Rapport de projet

ROLIT - Anastasia Cordun, Niekita Joseph

Introduction

Le jeu Rolit est un jeu de stratégie dont les règles sont intuitives et assez simples - le programmer, cependant, est un défi à part entière. En commençant par la gestion du jeu en elle-même à l'interface, en passant par les sauvegardes, ce projet présente de nombreuses opportunités de mettre en pratique nos connaissances et des concepts de programmation de logiciels.
Ce projet vise donc à programmer un logiciel performant permettant de jouer au jeu ROLIT, de sauvegarder ses parties et de modifier les paramètres de son jeu comme souhaité.

Le guide d'utilisateur est fourni dans le readme.txt accompagnant le code.

Sommaire

Concernant le code

Structures de données utilisées

Notre plateau de jeu est défini par une liste de liste, chaque liste représentant une ligne du plateau. A une plus petite échelle, des dictionnaires sont utilisés pour passer de certains mots clé à des mots entiers utilisables pour par exemple, les couleurs. Des classes sont utilisées pour enregistrer une partie, ou un joueur.

Bugs connus

Le lancé de dé laisse souvent le joueur vert commencer.
Les couleurs initiales ne s'affichent pas.
(Ces deux bugs ont été fixés dans des versions ultérieures. Après un certain nombre de changements, et faute de temps, la correction des bugs plus importants a été privilégiée au dépend de ces deux bugs.)

Modules utilisés

FLTK - Module obligatoire.
Random - Module utilisé pour le lancé dé.
Datetime - Module utilisé afin d'enregistrer l'heure et la date à laquelle une partie a été commencée.
Pygame - Module utilisé uniquement pour la musique de fond ainsi que les sons de clics.
Math - Module utilisé afin de dessiner le logo de Rolit manuellement.
Sys - Module utilisé afin de mettre fin au logiciel abruptement si nécessaire.
Time - Module utilisé afin de gérer les délais pour les affichages ou les bots.
Pickle - Module utilisé afin d'enregistrer les parties.
Os - Module permettant de vérifier la disponibilité d'un fichier de sauvegarde.

Liste des fonctions majeures

Gameplay principal

`nouvelle_partie (partie.py)`

Développeur principal : Anastasia Cordun, Niekita Joseph

Cette fonction permet d'initialiser une nouvelle partie. Elle contient l'animation du lancer de dé, ainsi que tous les appels de fonctions nécessaires pour l'initialisation.

Le défi ici était l'organisation avec la fonction `play`, de l'ordre des lignes et de l'affichage correct de ce qui était nécessaire.

`play(partie.py)`

Développeur principal : Anastasia Cordun, Niekita Joseph

Fonction principale démarrant la partie à partir d'un plateau de jeu.

En ne prenant en cours qu'un objet de type plateau, cette fonction permet de relancer aussi une partie ultérieurement.

`class Joueurs (moteur.py)`

Développeur principal : Niekita Joseph

Cette classe sert à initialiser les caractéristiques de chaque joueur (son score, son nom, et son identifiant dans le plateau.

Cette façon de le programmer, dans une classe, facilite l'accès à ces quelques caractéristiques et s'avère très pratique dans un programme aussi compact.

`class Plateau (moteur.py)`

Développeur principal : Anastasia Cordun, Niekita Joseph

La classe Plateau est au cœur du gameplay, représentant le tableau de jeu et intégrant toutes les règles nécessaires pour une partie fluide.

Fonctionnalités principales :

`creationtab`

Génère dynamiquement un plateau vide de la taille souhaitée, offrant une flexibilité aux joueurs.

`initialisationcolor`

Initialise les pions de départ, soit en couleurs, soit en formes pour le mode daltonien. Cette méthode garantit une disposition correcte, quel que soit le nombre de joueurs.

`valid_pos`

Vérifie si une position donnée est valide pour jouer un pion, en tenant compte des règles de capture.

`game_on`

Joue un coup à une position spécifique, effectue les captures nécessaires et met à jour les scores.

`capture_direction`

Implémente la logique des captures en explorant toutes les directions possibles (horizontale, verticale, diagonale).

