Fiche descriptive de projet Hub

Chessgame

Contexte et but du projet

J'aimerais faire un jeu d'échecs ou l'on pourra jouer dessus car cela travaillerait une partie algorithme est stratégie une fois le jeu terminer

Résumé du projet

- **Étude du jeu d'échecs**
- Rappel des règles officielles (déplacement des pièces, échecs, échec et mat, roque, promotion, etc.)
- Identification des cas particuliers à gérer
- **Modélisation des pièces et du plateau**
- Définition des classes : Piece, Pawn, Rook, Knight, Bishop, Queen, King
- Création de la classe Board pour représenter l'échiquier
- **Gestion des mouvements légaux**
- Implémentation des déplacements spécifiques à chaque type de pièce
- Vérification des collisions et des mouvements interdits
- Gestion des règles complexes : échec, roque, prise en passant, promotion
- **Système de jeu**
- Alternance des tours (blancs / noirs)
- Détection des échecs et échecs et mat
- Affichage du statut de la partie (fin de partie)
- Une
- **Interface graphique (avec pygame ou tkinter)**
- Dessin du plateau 8x8
- Affichage des pièces selon leur position
- Interaction souris : sélection et déplacement des pièces
- **Tests et vérifications**
- Vérification du comportement des pièces et des règles
- Tests de parties complètes, détection de fin de partie
- **Fonctionnalités optionnelles**
- **Nettoyage et documentation du code**
- Organisation du code en modules
- Ajout de commentaires/docstrings
- Fichier README.md et manuel d'utilisation

Porteur(s) du projet

Moi-même Antoine POLICAND chef de projet

Environnement technique / technologique

Matériel : un pc.

Langages : python

Environnement d'exécution : un terminal

Ressources : vidéos, site web.

Description du livrable

1

. **Fichiers Python principaux (programmes)**

- main.py
  - **Rôle** : point d'entrée du programme. Initialise la fenêtre de jeu et lance la boucle principale.
  - **Finition** : totalement fonctionnel, commenté, documenté.
- board.py
  - **Rôle** : contient la classe Board, qui gère l'état de l'échiquier, les positions des pièces, et les règles.
  - **Finition** : stable et testé, documenté, prêt pour déploiement.
- pieces.py
  - **Rôle** : définit les classes des différentes pièces (Pawn, Knight, Bishop, etc.), avec leurs règles de mouvement.
  - **Finition** : complet, testé, réutilisable.
- game.py
  - **Rôle** : logique de jeu (gestion des tours, détection d'échecs, fin de partie, ia).
  - **Finition** : terminé, documenté, en production.

2. Modules / Librairies externes

- pygame

- **Utilisation** : interface graphique, affichage du plateau, gestion des événements souris/clavier.
- **Finition** : entièrement intégré et utilisé dans le projet.

3. Assets (visuels)

- Dossier /assets/
  - **Contenu** :
    - Images PNG des pièces (roi, dame, cavalier, etc.) pour blanc et noir.
    - Image ou couleur pour le plateau.
  - **Finition** : tous les fichiers requis sont présents et chargés correctement.
  - **Licence** : libres de droit ou créés pour le projet.

4. Fichiers de configuration / ressources

- config.json ou équivalent
  - **Utilisation** : configuration des options du jeu (taille fenêtre, couleurs, etc.)
  - **Finition** : présent si le projet est étendu ; sinon non nécessaire.

5. Documentation

- README.md
  - **Contenu** : présentation du projet, installation, utilisation, dépendances, auteurs.
  - **Finition** : complet, clair, avec exemples d'exécution.
- INSTALL.md
  - **Contenu** : instructions détaillées d'installation (pip, OS supportés).
  - **Finition** : minimal ou intégré au README.
- docstrings dans les fichiers Python
  - **Finition** : chaque classe et fonction est documentée selon le style Python (PEP257).

6. Déploiement

- Lancement :
  - python3 main.py
  - **Finition** : exécutable directement après installation des dépendances.
- Environnement virtuel :
  - Fichier requirements.txt généré avec pip freeze.

- Permet d’installer les dépendances avec pip install -r requirements.txt.
- Finition : prêt pour GitHub ou une présentation.

Organisation et temporalité

1 – Tâche 1

Explications de la tâche

- **Implémentation d'un jeu d'échecs**
- **Première interface**
- **Recherche d'assets des pieces**
- **Dernier bug fix**
- **Test est vérification**
- **Gestion de mouvement**

Temps estimé pour cette tâche :

- **25 heures**
- **4 heures**
- **6 heures**

- **2 heures**
- **30 heures**
- **6 heures**

Estimation du projet

Environ 4 a 6 heures de travailler par jour.

70h de travaille