Projet Rolit

Rapport

Université Gustave Eiffel
Groupe de projet 12-6
KUNTZ Nathan, LI Jérémy, VISTE Lilian

Sommaire

Guide utilisateur

- **Installation**
    1. **Installer `Python 3.10` ou plus récent**
    2. **Télécharger le code source**
    3. **Décompresser le fichier téléchargé**
- **Utilisation**
    1. **Ouvrir un terminal dans le dossier où se trouve le code source**
    2. **Exécuter la commande `python main.py`**
- **Arguments Optionnels**
    - **`--graphical`/`--no-graphical` : Utiliser le mode graphique ou non (activé par défaut) (ATTENTION: le mode console n'est plus supporté, certaines fonctionnalités ne sont donc disponibles que dans le mode graphique)**
    - **`--nb_players <nombre de joueurs>` ou `-n <nombre de joueurs>` : Nombre de joueurs (1 à 4)**
    - **`--nb_manches <nombre de manches>` ou `-m <nombre de manches>` : Nombre de manches**
    - **`--ai` : Activer les IA, en mode graphique, le nombre d'IA sera choisit par l'utilisateur, en mode console, le nombre d'IA sera égal au nombre de joueurs - 1 (i.e. si l'argument `--nb_players` est égal à 4, il y aura 3 IA et 1 joueur humain)**

Bugs connus

- **Avec une ancienne version de `tkinter`, il est possible que le jeu plante. (`Seg fault` en Python ???)**
- **Le jeu peut ne pas fonctionner correctement sur Linux, notamment pour l'affichage des emojis (le bouton paramètre, le bouton retour dans les paramètres, les couronnes, etc.)**

Structures de données

- **La grille de jeu**

La grille de jeu est représentée par une matrice de taille 8x8, soit une liste de liste. Cela permet de visualiser plus facilement le plateau de jeu.

- **Les joueurs**

Les joueurs sont représentés par un nombre de 1 à 4. Ils sont définis comme constante dont le nom de la constante est une couleur (i.e. `RED = 1`).

- **Les scores**

Les scores sont enregistrés à l'aide de dictionnaires, associant la couleur des joueurs à leurs scores.

- **Les thèmes/couleurs**

Les thèmes/couleurs sont également définis par des dictionnaires, associant à un joueur, une chaîne de caractère représentant la couleur (couleur en hexadécimal pour le jeu en mode graphique ou couleur ANSI pour le jeu en mode console).

- **Les sauvegardes**

Le système de sauvegarde est très technique. Le jeu effectue toutes les sauvegardes en binaire, afin d'utiliser le moins d'espace disque possible. Chaque sauvegarde est codée sur 38 octets minimum. La sauvegarde est structurée de la manière suivante (à noter que l'ordre des bits/octets est de haut en bas en Little Endian) :

Grille de jeu de la manche	**32 octets (256 bits)**
Joueur qui a commencé la manche	**2 bits**
Nombre de joueurs	**2 bits**
Nombre d'IA	**2 bits**
Nombre de manches	**2 bits**
Manche actuelle	**2 bits**
Thème actuellement utilisé	**6 bits**
Scores de chaque manches	**Bits restants**

Structure des fichiers et Fonctions importantes

Le projet est divisé en plusieurs fichiers. *main.py* se trouve à la racine et permet l'exécution de l'application. Tous les fichiers ci-dessous sont dans le dossier *modules*, et sont donc des modules permettant différentes fonctionnalités.

- **cmd.py**

  ***cmd.py* est le fichier dans lequel se trouve toutes les fonctions concernant le jeu en mode console.**

  - `display_grid(...)`

    Cette fonction a été faite par Nathan. Elle permet simplement d'afficher la grille de jeu dans le terminal.

  - `mainloop(...)`

    Cette fonction a été principalement faite par Jérémy. C'est la fonction principale du mode console, qui exécute le jeu en mode console.

- **graphical.py**
  graphical.py est le fichier dans lequel se trouve toutes les fonctions concernant le jeu en mode graphique.
  - `display_grid(...)`
    Cette fonction a été principalement faite par Nathan. Elle permet d'afficher la grille de jeu dans une fenêtre.
  - `display_end_window(...)`
    Cette fonction a été principalement faite par Nathan. Elle permet d'afficher le score des joueurs à la fin d'une manche.
  - `menu_window_select(...)`
    Cette fonction a été faite par Nathan. Elle permet d'afficher un menu avec quatre boutons, afin de permettre à l'utilisateur de faire un choix.
  - `themes(...)`
    Cette fonction a été faite par Lilian. Elle permet de créer un bouton pour chaque thème défini dans le jeu, afin de pouvoir les sélectionner dans l'écran des paramètres.
  - `settings_menu(...)`
    Cette fonction a été principalement faite par Lilian. Elle permet d'afficher tout le menu des paramètres.
  - `mainloop(...)`
    Cette fonction a été principalement faite par Nathan. C'est la fonction principale du mode graphique, qui exécute le jeu en mode graphique.
  - `saves_list(...)`
    Cette fonction a été faite par Lilian. Elle permet de lister toutes les sauvegardes trouvées.
  - `name_input(...)`
    Cette fonction a été faite par Lilian. Elle permet de demander à l'utilisateur d'entrer une chaîne de caractère.
  - `save_menu(...)`
    Cette fonction a été principalement faite par Lilian. Elle permet d'afficher l'écran permettant de charger des parties sauvegardées.
- **rolit.py**
  rolit.py est le fichier contenant toutes les fonctions primaires du jeu. On peut dire qu'il contient l'engin du jeu. Toutes les fonctions de ce fichier ont été faites par Jérémy.
  - `check_capture(...)`
    Cette fonction permet de détecter si une boule jouée capture des boules d'autre couleur.
  - `test_adjacent(...)`

