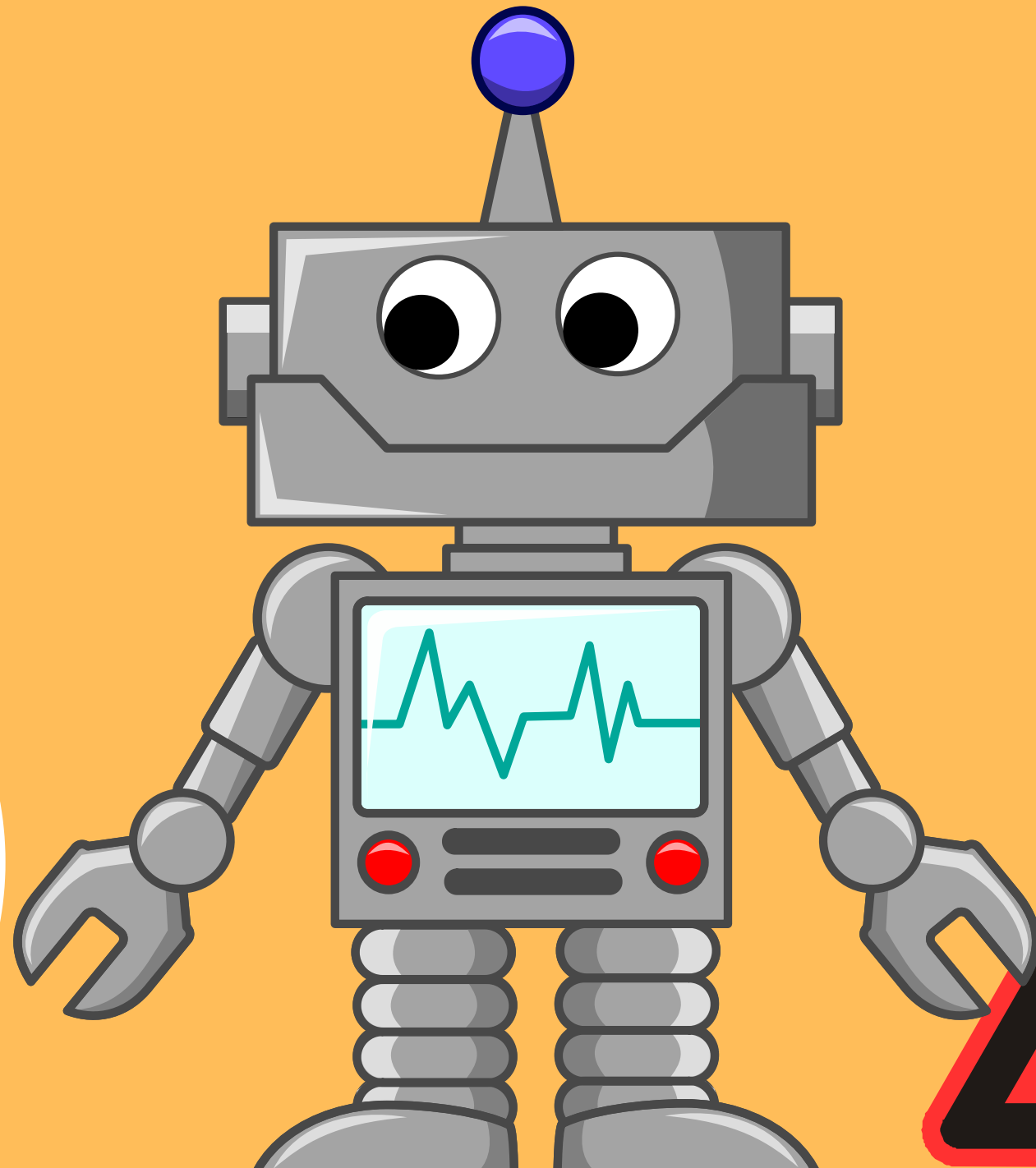
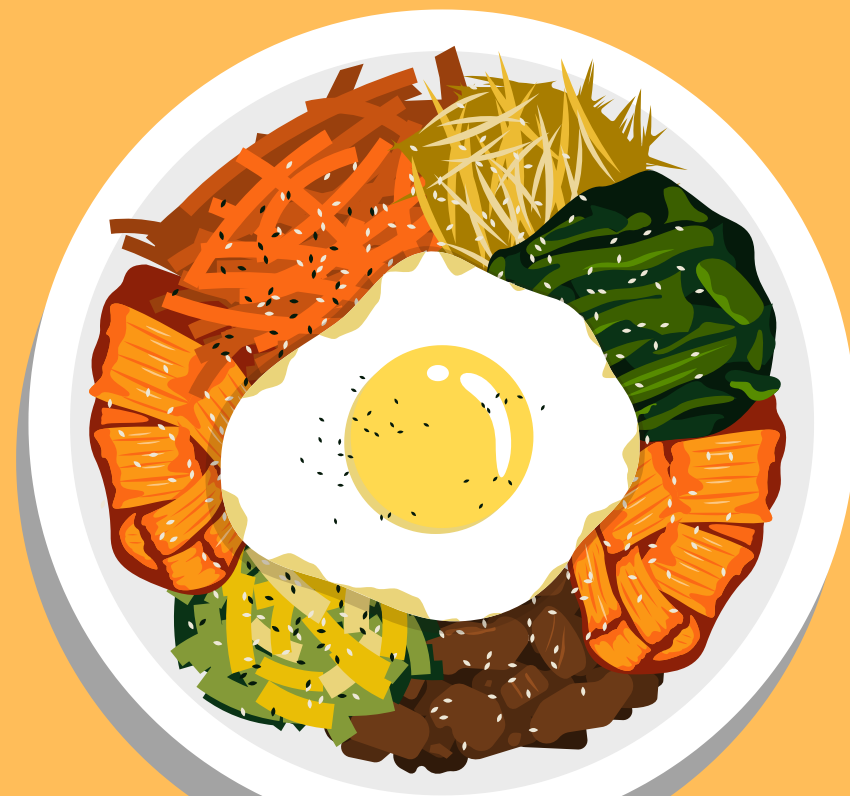


