LA HORDE

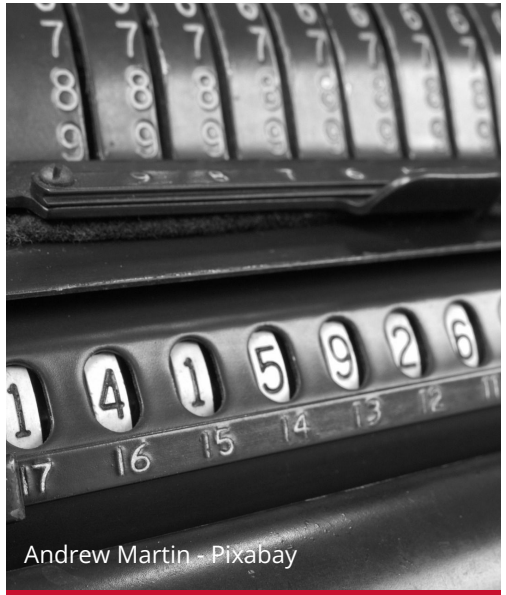
Gerd Altmann - Pixabay

Algorithmes d’apprentissage

- Apprentissage par renforcement
  - Dans quels cas ?
  - Quel fondement ?
  - Un exemple ?

ICEBREAKER

Algorithmes d’apprentissage


Andrew Martin - Pixabay

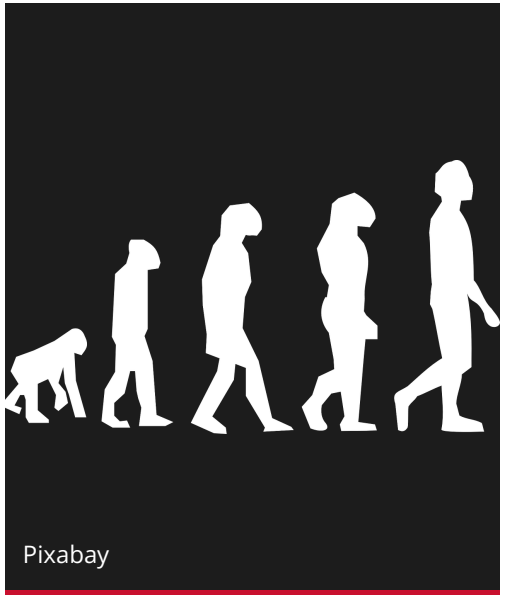
28/10/2022

Histoire de l’Intelligence Artificielle


Reto Scheiwiller - Pixabay

16/11/2022

Pratique : Données et algorithmes d’apprentissage


Pixabay

22/02/2023

Darwin et les algorithmes génétiques


Pavel Danilyuk - Pexels

24/03/2023

L’apprentissage par renforcement et applications

Module Algorithmes d’Apprentissage

L’apprentissage par renforcement et applications

Comprendre le paradigme de l’apprentissage par renforcement

Richard Sutton

Richard Sutton est un informaticien et enseignant canadien considéré comme un pionnier de l'**apprentissage par renforcement**.

Il est notamment connu pour ses travaux sur le **temporal difference learning** et son travail sur les **gradients**.

Publications: Reinforcement Learning: An Introduction (1998), Neural Networks for Control (1991), Reinforcement Learning. Reprinting of a special issue of Machine Learning Journal(1992)

01

Illustrations réelles

Point de vue biologique :