`next_turn`

Gère le passage au prochain joueur tout en mettant à jour l'ordre de jeu.

`fin_de_manche, new_manche`

Assurent la gestion des manches, vérifiant leur fin et réinitialisant le plateau pour une nouvelle manche.

Cette classe centralise les mécaniques essentielles tout en restant modulable et extensible, ce qui a permis d'y intégrer facilement des fonctionnalités comme les bots ou les sauvegardes.

Gameplay - fonctionnalités additionnelles

Développeur principal : Anastasia Cordun

Les bots ajoutent une couche de défi supplémentaire au jeu. Chaque niveau de difficulté a été soigneusement conçu :

`bot_move`

Pour un bot facile, cette fonction effectue un choix aléatoire parmi les mouvements possibles. bot_coup_moyen: optimise les captures en choisissant le coup capturant le plus de pions à chaque tour.

`bot_coup_difficile`

Utilise une évaluation stratégique (evaluate_position) pour maximiser les chances de victoire sur le long terme.

`count_captures`

Calcule le nombre de pions capturés après un coup.

`evaluate_position`

Analyse l'état du plateau pour attribuer une note à une position donnée, utilisée dans les décisions du bot difficile.

Grâce à ces bots, le jeu peut s'adapter aux préférences des joueurs, de l'amusement casual à un véritable défi stratégique.

Sauvegarde

`quick_save, reprendre_partie (saving.py)`

Développeur principal : Niekita Joseph

Ces deux fonctions s’actionnent lorsque l’utilisateur ouvre les paramètres lors d’une partie. Cela lui permet d’enregistrer temporairement sa partie afin d’y revenir immédiatement, les paramètres souhaités modifiés.

La difficulté lors de l’implémentation a surtout été de transitionner entre les différentes pages (partie -> paramètres -> partie, au lieu de partie -> paramètres -> menu, comme la page à été codée précédemment.)

`affiche_sauvegarde (saving.py)`

Développeur principal : Niekita Joseph

Cette fonction affiche la page de sauvegarde ainsi que les différentes options.

La difficulté ici était d'afficher correctement les données souhaités d'une sauvegarde ainsi que les options nécessaires uniquement lorsque le fichier en question en contenait une. Cela a nécessité une utilisation du module Pickle et d'un ajout d'un certain nombre d'objets dans la classe Plateau.

Interface

`afficher_menu`

Développeur principal : Anastasia Cordun

Affiche une page intuitive permettant de modifier les paramètres du jeu : activation ou désactivation de la musique, changement du volume sonore, un mode daltonien pour une meilleure accessibilité et un changement des thèmes graphiques.

Les transitions entre le menu principal, les paramètres et le jeu en cours sont fluides grâce à une gestion minutieuse des boutons et variables globales.

`afficher_parametres (menu.py)`

Développeur principal : Anastasia Cordun, Niekita Joseph

Cette fonction affiche la page de paramètres à partir du menu, ou d'une partie.

Le défi principal était d'avoir une transition claire entre chaque page, ainsi que d'afficher le bouton de retour approprié : Menu, ou Retour vers la partie. Il a été nécessaire de faire appel à des variables globales dans l'entièreté du programme, stockées dans le fichier de référence `config.py`.

`afficher_choix (menu.py)`

Développeur principal : Anastasia Cordun

Affiche une interface interactive pour configurer les paramètres de jeu avant de commencer une partie, incluant des sliders pour ajuster les options.

`afficher_choix_bot (menu.py)`

Développeur principal : Anastasia Cordun

Permet aux joueurs de sélectionner le niveau de difficulté des bots avant de commencer une partie. L'interface est claire, avec des boutons bien espacés et des textes explicites pour chaque option.

`afficher_rules (menu.py)`

Développeur principal : Anastasia Cordun

Cette fonction présente les règles du jeu de manière progressive et animée, rendant leur lecture plus engageante et mémorable. Les animations de texte, associées à un design élégant, renforcent l'expérience utilisateur.