FOODOR

POISON




Projet : Apprentissage par renforcement

Présenté par : SOMBIE Bibata

Intervenant : Sylvain LAPEYRADE

Objectif

Créer un environnement personnalisé de type "jeu de survie" où un agent apprend à éviter le poison (ennemi) et manger de la nourriture à l'aide d'algorithmes de renforcement.

- 
- ▶ Environnement Pygame personnalisé
 - ▶ Agent Q-learning et PPO (Stable-Baselines3)
 - ▶ Mode Joueur humain et Mode Agent autonome
 - ▶ Interface graphique interactive

Principe du jeu

Présentation du jeu "Food or Poison"

Food or Poison est un mini-jeu Python où un robot doit manger pour gagner des points tout en évitant le poison.

Chaque déplacement coûte une pénalité, il faut donc choisir le chemin le plus court.

Après 4 poisons, la partie est perdue.

Après 5 food , niveau superieur

Ce jeu simple vise à être amélioré et peut aider les enfants à développer leur logique et leur compréhension du système de récompense.

Dexcription du Projet

⚙️ Tableau de Caractéristiques des Environnements

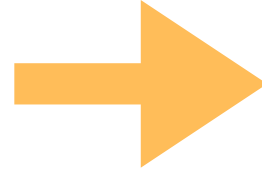
Caractéristique	Environnement PPO (environnement_ppo.py)	Environnement Q-Learning (train_qlearning.py)
Type d'interface	Basé sur gym.Env (Gymnasium)	Script personnalisé sans classe gym.Env
Grille	15x15 sur une surface de 600x600 px	15x15 sur une surface de 600x600 px
Actions	spaces.Discrete(4) (haut, bas, gauche, droite)	Implémentées manuellement dans QLearning()
Observation (état)	Différences de position entre le joueur et les deux autres blobs (vecteur de 4 valeurs)	Position discrétisée via binning()
Récompense positive	+250 si le joueur mange la nourriture	+250 (ou similaire, défini dans QLearning)



Caractéristique	Environnement PPO (environnement_ppo.py)	Environnement Q-Learning (train_qlearning.py)
Récompense négative	-250 si le joueur touche l'ennemi	-250 (ou similaire, défini dans QLearning)
Fonction reset()	Oui (définit les blobs à chaque épisode)	Non explicite, géré à l'intérieur de QLearning()
Agent	Entraîné avec PPO via Stable-Baselines3	Q-table manuelle entraînée par exploration
Utilisation de Blob	blob_base2.py avec collisions graphiques	blob_base.py avec logique manuelle
Affichage	Rendu Pygame déclenché par une interface externe (play_or_watch.py)	Affichage direct intégré à la boucle principale

Dexcription du Projet

Q-learning : table de Q-valeurs
PPO : Proximal Policy Optimization ?



- Q-learning est un algorithme de renforcement qui apprend une table d'état-action pour maximiser les récompenses futures, sans avoir besoin de connaître l'environnement à l'avance.
- PPO est un algorithme de renforcement basé sur une politique, qui améliore la stabilité de l'apprentissage en limitant les mises à jour brusques de l'agent, souvent utilisé avec des réseaux de neurones via Stable-Baselines3.

Dexcription du Projet



Tableau de fonctionnement des agents

Élément	Agent PPO	Agent Q-learning
Type d'algorithme	Apprentissage par politique (policy-based)	Apprentissage par valeurs (value-based)
Bibliothèque utilisée	Stable-Baselines3	Aucune (implémentation manuelle)
Type d'agent	Réseau de neurones	Table de Q-valeurs
Exploration / Exploitation	Gérée automatiquement par PPO	Gérée via epsilon-greedy
Structure d'entrée (état)	Vecteur de 4 valeurs (distances relatives)	États discrets (position binning)
Actions possibles	Haut, Bas, Gauche, Droite (4 actions)	Identiques
Apprentissage	Optimisation d'une politique par gradient	Mise à jour de la Q-table
Récompenses	+250 (nourriture), -250 (ennemi)	+1 (nourriture), -1 (ennemi)
Environnement	Basé sur gym.Env	Boucle de jeu personnalisée
Vitesse d'entraînement	Plus lent mais plus robuste	Rapide mais sensible à l'exploration
Visualisation	play_or_watch.py	test_Agent_Qlearning.py
Sauvegarde du modèle	Fichier .zip contenant le modèle PPO	Fichier .npy ou .txt pour la Q-table
Utilisation recommandée	Projets complexes, grande échelle	Projets éducatifs, environnements simples

Dexcription du Projet

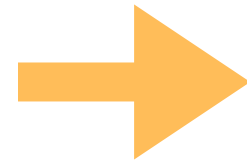
Les Classes principales du projet

Classe	Fichier	Rôle / Description
Blob	blob_base.py / blob_base2.py	Représente une entité mobile sur la grille (joueur, nourriture ou ennemi). Gère la position
BlobEnv	environnement_ppo.py	Environnement personnalisé compatible avec Gymnasium. Définit l'état, les actions possib
QLearning	train_qlearning.py	Implémente l'algorithme Q-learning : boucle d'apprentissage, mise à jour de la Q-table, po
PPO	Stable-Baselines3 (lib externe)	Agent basé sur un réseau de neurones. Utilise l'environnement BlobEnv pour apprendre u
gym.Env	gymnasium (hérité dans BlobEnv)	Classe de base de Gym pour créer un environnement d'apprentissage par renforcement.
spaces.Discrete / spaces.Box	gymnasium.spaces	Définissent l'espace d'action (4 directions) et l'espace d'observation (vecteur d'état).