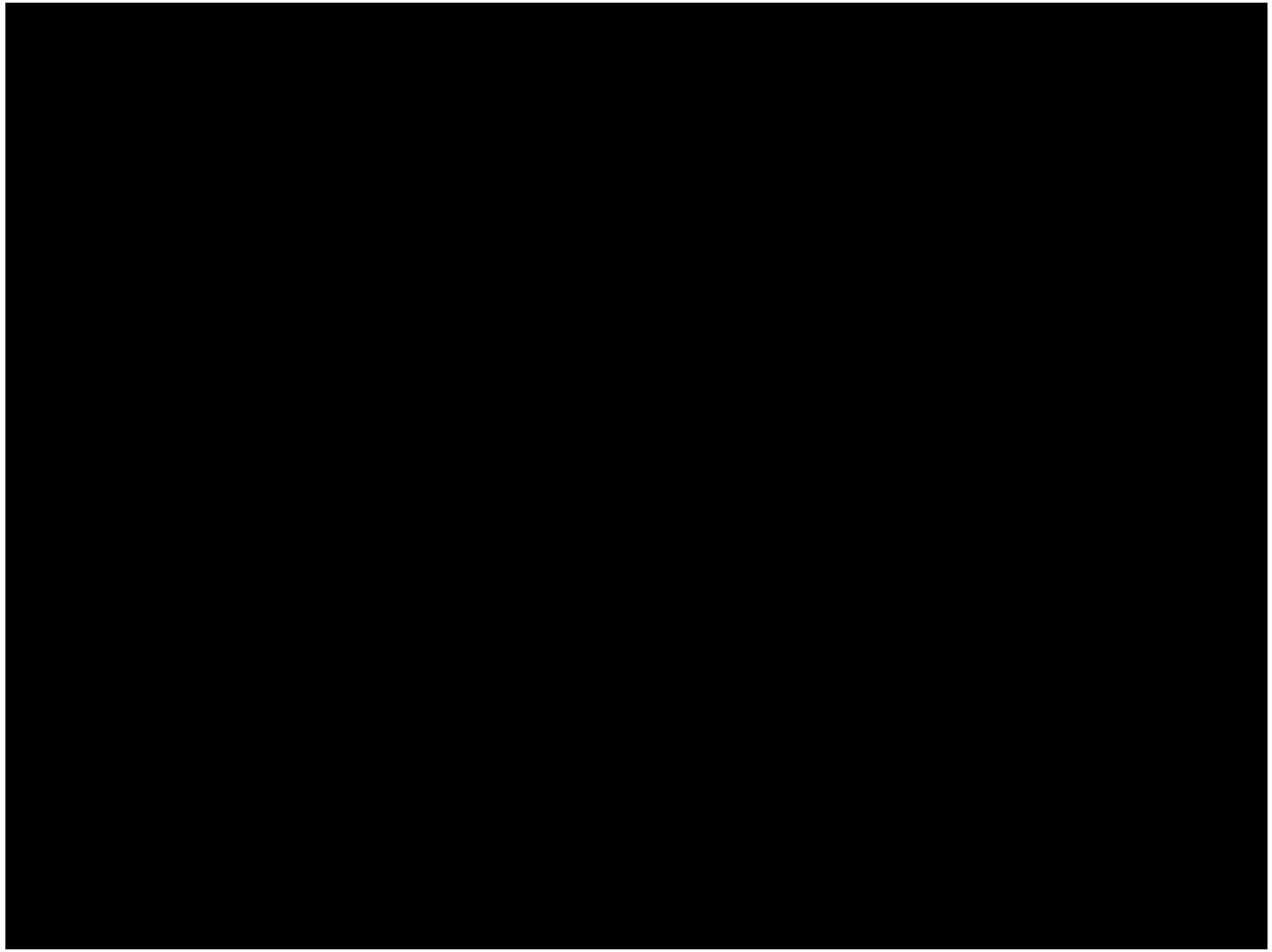
Boîte de Skinner

Point de vue biologique : *Pavlov*

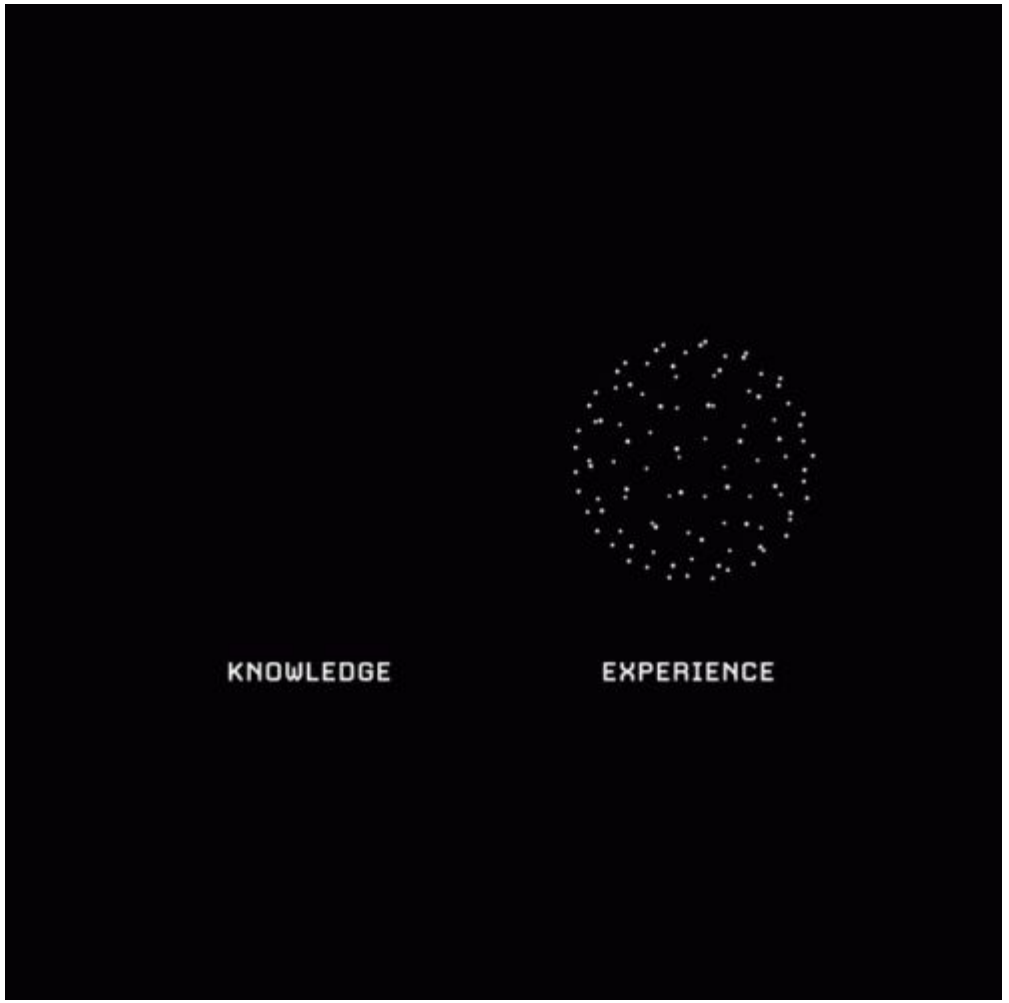
Point de vue gaming : *Atari*

Point de vue animation : *cinématique complexe*

Point de vue animation : *cinématique complexe*



Point de vue abstrait



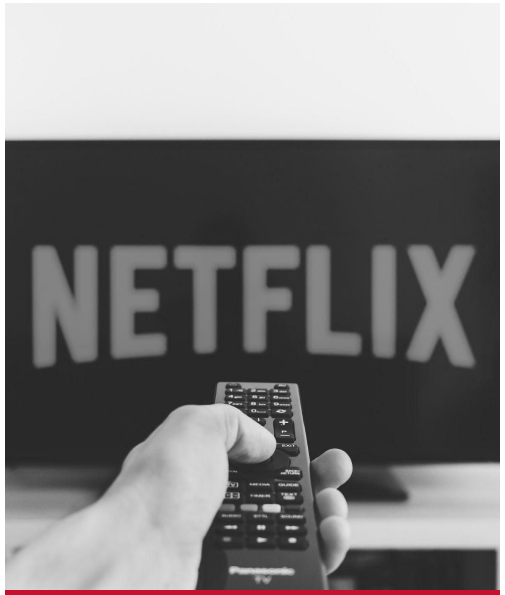
02

Cas d’usage

Une idée ?

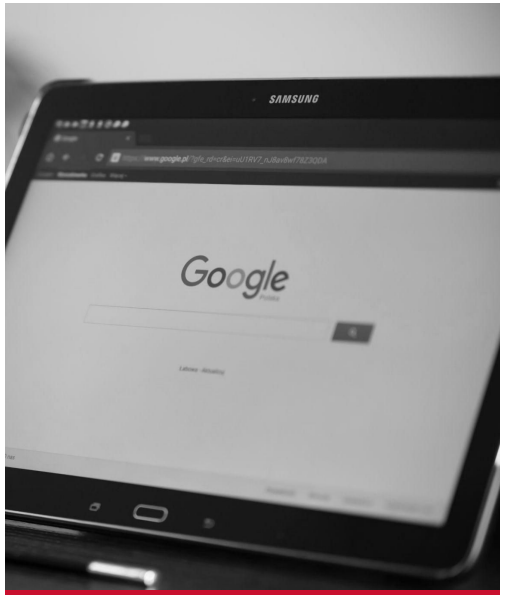
Exemples

Conduite autonome



Recommandations Netflix

Gestion d’inventaire



Moteurs de recherche

Exemples

RLHF, webGPT

Commerce d’instruments financiers

Robotique

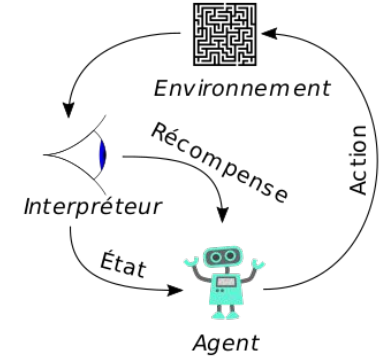
Cinématique

03

Théorie

Origines

1. Sutton : TD Learning
2. Watkins : Q-Learning
3. DeepMind : Deep Q Learning *(atari)*



Reinforcement Learning

1. Environnement
2. Agent
3. Actions
4. Récompenses

Reinforcement Learning

Peut être divers et partiellement ou totalement observable :

- salle de poker,
- vue du monde
- monde virtuel …

Environnement

Reinforcement Learning

Peut être divers, généralement représenté par la politique d'actions qu'il cherche à optimiser

- f(état)=action
- où f peut être un réseau de neurone

Agent

Reinforcement Learning

Les actions peuvent être finies, ou bien très nombreuses.
Elles permettent à l'agent d'interagir avec l'environnement en passant de l'état **t** à l'état **t+1**.