`changer_theme (config.py)`

Développeur principal : Niekita Joseph

Cette fonction permet de basculer entre les thèmes graphiques ("chaud", "froid", "verdure"), renouvelant l'expérience visuelle des joueurs.

Organiser le changement entre toutes les pages à partir d'une seule et même variable fut contre-intuitif au début, et assez long - il a fallu modifier chaque couleur utilisée afin de la synchroniser avec le reste. Une fois fait, cependant, tout est beaucoup plus fluide et clair à coder.

`class Slider (menu.py)`

Développeur principal : Anastasia Cordun

Représente un élément interactif pour ajuster des valeurs (ex.: taille du plateau, nombre de joueurs). Ses fonctionnalités incluent un affichage visuel précis avec graduation et un mise à jour dynamique en réponse aux clics de l'utilisateur. Ce composant modulaire est utilisé dans plusieurs écrans du jeu.

`draw_sidebar (graphics.py)`

Développeur principal : Niekita Joseph

Cette fonction dessine l'interface graphique du menu à droite du plateau de jeu, permettant aux utilisateurs d'avoir des informations nécessaires en un coup d'œil.

Cette dernière utilisant beaucoup de lignes (chaque élément de l'interface nécessite une ligne), une façon de la raccourcir aurait pu être explorée, à l'aide de boucles pour les éléments similaires notamment.

`draw_pions (graphics.py)`

Développeur principal : Anastasia Cordun

Dessine les pions sur le plateau, avec des formes alternatives en mode daltonien, assurant ainsi une meilleure accessibilité.

`gestion_clic (menu.py)`

Développeur principal : Anastasia Cordun, Niekita Joseph

Cette fonction gère la moitié des clics du logiciel (hormis le jeu, les choix initiaux avec les Sliders, la page de sauvegarde et de fin). Elle gère donc le menu principal, les paramètres, et les règles du jeu.

En commençant à coder, nous nous sommes rapidement rendues compte qu'il fallait trouver un moyen de différencier chaque page; sinon, les clics sur un bouton du menu principal allaient être confondus avec les boutons des paramètres.

Pour cela, nous avons utilisé des variables globales (similaire à la gestion des thèmes) stockées dans un fichier de référence, et beaucoup de conditions.

endgame (final.py)

Développeur principal : Niekita Joseph

Cette page s’affiche à la fin d’une partie, donnant les scores finals, le gagnant, et ce qui peut être fait par la suite ; enregistrer la partie, retourner au menu pour rejouer, ou quitter le logiciel.

Audio (audio.py)

Développeur principal : Anastasia Cordun

L'ambiance sonore du jeu est gérée par ces fonctions:

jouer_musique

Joue une musique de fond en boucle pour une immersion totale.

arreter_musique

Interrompt la musique en cours.

ajuster_volume

Permet un contrôle précis du volume, avec une transition fluide entre les niveaux.

jouer_son_clic1 et jouer_son_clic2

Jouent des sons distincts pour les clics dans le menu et pendant le jeu.

L'intégration du module Pygame a permis de gérer efficacement tous les aspects audio, avec une compatibilité multiplateforme.

Cohésion de groupe

Répartition des tâches, planning

La répartition des tâches s'est faite progressivement, en apportant chacune notre pierre à l'édifice à tour de rôle.

GitHub a été un outil majeur dans ce projet, et nous a permis d'avancer tranquillement de notre côté avant de mettre des parties du code en commun. Google Documents nous a permis de construire nos Cahier des Charges ainsi que ce rapport à distance.

Nous avons tout d'abord réparti la construction du jeu principal; une personne s'occupe de l'initialisation et du début de la partie sous forme de liste, puis une autre s'occupe de commencer une gestion des clics sur plateau.

Ensuite, une personne a avancé sur le gameplay et un début d'interface, et une fois la progression partagée, le second membre de l'équipe a amélioré de ce qui a été fait, avant d'ajouter de nouvelles fonctionnalités, comme la fin de l'interface, et une amélioration de la navigation entre chaque page.

Le projet fut conclu avec de nombreux ajouts successifs à tour de rôle, implémentant chacun des corrections où des fonctionnalités supplémentaires.