Dexcription du Projet

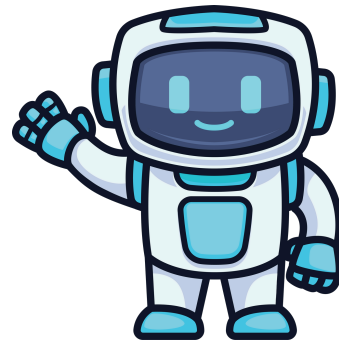


Commandes à compiler



Jouer en tant qu'humain :

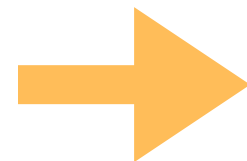
PPO : `python play_as_human_ppogame.py`



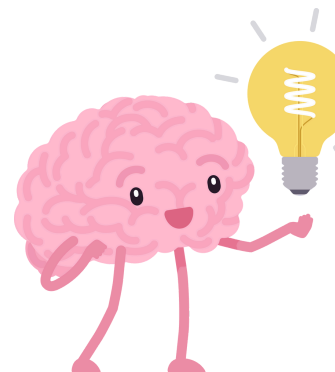
Q-Learning : `python play_as_human_QLgame.py`

Observer un agent IA :

PPO : `python test_ppoAgent.py`



Q-Learning : `python test_Agent_Qlearning.py`



Lancer le menu interactif :

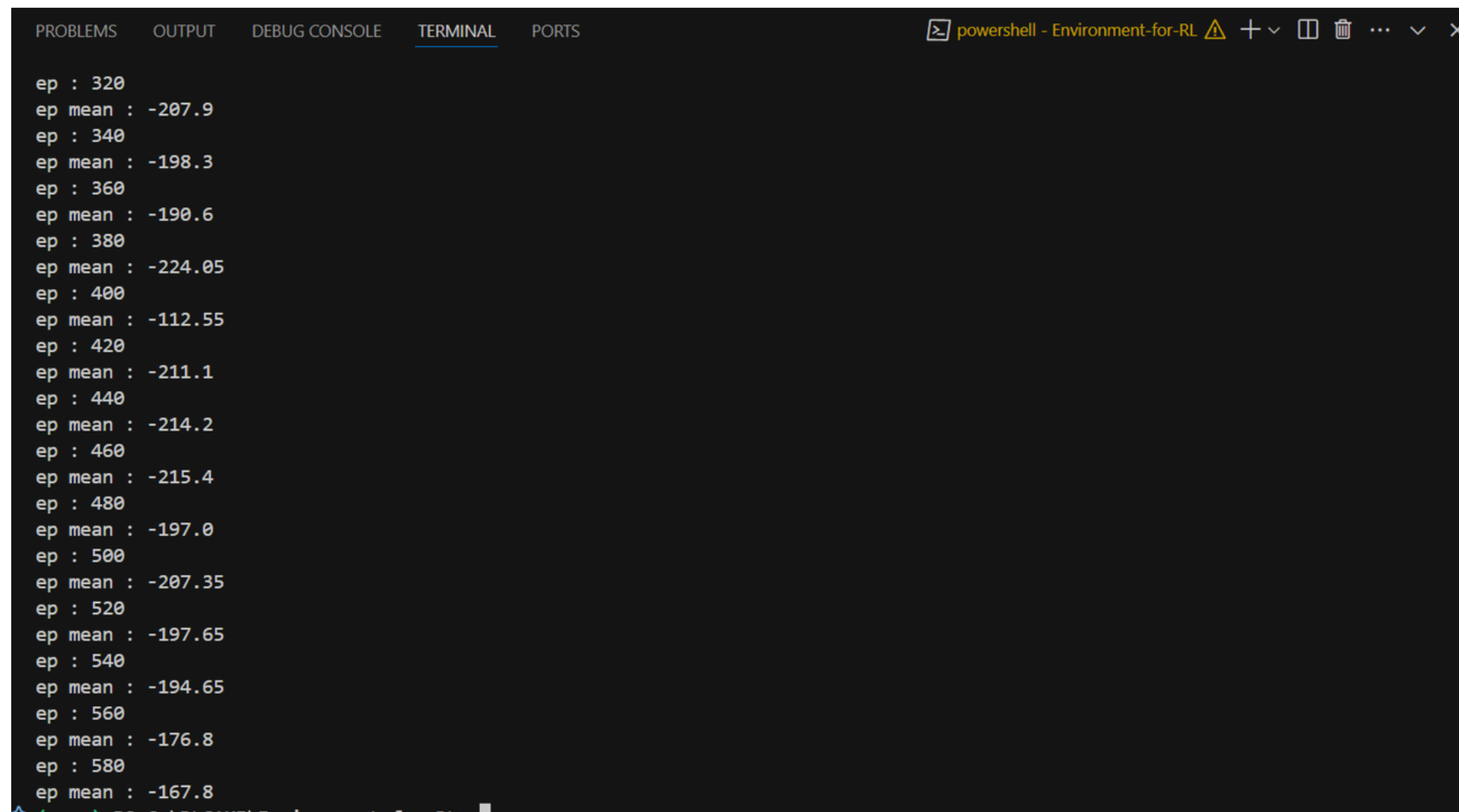
`python play_or_watch.py`

(Permet de choisir entre jouer ou observer l'agent PPO)

Dexcription du Projet

Agent Q-Learning en cours d'apprentissage

L'agent Q-Learning est en train d'apprendre et améliore progressivement son score au fil des épisodes.