Actions

Reinforcement Learning

Il résulte de chaque action, des récompenses ou punitions dont l'agent tire des leçons. L'agent affine sa politique d'action / son comportement / son schéma de réflexion grâce à ces dernières.
Il existe un compromis entre récompense à **court-terme** et à **long terme.**

Récompense

Reinforcement Learning

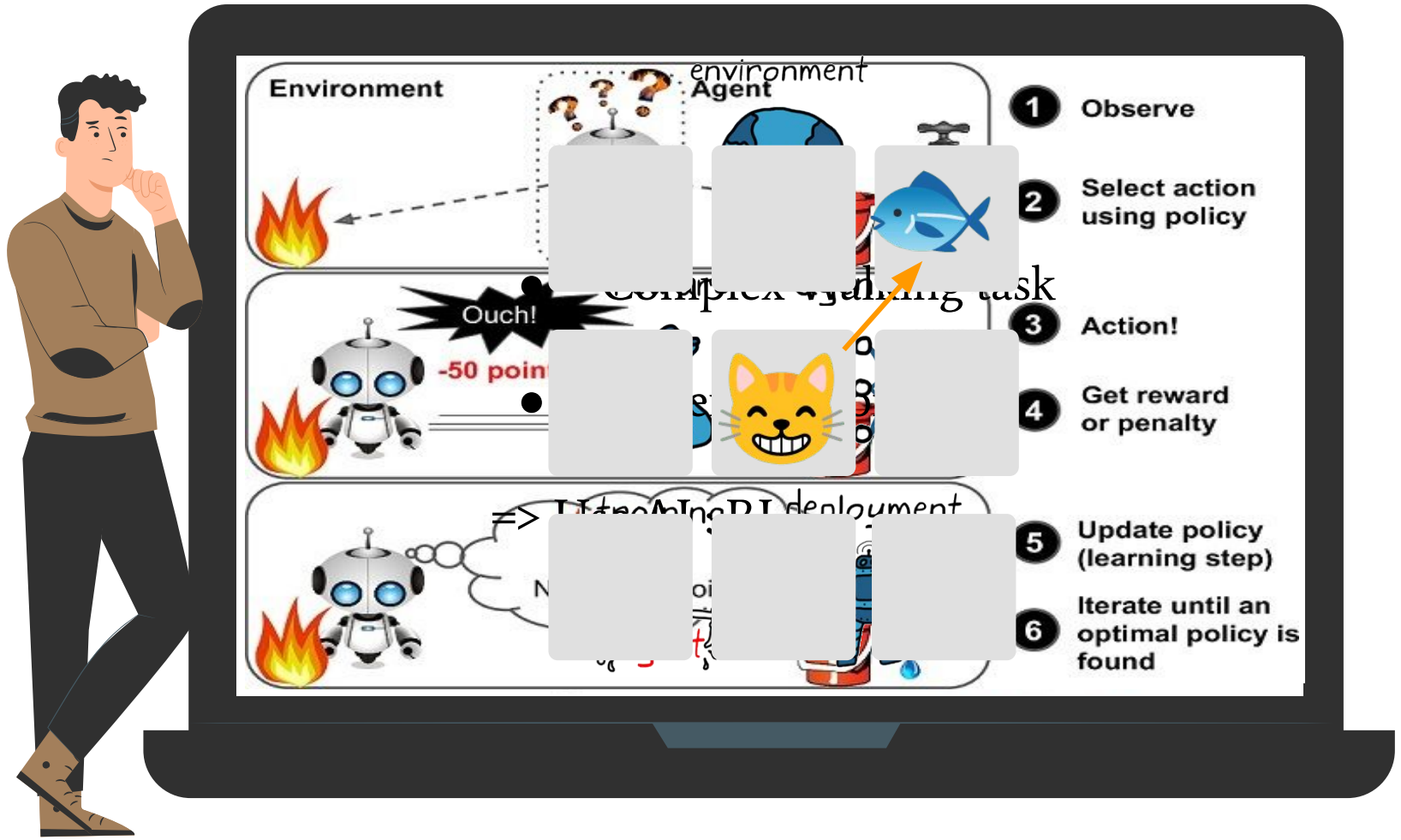
Il permettra d'affiner la politique de choix d'action en fonction de l'état de l'environnement et de l'agent.

On en discerne 2 :

- **on-policy**
- **off-policy**

Algorithme d'apprentissage

Discount rate



Algorithmes d’apprentissage par renforcement

Q-Learning

Le Q-learning converge vers une stratégie optimale, c'est-à-dire qu'il conduit à maximiser la récompense totale des étapes successives.

Il est indépendant de l’environnement.
off-policy

algorithme d’apprentissage par renforcement

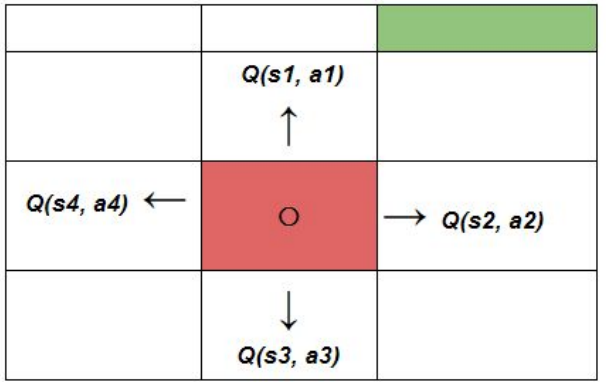
Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	327	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	499	0	0	0	0	0	0

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

La fonction de valeur action-état dans une table est initialisée à 0 pour chaque couple action-état. Chaque cellule est mise à jour pendant l'exécution de l'algorithme Q-learning.



