Documentation Technique pour le Jeu de Morpion

Introduction

Cette documentation détaille l'architecture, les fonctionnalités et la structure du code source du jeu de Morpion développé avec PyQt5. Ce jeu implémente plusieurs modes de jeu, notamment Joueur contre Joueur et Joueur contre IA, avec des niveaux de difficulté croissants pour l'IA.

Architecture du Jeu

1. Structure Générale

Le jeu est structuré en un seul fichier Python contenant une classe principale `Morpion` qui encapsule toute la logique de jeu. Les principales sections sont :

- **Interface Utilisateur (UI)** : Créée avec **PyQt5**, elle permet l'interaction entre l'utilisateur et la logique du jeu.
- **Logique du Jeu** : Gestion des règles, vérification de la victoire/défaite/égalité, et implémentation des différents niveaux de l'IA.
- **Point d'entrée** : La section `if __name__ == "__main__"` initialise l'application.

2. Structure de la Classe `Morpion`

Attributs

- `self.grid` : Une matrice 3x3 représentant les boutons de la grille de jeu.
- `self.current_player` : Indique le joueur en cours ("X" ou "O").
- `self.mode_combo` : Sélection du mode de jeu (Joueur contre Joueur ou Joueur contre IA).
- `self.level_combo` : Sélection du niveau de difficulté de l'IA.

Méthodes Principales

1. **Initialisation et UI** :
    - `__init__()` : Initialise la fenêtre et les attributs.
    - `initUI()` : Configure les composants de l'interface (grille, menus, etc.).
2. **Logique de Jeu** :
    - `make_move(x, y)` : Gère le mouvement d'un joueur humain.
    - `ai_move()` : Appelle les méthodes de l'IA en fonction du niveau choisi.
    - `check_winner()` : Vérifie si un joueur a gagné.
    - `is_draw()` : Vérifie si la grille est pleine sans gagnant.

- reset_game() : Réinitialise le jeu.

3. **IA** :
   - simple_ai() : Joue un coup aléatoire.
   - intermediate_ai() : Joue stratégiquement pour gagner ou bloquer l'adversaire.
   - expert_ai() : Utilise l'algorithme **Minimax** pour déterminer le meilleur coup.

Logique du Jeu

1. Vérification de la Victoire

La méthode check_winner_board() vérifie les conditions suivantes sur la grille :

- Trois cases identiques alignées dans une **ligne**.
- Trois cases identiques alignées dans une **colonne**.
- Trois cases identiques alignées dans une **diagonale**.

Exemple de vérification des lignes :

```
for row in board:
    if row[0] == row[1] == row[2] and row[0] != "":
        return True
```

2. Détection de l'Égalité

La méthode is_draw_board() retourne True si toutes les cases sont remplies, et aucune condition de victoire n'est satisfaite.

Code :

```
def is_draw_board(self, board):
    return all(cell != "" for row in board for cell in row)
```

3. Algorithme Minimax

L'IA experte utilise l'algorithme **Minimax** pour évaluer tous les coups possibles et choisir celui qui maximise ses chances de gagner ou minimise ses risques de perdre. Cela inclut :

- La simulation des coups possibles.
- L'évaluation des scores pour chaque scénario (victoire, défaite, égalité).
- Le retour du meilleur coup pour le joueur actuel.

Code :

```
def best_move(self, board, sign):
    opponent = "X" if sign == "O" else "O"
    best_score = - float("inf") if sign == self.current_player else float("inf")
    best_move = None
```

```
    for i in range(3):
        for j in range(3):
            if board[i][j] == "":
                board[i][j] = sign
                if self.check_winner_board(board):
                    score = 1 if sign == self.current_player else - 1
                elif self.is_draw_board(board):
                    score = 0
                else:
                    score, _ = self.best_move(board, opponent)
                board[i][j] = ""
                if (sign == self.current_player and score > best_score) or (sign !=
self.current_player and score < best_score):
                    best_score = score
                    best_move = (i, j)

    return best_score, best_move
```

Interface Utilisateur (UI)

1. Menus

- **Mode de Jeu** : Sélection entre :
  - Joueur contre Joueur
  - Joueur contre IA
- **Niveau de l'IA** : Sélection entre :
  - Simple
  - Intermédiaire
  - Expert

2. Grille de Jeu

- La grille est une matrice 3x3 de boutons PyQt5.
- Chaque bouton représente une case du jeu.

3. Notifications

- Utilisation de QMessageBox pour afficher :
  - Les messages de victoire/défaite.
  - Les messages d'égalité.

Scénarios de Test utilisateur E2E

Étape	Scénario	Critère de Réussite	Succès ?
Démarrage du jeu	Afficher une grille vide	La grille est initialisée avec des cases vides.	O
Tour du joueur	Cliquer sur une case vide	La case est marquée avec le symbole du joueur.	O
Tour de l'IA	L'IA joue après le joueur	Une case vide est marquée par l'IA.	O
Détection de la victoire	Simuler une victoire	Le message de victoire s'affiche.	O
Détection de l'égalité	Simuler une grille pleine sans gagnant	Le message d'égalité s'affiche.	O
Niveaux de l'IA	Tester les différents niveaux de l'IA	Les comportements de chaque niveau sont respectés.	O

Conclusion

Ce jeu de Morpion est conçu pour être extensible et maintenable grâce à une architecture modulaire. L'utilisation de PyQt5 offre une interface intuitive, tandis que l'implémentation de plusieurs niveaux d'IA permet de varier les défis pour le joueur. Nous avons par ailleurs implémenté des tests unitaires qui vérifient le bon fonctionnement du code avec des assertions.