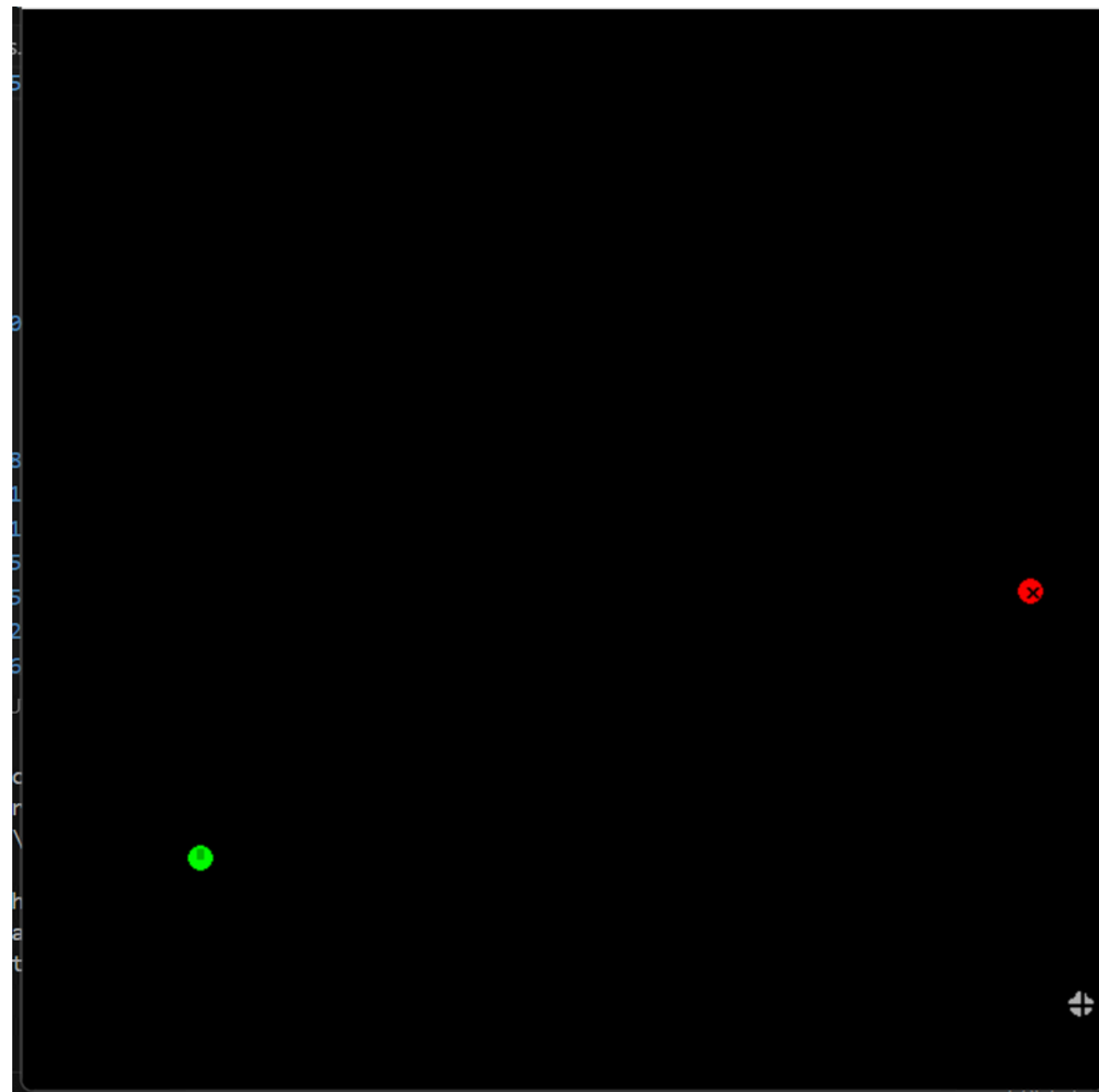
The screenshot shows a PowerShell terminal window titled "powershell - Environment-for-RL". The terminal displays the following output:

```
ep : 320
ep mean : -207.9
ep : 340
ep mean : -198.3
ep : 360
ep mean : -190.6
ep : 380
ep mean : -224.05
ep : 400
ep mean : -112.55
ep : 420
ep mean : -211.1
ep : 440
ep mean : -214.2
ep : 460
ep mean : -215.4
ep : 480
ep mean : -197.0
ep : 500
ep mean : -207.35
ep : 520
ep mean : -197.65
ep : 540
ep mean : -194.65
ep : 560
ep mean : -176.8
ep : 580
ep mean : -167.8
```