Cette fonction permet de détecter si, à un emplacement sur la grille de jeu, des boules sont adjacentes à cet emplacement de manière orthogonale. Elle est utilisée lors de la vérification de la validité d'un coup.

- **`play(...)`**
  Cette fonction permet de placer une boule sur la grille de jeu, si et seulement si le coup est valide.
- **`ai_play(...)`**
  Cette fonction permet de faire jouer une IA. Elle calcule le meilleur coup en fonction du nombre de boules capturées.

- **saver.py**
  ***saver.py* est le fichier contenant toutes les fonctions nécessaires au système de sauvegarde. Toutes les fonctions de ce fichier ont été faites par Nathan.**
  - **`pack_grid(...)`**
    Cette fonction permet de convertir une grille de jeu sous la forme de liste de liste en binaire.
  - **`unpack_grid(...)`**
    Cette fonction fait l'opération inverse de `pack_grid`
  - **`save(...)`**
    Cette fonction permet de sauvegarder l'intégralité d'une partie, sous forme d'un fichier binaire.
  - **`recall(...)`**
    Cette fonction permet de charger une partie à partir d'une sauvegarde.

Organisation du groupe

Comme indiqué dans les Cahiers des Charges, notre équipe se répartissait uniformément les rôles. A chaque implémentation de fonctionnalités, tous les membres devaient être d'accord sur l'idée, ainsi que sur la manière de l'implémenter. Un des membres s'occupait ensuite de faire la base de la fonctionnalité, et ensuite chacun d'entre nous pouvait contribuer afin d'améliorer, optimiser et/ou ajuster le code.
Afin de synchroniser notre avancée dans le projet, nous avons créé un dépôt privé sur GitHub. Ainsi nous pouvions contribuer sans se soucier des conflits potentiels, dues à des modifications simultanées d'un même fichier.
En ce qui concerne la planification, nous avons décidé d'être toujours en avance par rapport à ce qui avait été dit et prévu à chaque rendez-vous. Par exemple, le client exigeait pour le premier rendez-vous uniquement une version du jeu jouable en mode console, mais à ce moment, le mode graphique de notre jeu était déjà

implémenté. Cela nous permettait d'avoir à chaque fois une longueur d'avance sur les fonctionnalités proposées, et ainsi de pouvoir ajuster et d'améliorer celles validées par le client.

Répartition du travail

Contribution en pourcentage:

- **KUNTZ Nathan: 35%**
- **LI Jérémy: 35%**
- **VISTE Lilian: 30%**

Ressenti du groupe

Le groupe marchait très bien. Le travail était effectué avec efficacité. Le déroulement du projet était assez fluide. La collaboration se faisait la plupart du temps sans difficulté grâce à Github et des envois de code coordonnés. Les améliorations auxquelles l'on pensait étaient généralement ajoutées, après s'être concertés sur comment l'implémenter et si cela ajoutait réellement un plus ou pas. Dans l'ensemble, nous avons pu terminer le projet en avance !

Ressentis individuels

KUNTZ Nathan

Ce projet a été une très bonne expérience. Bien que facile, il m'a permis de bien me rendre compte du cycle de projet classique et de travailler avec des personnes agréables à côtoyer.

LI Jérémy

C'était un plaisir de travailler avec ce groupe. On échangeait facilement, ce qui nous a donné la possibilité d'avancer chacun à notre rythme, et notamment m'a permis de continuer même quand on n'était pas en présentiel. Ce travail, même s'il était simple, m'a permis de comprendre les difficultés derrière la réalisation d'un projet en groupe, par exemple le fait de devoir expliquer le code qu'on a écrit.

VISTE Lilian

Notre groupe s'entend très bien de mon point de vue ce qui permet de partager nos avancées et les expériences de certains pour créer un environnement où l'on peut s'améliorer avec les autres. Le projet a bien avancé tout au long de la phase de développement, je suis tout de même content de ce que j'ai pu faire avec mes

connaissances actuelles. J'ai réellement apprécié travailler sur le projet avec ces personnes que je remercie de m'avoir aidé.