Q-Learning

L'algorithme calcule une fonction de valeur action-état :

$$Q : S \times A \to \mathbb{R}$$

$$Q[s,a] := (1-\alpha)Q[s,a] + \alpha\left(r + \gamma \max_{a'} Q[s',a']\right)$$

α : facteur d'apprentissage

γ : facteur d'actualisation

α : facteur d'apprentissage

γ : facteur d'actualisation

$$Q[s,a] := (1-\alpha)Q[s,a] + \alpha\left(r + \gamma \max_{a'} Q[s',a']\right)$$

Q-Learning

```
entrée : taux d'apprentissage α > 0 et un petit ε > 0, taux γ > 0
sortie : tableau Q[., .]

initialiser Q[s, a] pour tout état s non final, toute action a de façon arbitraire,
        et Q(état terminal, a) = 0 pour toute action a

répéter
      //début d'un épisode
      s := état initial

      répéter
              //étape d'un épisode
              choisir une action a depuis s en utilisant la politique spécifiée
                      par Q (par exemple ε-greedy)
              exécuter l'action a
              observer la récompense r et le nouvel état s'

              Q[s, a] := Q[s, a] + α[r + γ max_a' Q(s', a') - Q(s, a)]

              s := s'
        jusqu'à ce que s soit l'état terminal
```

Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	327	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	499	0	0	0	0	0	0

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

La fonction de valeur action-état dans une table est initialisée à 0 pour chaque couple action-état. Chaque cellule est mise à jour pendant l'exécution de l'algorithme Q-learning.

SARSA

$$Q[s,a] := (1-\alpha)Q[s,a] + \alpha\,(r + \gamma Q[s',a'])$$

SARSA converge vers une stratégie optimale, c'est-à-dire qu'il conduit à maximiser la récompense totale des étapes successives.