Dexcription du Projet

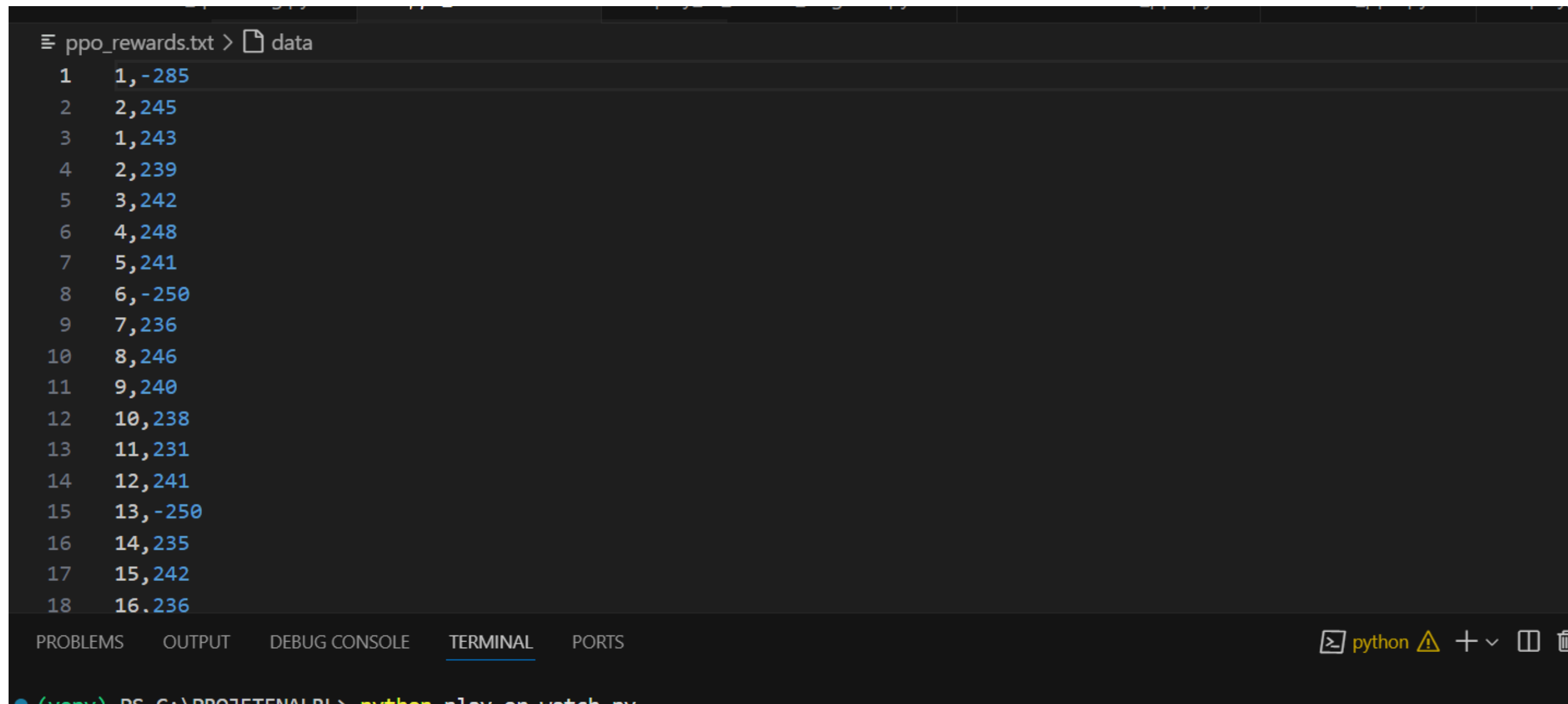
Visualisation Environnement de l'agent Q-Learning en train de jouer

L'agent Q-Learning évolue dans une grille où il prend des décisions pour maximiser son score en temps réel.



