Sujet de TP 1 (Morpion) Pygame + Q-learning

Contexte et objectif
Vous allez créer deux jeux jouables en Pygame (Morpion puis Puissance 4), puis entraîner un agent par apprentissage par renforcement (Q-learning) pour qu'il apprenne à jouer. Le but n'est pas "juste" de faire une IA qui joue, mais de construire une architecture propre (environnement/agent), de définir correctement états/actions/récompenses, et de mettre en place une boucle d'entraînement et d'évaluation.

Livrables attendus (pour chaque jeu)

1. Un jeu Pygame jouable humain vs humain.
2. Un mode humain vs IA (IA = Q-learning).
3. Un mode IA vs IA (évaluation automatisée, sans rendu graphique ou avec rendu accéléré).
4. Un rapport court (1–2 pages) : représentation d'état, récompenses, hyperparamètres, résultats (courbes/taux).

Règles générales d'architecture (imposées)
Vous devez séparer clairement :

- L'environnement (le jeu) : règles, coups légaux, application d'un coup, détection fin de partie.
- L'agent : choix d'action (exploration/exploitation) + mise à jour Q-learning.
- L'entraînement : boucle d'épisodes, suivi des scores, sauvegarde/chargement des Q-valeurs.
- L'interface Pygame : affichage + gestion événements clavier/souris + choix des modes.

Vous devez pouvoir exécuter le jeu dans 3 modes :

- MODE 1 : humain vs humain
- MODE 2 : humain vs agent
- MODE 3 : agent vs adversaire (random ou agent) sans interaction

Partie A — Morpion (Tic-Tac-Toe)

A1. Spécifications du jeu

- Plateau 3x3.
- Deux joueurs : X et O.
- Un coup = placer son symbole dans une case vide.
- Fin : victoire (3 alignés) ou match nul (plateau plein).

A2. Représentation de l'état
Objectif : une représentation compacte, "hashable", qui identifie un plateau.

Exemples de représentations possibles (choisissez-en une) :

- Tuple de 9 cases (valeurs {0, 1, 2} ou {-1,0,+1})

Important : l'état doit inclure le plateau ET le joueur courant (sinon ambiguïté).

A3. Actions légales

- Les actions sont les indices des cases vides : {0..8} filtrés.
- Votre environnement doit fournir legal_actions(state).

Pseudo-code (actions légales)

- actions = []
- pour i dans 0..8 :
    - si board[i] est vide : actions.append(i)
- retourner actions

A4. Récompenses (à respecter)
Version simple (recommandée) :

- +1 si l’agent gagne
- -1 si l’agent perd
- 0 sinon (dont match nul)
  Option autorisée : petite pénalité par coup (-0.01) pour encourager des parties courtes, mais ce n’est pas obligatoire.

Attention : la récompense doit être du point de vue de l’agent. Si vous entraînez X, alors “victoire de X” = +1.

A5. Q-learning tabulaire
Vous stockez Q(s,a) dans une structure de type dictionnaire.

- Clé possible : (state_key, action)
- Ou : Q[state_key] = tableau de taille |A|, avec des valeurs pour actions non légales ignorées.

Règle de mise à jour
À chaque transition (s, a, r, s’, done) :

Pseudo-code

- q_old = Q[s,a] (0 si absent)
- si done :
  - target = r
  - sinon :
    - target = r + gamma * max(Q[s’,a’] pour a’ dans legal_actions(s’))
- Q[s,a] = q_old + alpha * (target - q_old)

A6. Politique ε-greedy
Pseudo-code (choix action)

- si random() < epsilon :
  - retourner une action aléatoire parmi legal_actions(s)

- sinon :
    - retourner argmax_a Q(s,a) sur legal_actions(s) (en cas d'égalité, départager aléatoirement)

A7. Boucle d'entraînement (épisodes)
Vous jouez beaucoup de parties (épisodes). Un épisode = une partie complète.

Pseudo-code (entraînement morpion)

- initialiser Q vide
- pour episode de 1..N :
    - s = reset()
    - done = faux
    - tant que non done :
        - a = agent.act(s, legal_actions(s), epsilon)
        - (s', r, done) = env.step(a)
        - agent.learn(s, a, r, s', done)
        - s = s'
    - mettre à jour epsilon (décroissance)

A8. Adversaires pour entraîner et évaluer
Vous devez implémenter au moins :

- Adversaire aléatoire (joue un coup légal au hasard)
  Option : self-play (agent vs agent), mais attention à l'instabilité au début.

Évaluation minimale (à fournir)

- Taux de victoire vs aléatoire sur 200 parties avec epsilon=0 (agent en greedy)
- Taux de défaite (doit tendre vers 0)
- Taux de nul (souvent élevé si l'agent devient bon)

A9. Intégration Pygame (sans ralentir l’entraînement)
Exigences :

- Interface graphique pour jouer et observer.
- Mode “entraînement rapide” sans rendu OU rendu accéléré (ex: 1000 épisodes sans afficher).
- Boutons/clavier pour : changer de mode, reset, activer/désactiver affichage, vitesse.