Il est indépendant de l'environnement.
on-policy

algorithme d'apprentissage par renforcement

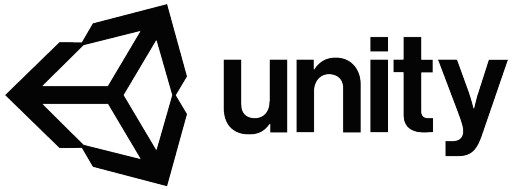
04

Retour d’expérience sur projet

Unity ML Agent et Machine Learning

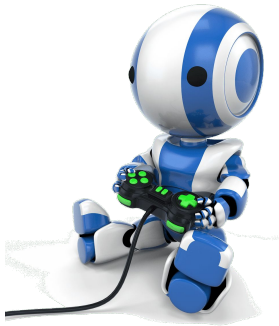
Learning

Interface



Unity, C#

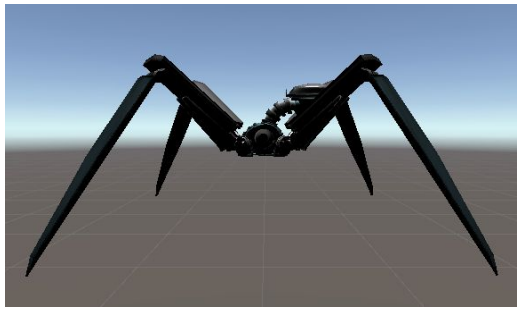
3D

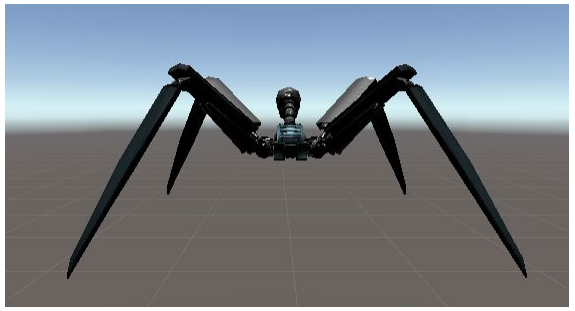


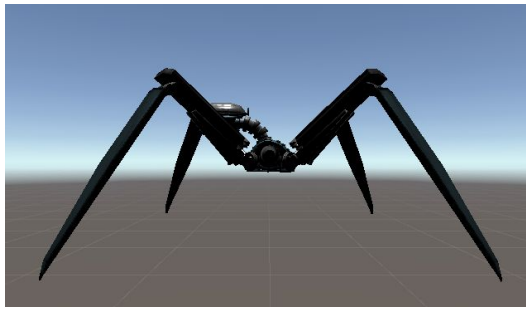
Reinforcement Learning

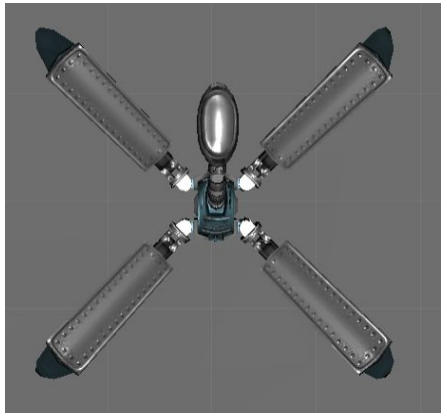
AI

The monster 🎃

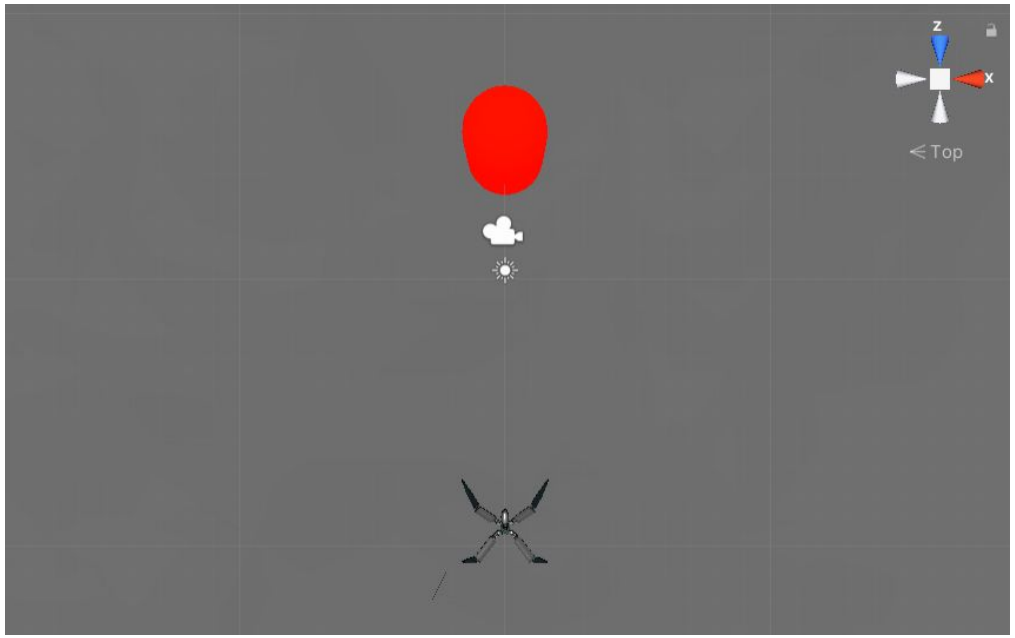


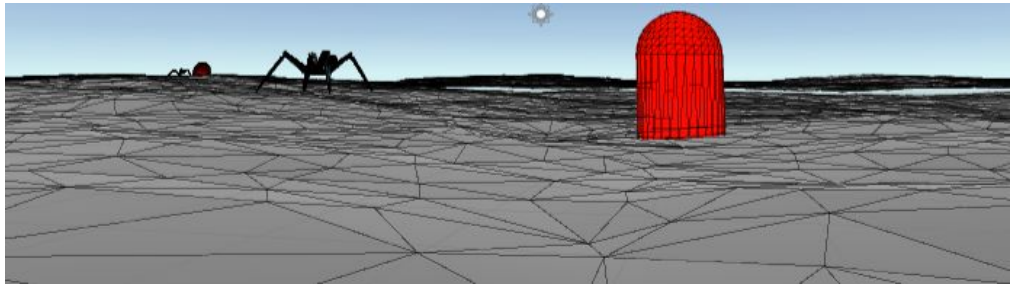


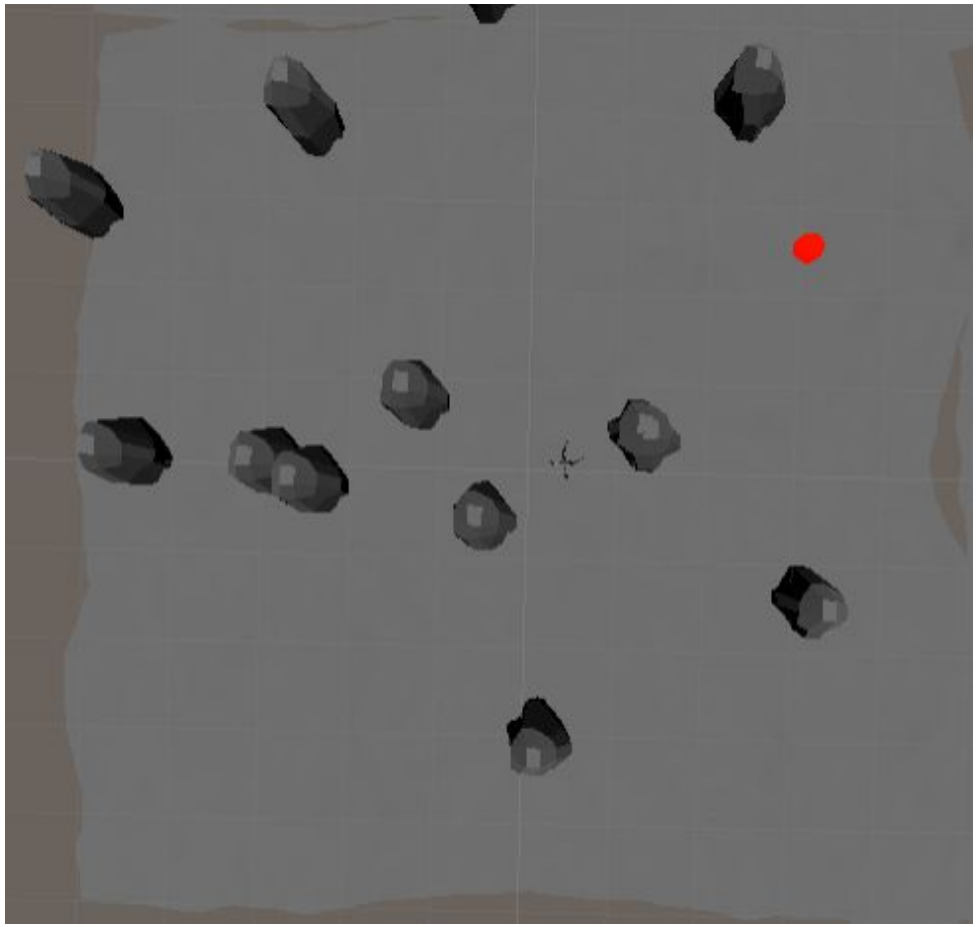




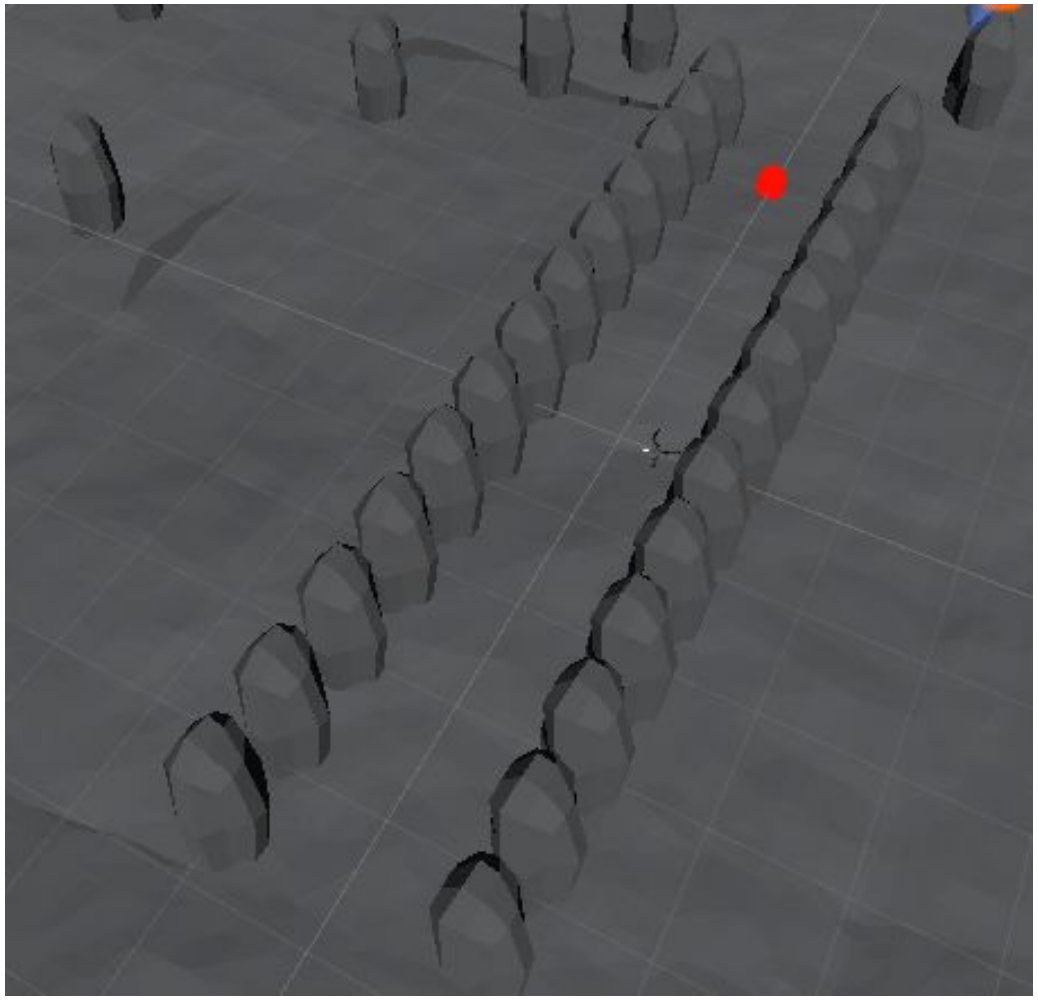
Its playground😉







Some epochs

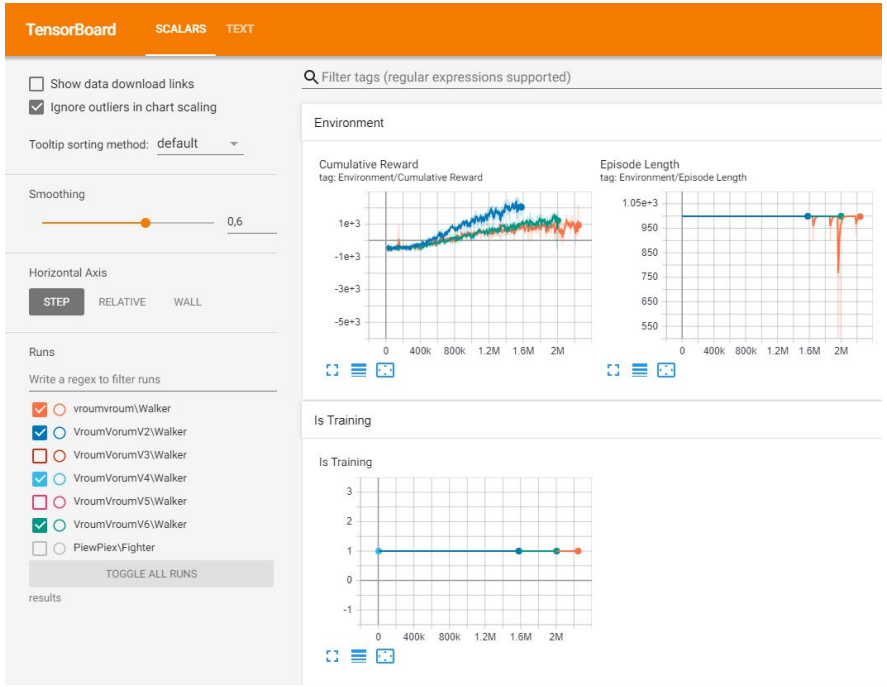


Its training 😉

```
mlagents-learn config/trainer_config.yaml --run-id=Fighter
```

Terminal command
mlagents-learn

Connect to WalkTrainer