Dexcription du Projet

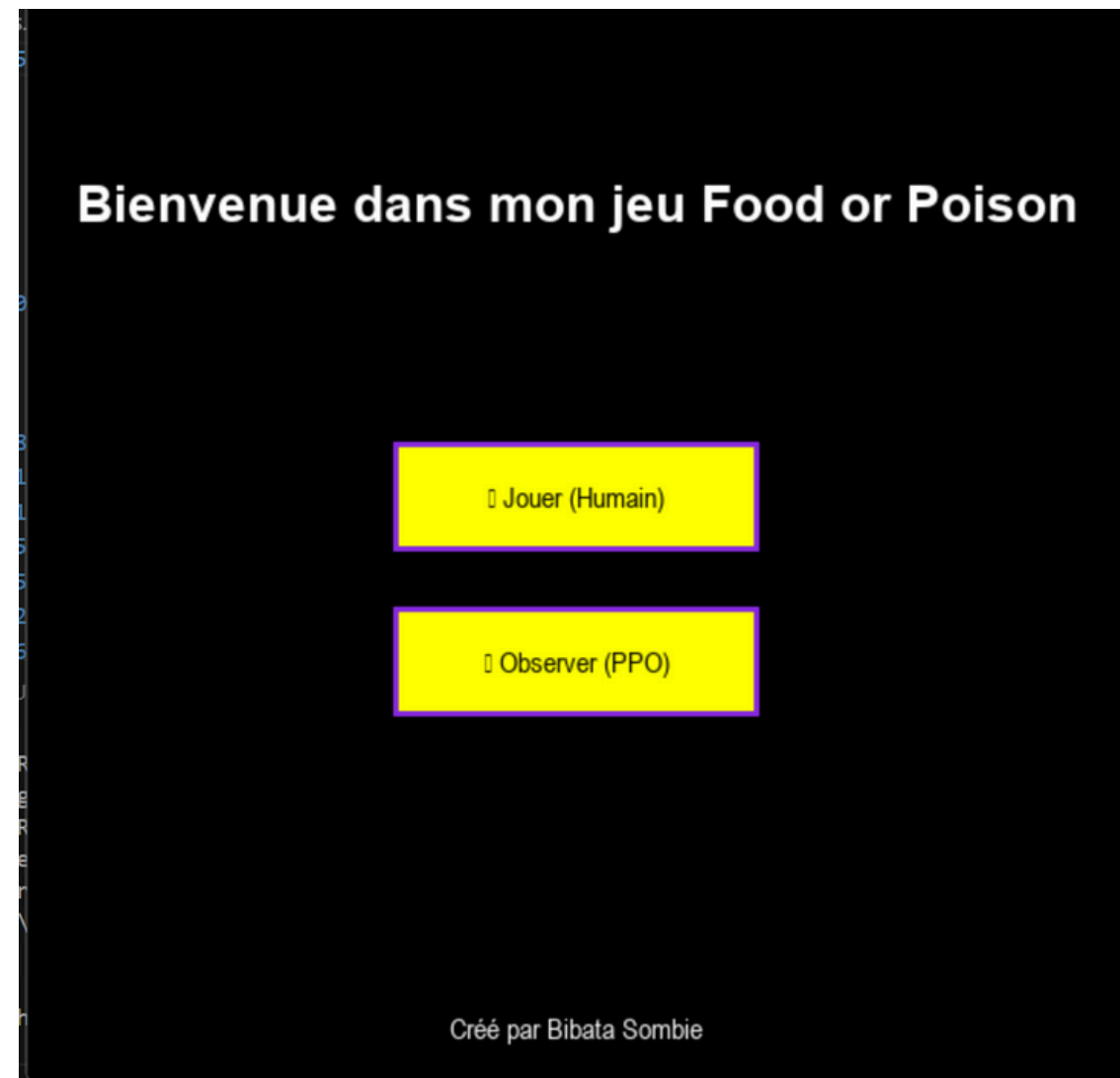
Agent PPO en cours d'apprentissage



```
ppo_rewards.txt > data
1 1,-285
2 2,245
3 1,243
4 2,239
5 3,242
6 4,248
7 5,241
8 6,-250
9 7,236
10 8,246
11 9,240
12 10,238
13 11,231
14 12,241
15 13,-250
16 14,235
17 15,242
18 16,236
```

