Documentation Technique – Jeu Morpion Python

I. Introduction

A. Présentation du jeu

Le projet Morpion est un jeu de réflexion classique développé en Python à l'aide du module Tkinter pour l'interface graphique. Deux joueurs s'affrontent en plaçant tour à tour leur symbole (« X » ou « O ») sur une grille de 3×3. Le premier joueur à aligner trois symboles identiques horizontalement, verticalement ou en diagonale remporte la partie.

B. Objectifs de la documentation technique

Cette documentation présente l'architecture du jeu, son fonctionnement interne, et la structure de son code. Elle s'adresse aux développeurs souhaitant comprendre le projet.

II. Architecture du jeu

A. Moteur de jeu

Le moteur du jeu repose sur la logique suivante :
- Une grille représentée par une liste Python de 9 cases (indices 0 à 8).
- Une fonction verifier_gagnant(grille) qui détermine si un joueur a gagné.
- La gestion des tours alternés entre les joueurs X et O.
- La détection d'une victoire ou d'un match nul, suivie d'une réinitialisation.

B. Système de rendu

L'interface graphique est construite avec Tkinter, en utilisant une grille (layout grid()) composée de :
- 9 boutons (3×3) représentant les cases du plateau.
- Un bouton "Rejouer" pour relancer une partie.

C. Gestion des ressources

Le jeu n'utilise aucune ressource externe (images, sons, fichiers). Tous les éléments visuels sont créés dynamiquement via Tkinter.

III. Conception du gameplay

A. Mécaniques de jeu

Chaque clic valide sur une case vide inscrit le symbole du joueur actif. La fonction verifier_gagnant() teste toutes les combinaisons gagnantes possibles. Si un gagnant est trouvé, une boîte de dialogue affiche le résultat. Si toutes les cases sont remplies sans gagnant, un message de "Match nul" apparaît.

B. Systèmes de contrôle
Le jeu repose sur les événements de clic utilisateur sur les boutons Tkinter. Les boutons sont désactivés exprès, lorsqu'ils contiennent déjà un symbole.

C. Interactions joueur-environnement
L'environnement graphique agit comme une interface simple et intuitive : chaque bouton représente une action possible et la fenêtre principale gère la boucle de jeu via root.mainloop().

IV. Graphismes et Animation

A. Modélisation
Aucune modélisation graphique complexe n'est utilisée. Les boutons Tkinter servent de composants visuels de base.

B. Textures et styles
Les polices et tailles sont définies directement dans le code : font=("Arial", 30) pour les boutons de jeu.

C. Animation
Le jeu ne contient pas d'animations. L'affichage repose sur la mise à jour immédiate des boutons.

V. Son et Musique
Aucun son ou musique n'est intégré.

VI. Optimisation et Performance

A. Gestion des ressources système
Le programme est léger et n'utilise que la mémoire nécessaire à Tkinter et à la liste de la grille.

B. Optimisation du code
La logique est simple, et la fonction verifier_gagnant est optimisée avec une complexité O(1).

C. Tests de performance
Aucun test de performance n'est requis : le jeu répond instantanément.

VII. Documentation du Code

A. Structure du code source
1. Fonction de vérification de victoire : verifier_gagnant(grille)
2. Classe principale MorpionApp (init, clic_case, reinitialiser)
3. Programme principal d'exécution (root.mainloop())

B. Commentaires et documentation interne

Le code contient des commentaires détaillant le rôle et les paramètres de chaque fonction.

C. Bonnes pratiques de codage

Respect des conventions PEP8, avec une bonne indentation, des noms explicites et séparation entre logique et interface.

X. Déploiement et Maintenance

A. Configuration requise

Python 3.x et Tkinter.

B. Procédure d'installation

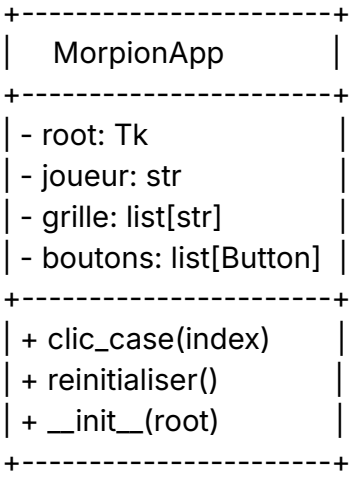
1. Télécharger morpion.py
2. Exécuter : python morpion.py
3. Le jeu s'ouvre dans une fenêtre Tkinter.

C. Gestion des mises à jour

Potentiel système de score ou modernisation de l'interface avec animation ou sons (CustomTkinter).

XI. Annexes

A. Diagramme UML simplifié



B. Exemple de grille logique

Indices :
0 | 1 | 2
----
3 | 4 | 5
----
6 | 7 | 8

C. Capture d’écran