Estimation de la répartition des tâches

Anastasia Cordun : 50%
Niekita Joseph : 50%

Nous avons chacun programmé une partie égale du jeu, en passant et corrigeant les éventuels bugs sur le travail de l'autre. Les cahiers des charges ont été faits au fur et à mesure à deux, à distance.

Ressenti général du groupe

En débutant la programmation, nous avions quelques doutes quant aux délais à respecter, et au fait de devoir progresser dans ce projet parallèlement aux projets de cours. Avec un peu de temps, nous avons réussi à adopter un certain rythme de programmation et d'échange, et avons pu mener ce projet à bien.

Ressenti individuel

Anastasia Cordun

Ce projet a été une expérience incroyablement enrichissante et inspirante. J'ai adoré travailler sur toutes les facettes du jeu Rolit, qu'il s'agisse de la programmation des mécaniques, de la création des bots ou encore du soin apporté au design. Chaque détail, des animations à l'accessibilité pour les daltoniens, a été pensé pour offrir une expérience fluide et agréable.

Je crois que l’une des choses dont je suis le plus fière, c’est l’ensemble du design du jeu. Des règles claires et animées pour les débutants avec le style des novels visuel/2D RPG,
aux thèmes graphiques modulables, et ce mode daltonien, venue d'une idée aussi aléatoire que géniale, apporte une touche unique pour que chaque joueur se sente à l’aise et pleinement engagé.

J'ai aussi adoré imaginer et coder les différentes pages et fenêtres : paramétrage des joueurs, sélection des bots, menu principal…ces éléments apportent une vraie cohérence et rendent le jeu accueillant.

Travailler sur la logique du jeu a été une expérience à la fois frustrante et amusante : passer des heures à traquer les bugs, puis finalement tout voir fonctionner parfaitement, c’était incroyablement satisfaisant!

Les bots aussi ont été un vrai défi technique. Réfléchir à des niveaux de difficulté progressifs et voir le bot difficile réellement poser problème, fût extrêmement gratifiant.
Cette combinaison entre stratégie et esthétique montre à quel point ce projet a su mêler technique et créativité.

Si j’avais eu plus de temps, j’aurais aimé ajouter encore plus de personnalisation et peut-être de nouveaux modes de jeu.
Mais même tel quel, je suis très fière de ce qu’on a réalisé. Ce genre de projet, avec un équilibre entre logique, design et expérience utilisateur, correspond parfaitement à ce que j’aime faire.
En résumé, une merveilleuse expérience!

Niekita Joseph

Ce projet peut sembler assez intimidant au début; on se demande comment réussir à gérer rien que la capture des pions, et on s’inquiète d’avoir suffisamment de temps pour finir tout ce qu’on aimerait y ajouter. J’avais ces doutes face à toutes les possibilités d’ajouts qu’on aurait pas le temps d’implémenter, mais au final, nous avons réussi à compléter les ajouts principaux à temps.

Cependant, une fois réellement lancés, programmer devient certainement bien plus amusant. Certaines parties du code que je redoutais ont été plus simple à compléter grâce à ma coéquipière. Échanger nos idées et voir ce petit jeu de société se construire en un vrai logiciel fût gratifiant! Je ne pensais honnêtement pas pouvoir autant y insérer nos goûts personnels, et pourtant, nous avons eu maintes occasions de le faire.

Je suis tout particulièrement fière de l’esthétisme du logiciel et de la fluidité de la navigation. Le tout offre un ressenti très proche d’une navigation sur n’importe quel autre logiciel, chose que je n’osais jamais espérer d’un simple programme Python. Le système de sauvegarde, que je pensais long et laborieux, voire complètement impossible, se révéla assez intuitif et compréhensible, ce qui m’a permis de le compléter sans trop de mal. Je trouve que la révélation que certaines choses sont plus faciles qu’elles n’y paraissent est justement assez satisfaisante.

Malgré le fait que coder un programme de cette ampleur fut nouveau pour moi, j’ai pu apprendre au fur et à mesure du projet et suis complètement satisfaite du résultat!