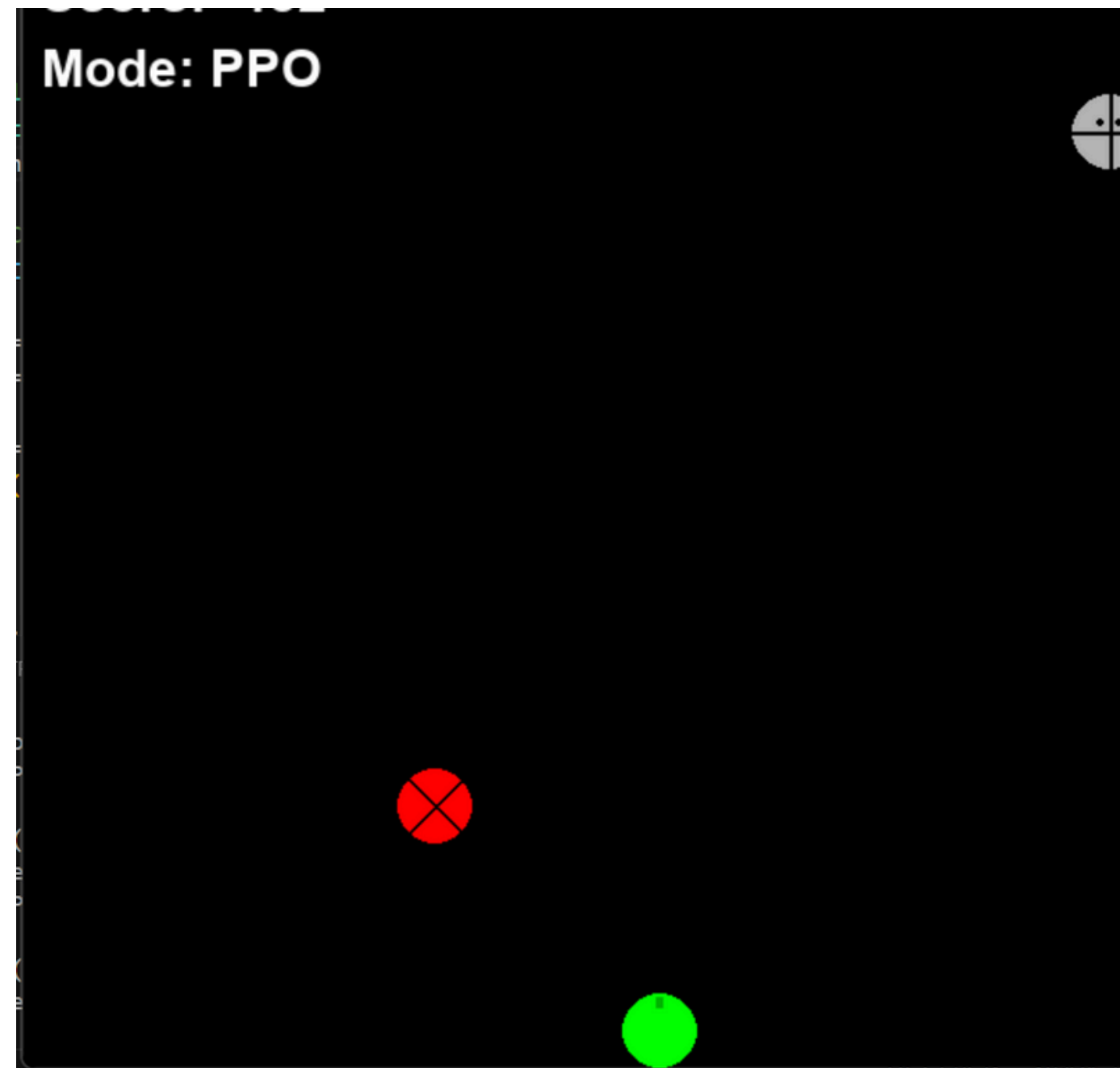
Dexcription du Projet

- Interface d'accueil pour jouer ou observer un agent joue



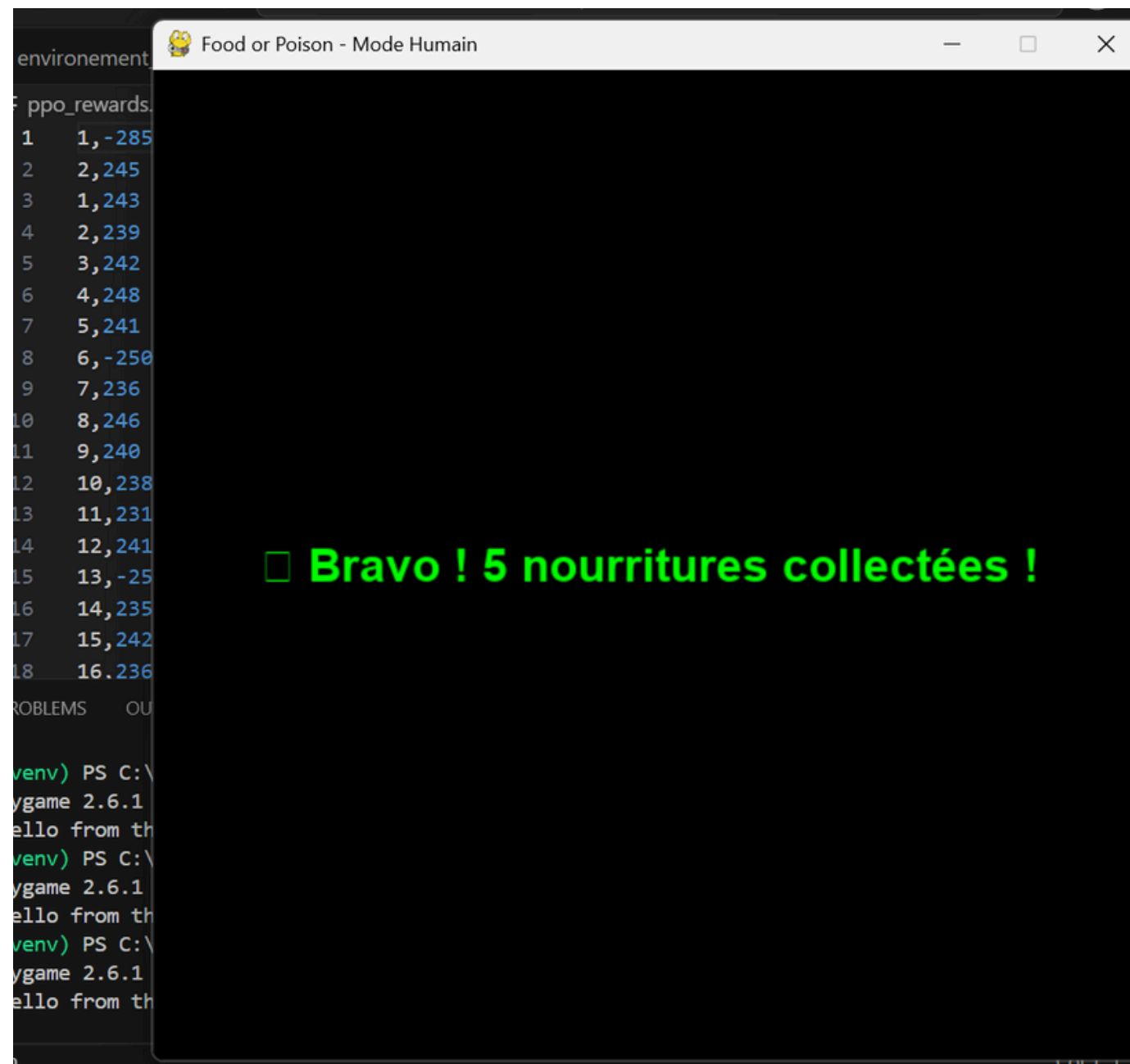
Dexcription du Projet

Visualisation de l' Environnement de l'agent PPO en train de jouer



Dexcription du Projet

Fin de partie(play_as_human)