Unity



Its training 😉

mlagents-learn

Anaconda Powershell Prompt (Anaconda3)

```
(base) PS C:\Users\asus\Desktop\ProjetSemestriel\Naksnineiiit> mlagents-learn config/trainer_config.yaml --run-id=walker0

2021-01-12 09:54:15.642934: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic l
ibrary cudart64_101.dll
WARNING:tensorflow:From c:\programdata\anaconda3\lib\site-packages\tensorflow\python\compat\v2_compat.py:96: disable_reso
urce_variables (from tensorflow.python.ops.variable_scope) is deprecated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```


```
Version information:
 ml-agents: 0.22.0,
 ml-agents-envs: 0.23.0.dev0,
 Communicator API: 1.2.0,
```

wait for Unity to start simulation

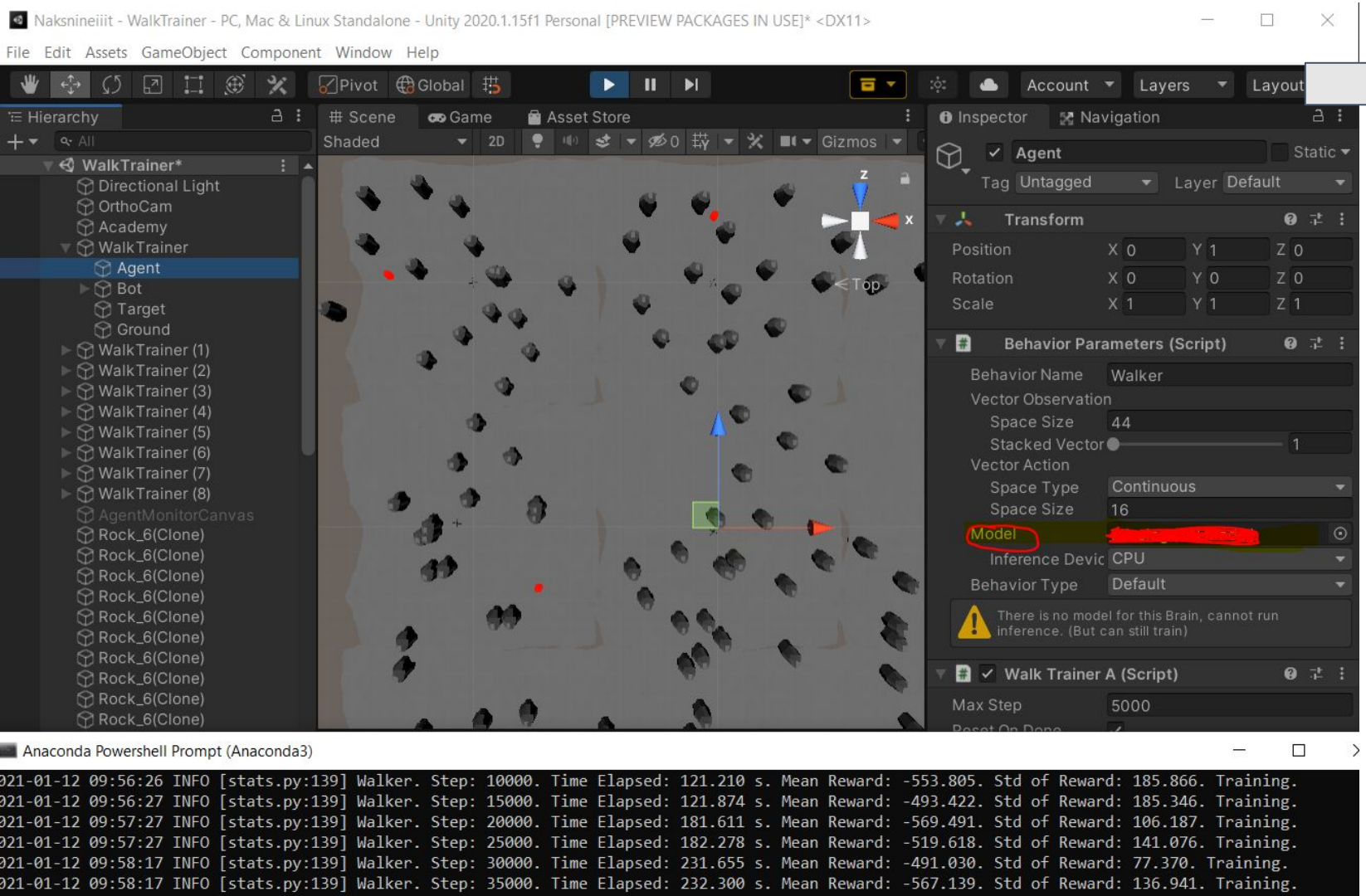
Establish link between :

- Python AI RL
  - TensorFlow
  - PyTorch
- Unity 3D
  - Simulation

Its training 😉

Unity WalkTrainer

