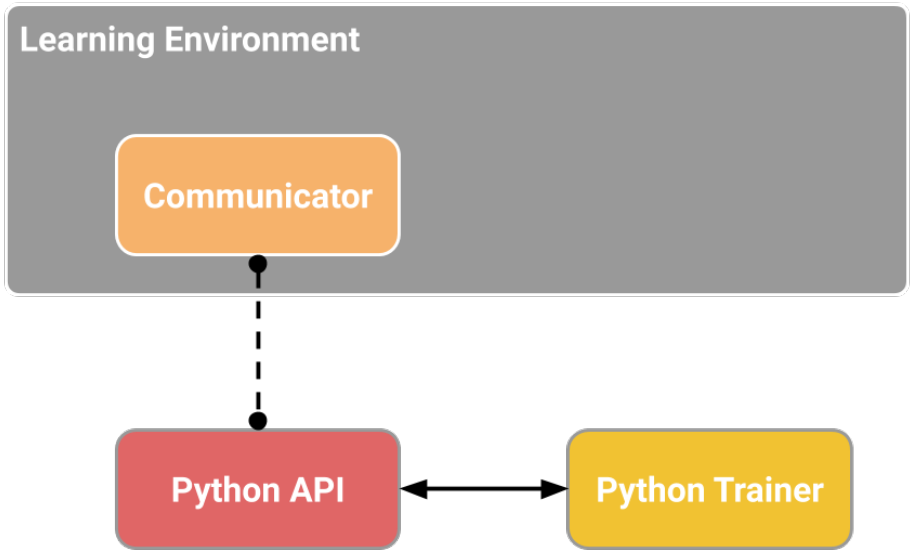
Aquestes entitats únicament seran detectables si estan dintre d'un rang esfèric que tindrà com radi el rang de desplaçament del NPC. L'objectiu en aquest cas es que, deixi de fer el moviment cíclic d'esquerra detrac i que salti sobre l'objectiu fent col·lisió, les col·lisions encertades donaran feetback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

4.3 Arquitectura, Algorismes, Propostes existents

La arquitectura del problema que abordo be 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 estadistiques d'aprenentage.

A continuació explicaré com es l'arquitectura de ML-Agents Toolkit. ML-Agents Toolkit te els següents quatre blocks 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 esta l'escena d'Unity i els personatges. Els personatges que per requeriment del problema s'han definits com Agents, podrán observar, prendre desicions i tenir recompenses.

4.3.2 Agent

Objecte d'Unity que hereta de la clase *Agent* amb metodes que poden ser sobrescrits.

La seva API contempla mètodes per poder generar observacions del medi, per prendre accions i per assignar recompenses.

Cada Agent esta relacionat amb un Behavior Parameter class i a un Decision Requester.

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* esta en mode *Learning* vol dir que utilitza el model generat per la sobrescritura dels metodes 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, s'utilitza la funcio *heuristic* de la clase *Agent* per interacturar amb l'Agent.

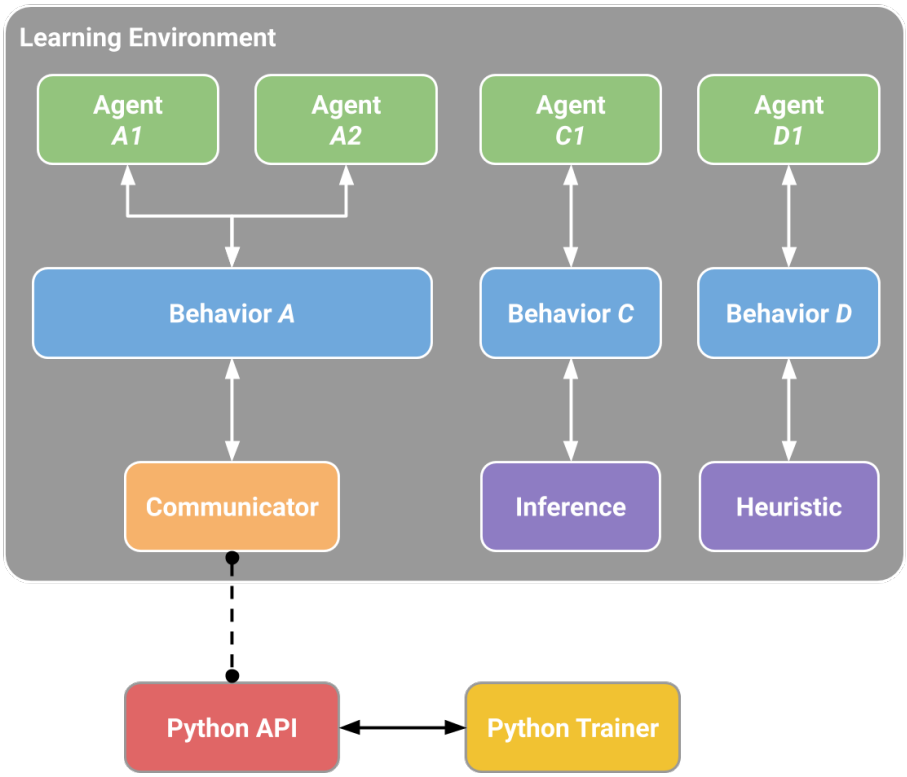
4.3.3.3 Inference

Quan el *Behavior* esta en mode *Inference* implica que hi ha una xarxa neuronal amb una politica apresa, i l'objecte que conte aquesta xarxa neuronal es comportarà segons lo que s'hagi apres.

4.3.4 Requester

Es 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.



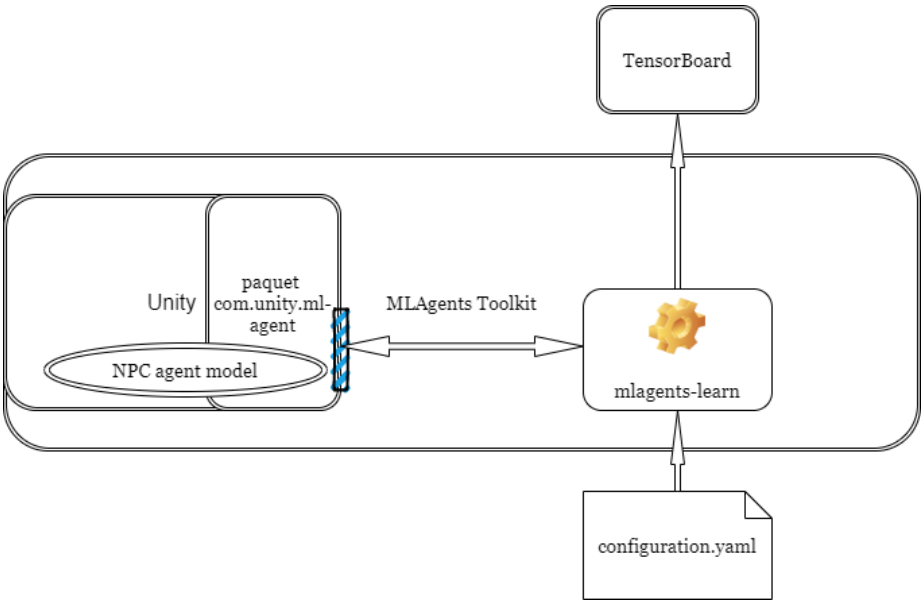
4.3.5 Tensorboard

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

Per observar el procés d'entrenament quan esta 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.

Arquitectura final



5 Experiments

5.1 Metodologia i Resultats

6 Conclusions i línies futures

7 Referències