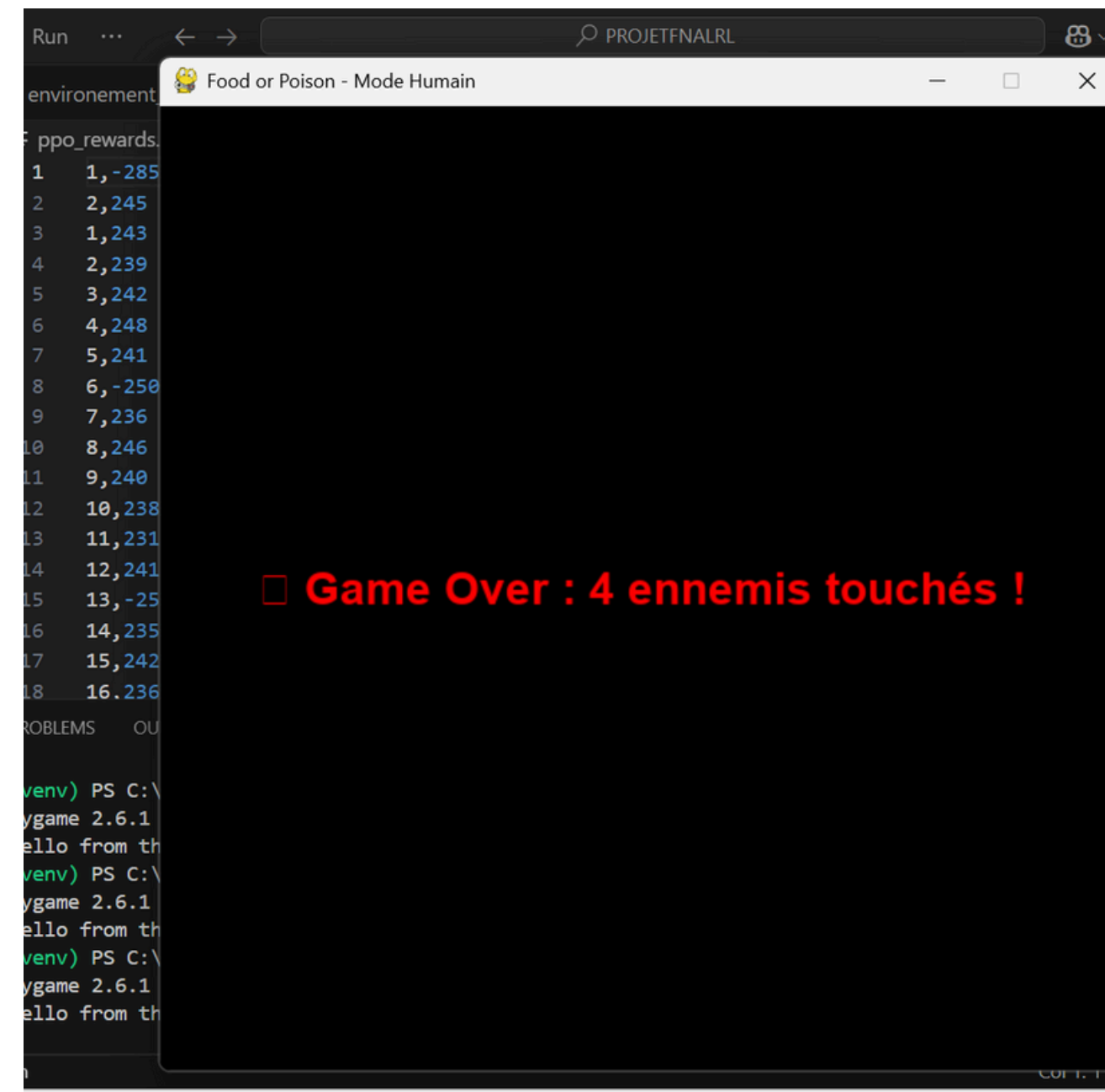


```
environnement
ppo_rewards
1 1,-285
2 2,245
3 1,243
4 2,239
5 3,242
6 4,248
7 5,241
8 6,-250
9 7,236
10 8,246
11 9,240
12 10,238
13 11,231
14 12,241
15 13,-25
16 14,235
17 15,242
18 16,236
```

ROBLEMS OU

venv) PS C:\
ygame 2.6.1
ello from th
venv) PS C:\
ygame 2.6.1
ello from th
venv) PS C:\
ygame 2.6.1
ello from th

Bravo ! 5 nourritures collectées !



```
Run ... < -> PROJETFINALRL
environnement
ppo_rewards
1 1,-285
2 2,245
3 1,243
4 2,239
5 3,242
6 4,248
7 5,241
8 6,-250
9 7,236
10 8,246
11 9,240
12 10,238
13 11,231
14 12,241
15 13,-25
16 14,235
17 15,242
18 16,236
```

ROBLEMS OU

venv) PS C:\
ygame 2.6.1
ello from th
venv) PS C:\
ygame 2.6.1
ello from th
venv) PS C:\
ygame 2.6.1
ello from th

Game Over : 4 ennemis touchés !

Difficultés rencontrées

- Utilisation d'un seul environnement, ce qui limitait la comparaison
- Présence d'un seul blob au départ (pas de diversité dans le jeu)
- Temps d'apprentissage long pour l'agent
- Difficulté à équilibrer les récompenses
- Compréhension des bibliothèques comme Gym et Stable-Baselines3
- Visualisation de l'agent pas claire au début
- Organisation du dépôt GitHub peu lisible

solutions proposées

- Création de deux environnements distincts : PPO (Gym) et Q-learning (manuel)
- Ajout de plusieurs blobs (nourriture et ennemi) avec comportements différents
- Laisser l'agent apprendre sur une longue durée avec plus d'épisodes
- Ajustement progressif des récompenses (+1, -1, etc.)
- Étude des documentations officielles et projets GitHub similaires
- Implémentation d'une interface d'observation claire avec Pygame
- Réorganisation du projet avec un README structuré