```
2021-01-12 10:10:41 INFO [trainer_controller.py:85] Saved Model
```



Loop for a predefined number of steps

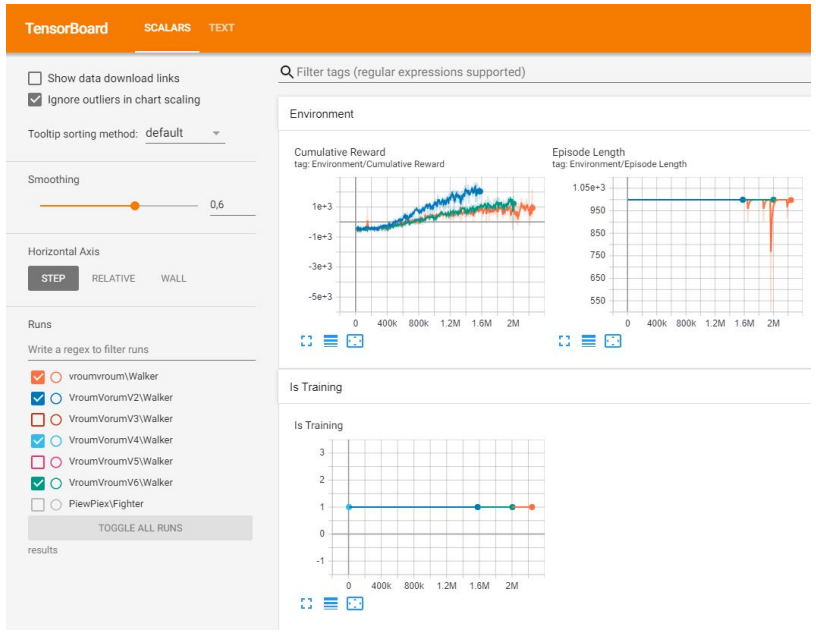
For each epoch :

- nb of steps,
- time elapsed
- Mean and Std of Reward

Save our model at regular checkpoints

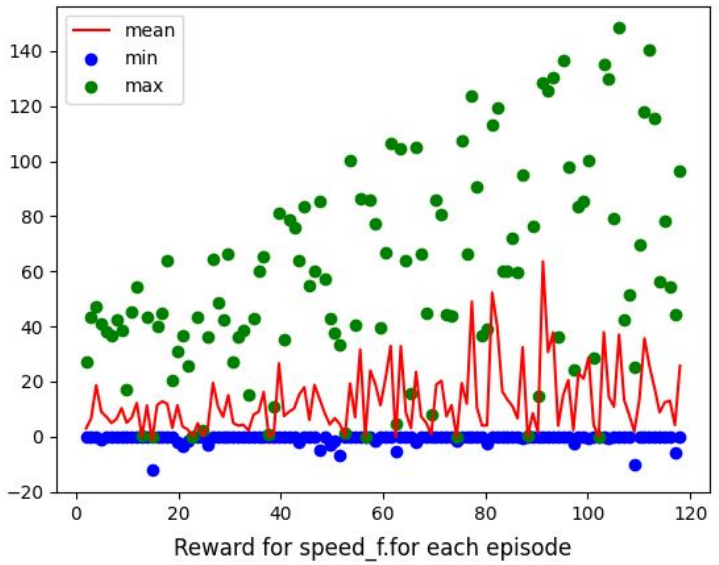
Its training 😉

TensorBoard



See **Cumulative reward** on our model

- To validate or
- reject it



Graph 2 : local stats

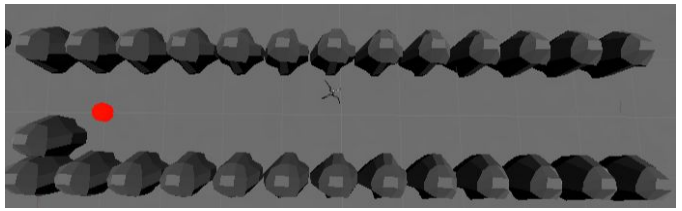
See **Each reward separately**

- To validate or
- reject it

Upgrades pushed to the initial project

- Reward engineering => understand its impact, reduct angle weight in walking

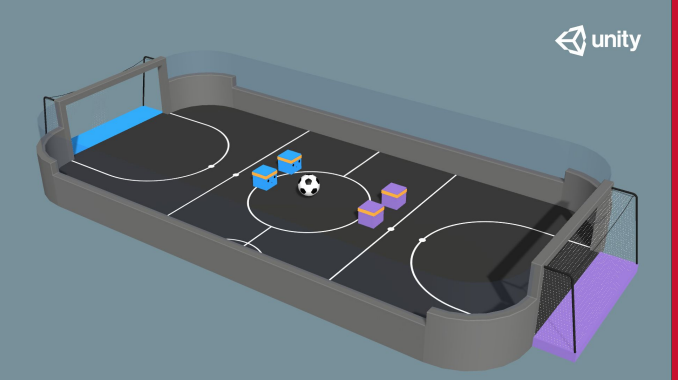
- Design of specific playground for training :
  - Learn simple tasks : walk forward, walk behind …
  - And do some transfer learning over pre trained model
  - Rearrangement the base training ground to add obstacle

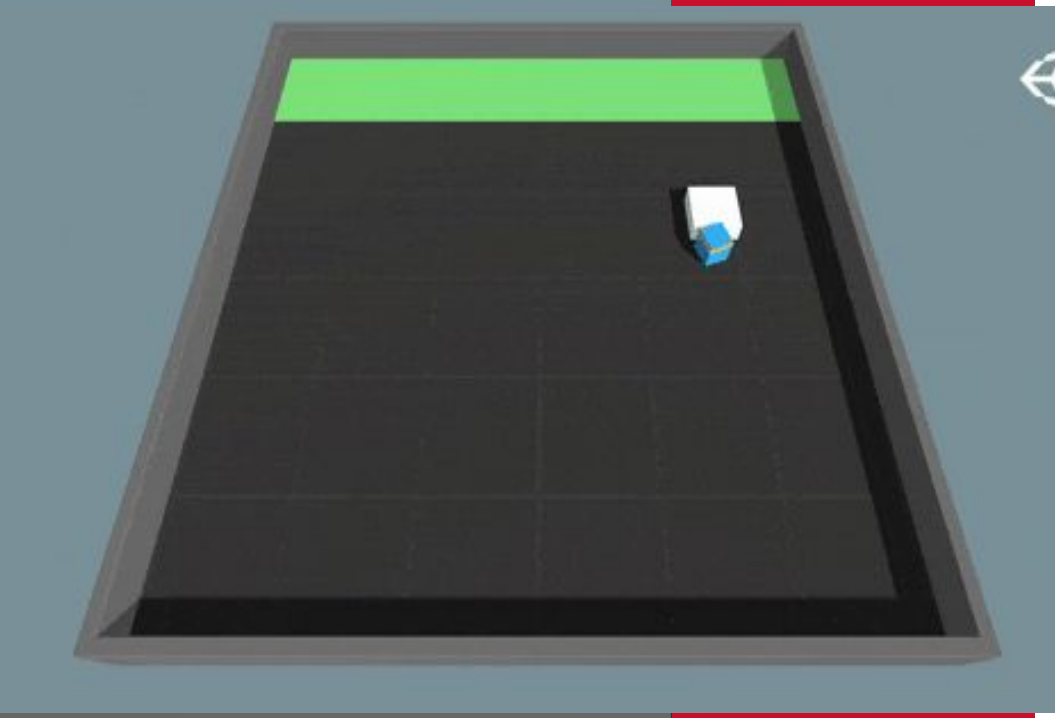


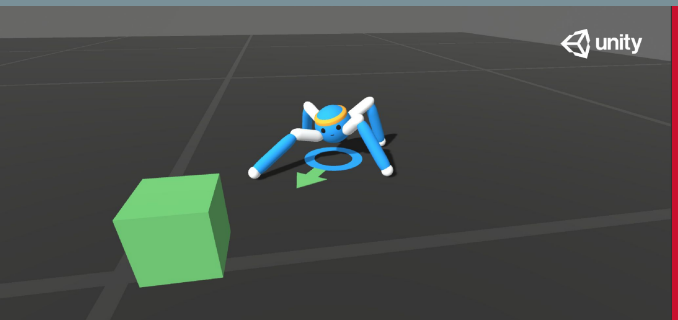
- Addition of heath bars and deaths to the bots.

