LES TROPHÉES NSI

Édition 2022

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET

Ce document est l'un des livrables à fournir lors du dépôt de votre projet : 4 pages maximum (hors documentation).

Pour accéder à la liste complète des éléments à fournir, consultez la page **Préparer votre participation**.

Vous avez des questions sur le concours ? Vous souhaitez des informations comp lémentaires pour déposer un projet ? Contactez-nous à **info@trophees-nsi.fr**.

NOM DU PROJET :

PyChess

> PRÉSENTATION GÉNÉRALE :

- *Idée et objectifs*
- *Origines et intérêts du projet*
- *(...)*

-L'idée du projet est simple, reprendre le jeu d'échec classique avec toutes ses règles et le coder, tout simplement.

-Le jeu est jouable en 1 contre 1 avec une interface qui s'affiche dans la console de l'IDE.

-Le but est de pouvoir joueur aux échecs uniquement avec un ordinateur

-Le projet d'origine est seulement un projet de cours, à la base, censé être un jeu de poker, il sera finalement dérivé en jeu d'échec.

-Il n'y a pas réellement d'intérêt mis à part d'être un jeu d'échec

> ORGANISATION DU TRAVAIL :

- *Présentation de l'équipe (prénom de chaque membre et rôle dans le projet)*
- *Répartition des tâches*
- *Organisation du travail (répartition par petits groupes, fréquence de réunions, travail en dehors de l'établissement scolaire, outils/logiciels utilisés pour la communication et le partage du code, etc.)*

- Nicolas GAUVAN : Principal codeur, a codé en grande partie le jeu d'échec.
- Manatoa LOUIS : Artiste, Graphiste : réalisateur de la vidéo présentation, codeur secondaire, a aidé au code.

(Noah COUILLAUD : Manager)

LES ÉTAPES DU PROJET :

• *Présenter les différentes étapes du projet (de l'idée jusqu'à la finalisation du projet)*

1ère étape : Réflexion sur les règles du jeu d'échec, la structure du code, les différentes bibliothèques à créer et les interactions entre elles. Création d'un carnet des charges pour savoir que faire, dans quel ordre et comment s'y prendre.

2ème étape : Créer les bibliothèques nécessaires

3ème étape : Coder le jeu sur le main.py et résoudre les bugs qui apparaissent

4ème étape : Pleurer de ses nuits blanches

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

- *Avancement du projet (ce qui est terminé, en cours de réalisation, reste à faire)*
- *Approches mises en œuvre pour vérifier l'absence de bugs et s'assurer de la facilité d'utilisation du projet*
- *Difficultés rencontrées et solutions apportées*

Avancement du projet :

- Ce qui est terminé :
- La création de l'échiquier
- L'affichage de l'échiquier dans la console
- La création de toutes les bibliothèques nécessaires (chessboard.py , rules.py , piece.py)
- Le déplacement des pièces
- Les coups mangeant de pièces
- Le déroulement de la partie
- Le roque
- La promotion du pion
- L'empêchement pour les pièces de passer à travers les autres pièces (sauf cavalier)
  - L'état Echec
- L'état échec et mat

- En cours de réalisation :

- L'état PAT (match nul)

Ce qui reste à faire :

- Le « En passant »
- Le mur entre les rois
- Tester si il reste encore des bugs
- Système de pendules pour chaque joueurs

> OUVERTURE :

- *Idées d'améliorations (nouvelles fonctionnalités)*
- *Stratégie de diffusion pour toucher un large public (faites preuve d'originalité !)*
- *Analyse critique du résultat (si c'était à refaire, que changeriez-vous dans votre organisation, les fonctionnalités du projet et les choix techniques ?)*

Idées d'amélioration :

-Créer une interface graphique :

-Jouer les coups en cliquant sur la pièce voulue et cliquer sur la case voulu pour se déplacer

-Afficher les coups possibles de la pièce sur l'échiquier

-Ajouter des skins personnalisable pour chaque pièces

-Ajouter différents style d'échiquier

-Possibilités de partager sur les réseaux sociaux, sur des sites web, en parler aux entourages.

-Analyse critique du résultat :

- Le code n'est pas du tout optimisé, en effet, il y a énormément de lignes de code qui peuvent être réduites par des fonctions

- Nous avons ajouté certaines fonctions pour parvenir à résoudre des bugs. Par conséquent, il y a des fonctions qui ne devraient pas être présentes si nous avions mieux pensé le projet.

- Nous avons passé beaucoup de temps sur la création des modules, ainsi, si c'était à refaire, nous pourrions utiliser les modules Chess déjà existant afin de pouvoir se concentrer davantage sur la personnalisation du jeu.

DOCUMENTATION

- *Spécifications fonctionnelles (guide d'utilisation, déroulé des étapes d'exécution, description des fonctionnalités et des paramètres)*
- *Spécifications techniques (architecture, langages et bibliothèques utilisés, matériel, choix techniques, format de stockage des données, etc)*
- *Illustrations, captures d'écran, etc*

Techniques :

-Langage : Python

-IDE : Pycharm

-Bibliothèques utilisées : « chessboard.py », « piece.py » , « rules.py » toutes crées de nos mains

- Matériel : Ordinateur

```python
class Piece:

    def __init__(self, color, axis, ordo, board, number=1, point=1):
        self.color = color
        self.axis = axis
        self.ordo = ordo
        self.number = number
        self.board = board
        self.point = point

    def get_ordo(self):
        return self.ordo

    def get_axis(self):
        return self.axis

    def get_color(self):
        return self.color

    def rollback_move(self, move):
        abs_dict = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7, 'h': 8}
        self.ordo = int(move[3])
        self.axis = abs_dict[move[2]]
        return True

    # Permet de savoir si une pièce est mangeable par une autre pièce
    def can_be_eaten(self, piecelist):
        for p in piecelist:
            if (self.ordo, self.axis) in p.all_eat_moves():
                return True
        return False

    def machine_move(self, move):
        self.ordo = move[0]
        self.axis = move[1]
        return True
```

```python
# Fonction montrant l'échiquier du côté blanc
def show_chessboard_white_side(self):
    b = [['  '], ['_' for i in range(21)]]
    c = [['  '], ['‾' for i in range(21)]]
    d = ['    ']
    print(*b[0], *b[1])
    for i in range(8, 0, -1):
        a = [self.chessboard[i][0], '|']
        for j in range(1, 9):
            a.append(self.chessboard[i][j])
            a.append('|')
        print(*a)
        a = ['|']
    print(*c[0], *c[1])
    for i in range(1, 9):
        d.append(self.chessboard[0][i])
        d.append(' ')
    print(*d)
    return True
```