05

Configuration & exploration ml-agents







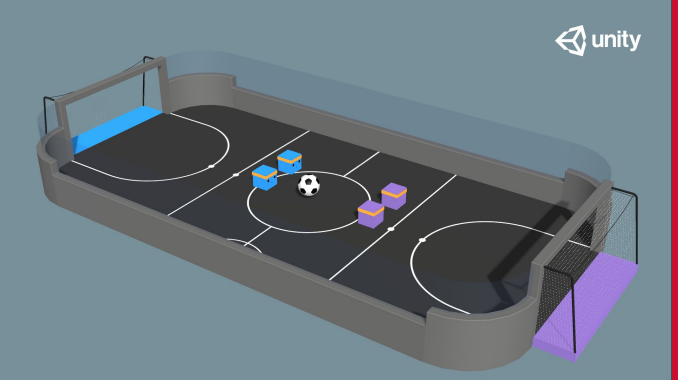
A vos claviers ;)

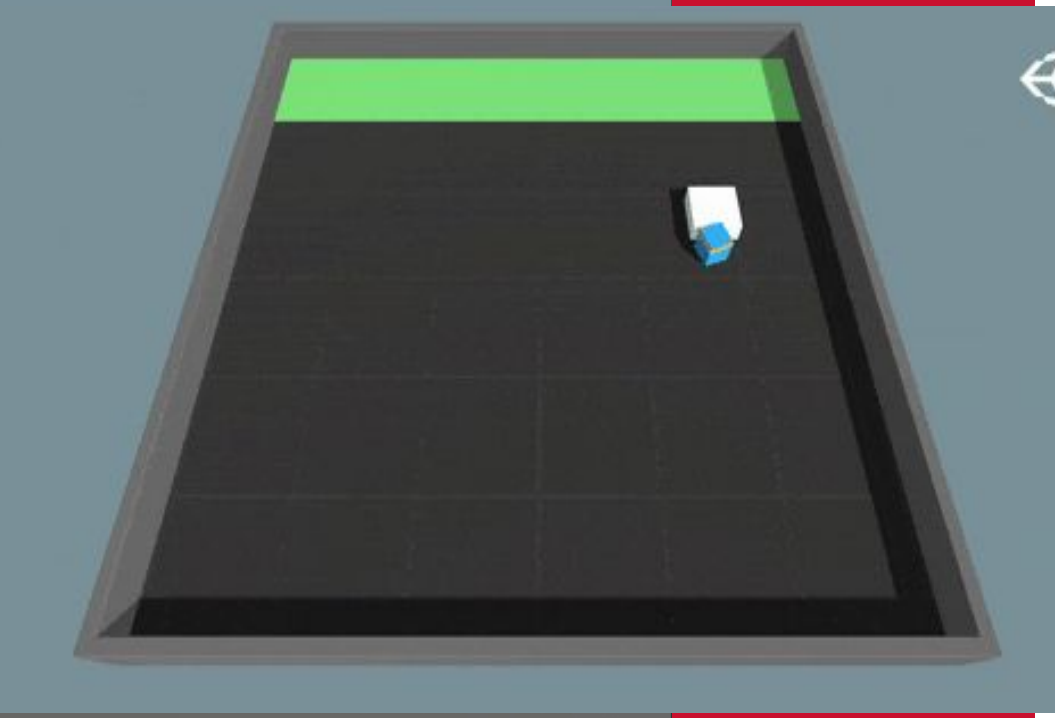
1. https://github.com/Unity-Technologies/ml-agents/blob/release_18_docs/docs/Learning-Environment-Examples.md

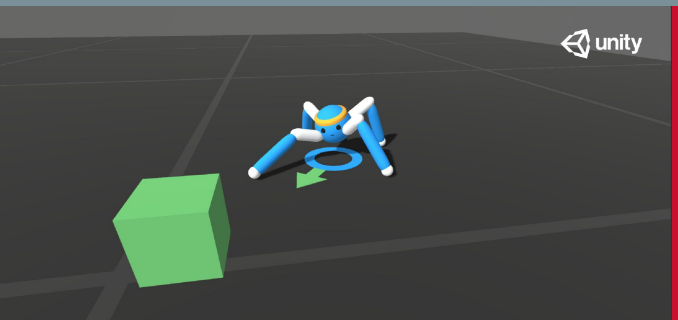
2. git clone --branch release_18 https://github.com/Unity-Technologies/ml-agents

3. Package manager > Add package from disk

4. Entraînez un agent sur le projet PushBlock
   https://github.com/Unity-Technologies/ml-agents/tree/release_18_branch/Project/Assets/ML-Agents/Examples/PushBlock/Scenes/PushBlock.unity







A vos claviers ;)

Lancer l'entrainement coté mlagents:

- mlagents-learn config/ppo/PushBlock.yaml --run-id=test_0t

Visualiser l'entrainement :

- tensorboard --logdir results

Récupérer notre **réseau de neurones** après entraînement :

- *results/pb_01/PushBlock.onnx*

L'ouvrir dans **Netron** pour le visualiser ;)

Merci

Présentation réalisée par Quentin Pestre-Sorge
quentin@modernov.com