MacGyver must flee

MacGyver must flee est un petit programme de jeu de labyrinthe dont le but est de faire échapper le personnage principal (MacGyver) après qu'il ai collectionnés les objets utiles à son escapade.

A. Explication des choix algorithmiques (elle est faite par fichier)

a. Le fichier maze.py

1. Documentation sur le programme.
2. Importation des fichiers nécessaires au programme (pygame + mes modules)
3. Main loop; la boucle principale du programme. Le programme est en marche tant qu'elle tourne. Au démarrage de cette boucle, on affiche l'écran d'accueil.
4. Repeat loop; l'écran d'accueil s'affiche tant qu'elle continue de tourner. Lorsque le joueur passe l'accueil, on ferme la boucle d'accueil. On génère le niveau, créer le personnage, et affiche le labyrinthe avant de passer à la suite.
5. Game loop; si elle continue de s'exécuter, le joueur joue en déplaçant MacGyver. On rafraîchit l'écran avec les nouvelles positions à chaque coups du joueur. Aussi, chaque coup est soumis à une vérification pour savoir si le joueur a gagné ou perdu. Retour à l'accueil dans les deux cas.

b. Pour le fichier classes.py

1. Documentation du fichier
2. Importation des modules (pygame + constants.py)
3. **classe *Level***; permet de créer un niveau en lui passant le chemin vers un .txt à lire. Elle possède deux méthodes :
   la méthode *generate* permet de générer la structure du niveau et la stocke dans une liste; tandis que *show* affiche le contenu de cette liste.
4. **Classe *Bot*;** elle prend en paramètre le chemin vers un fichier image représentant un personnage et le niveau dans lequel se trouve ce dernier. Sa premiere méthode s'occupe du déplacement du personnage avec en paramètre le sens du déplacement et une liste qui contient les utilitaires à ramasser. Elle vérifie aussi qu'on ne sorte pas de l'écran, que la case de destination est libre avant de se déplacer. La seconde méthode "pickUp" s'occupe du ramassage d'un objet en vérifiant sa

position par rapport à celle du personnage. Tandis que listUtility permet de lister les utilitaires.

5. **La classe *Utility***; prend en paramètre le chemin du fichier image de l'objet et la structure du labyrinthe. Sa méthode "putRandomly" se charge de disposer aléatoirement l'objet, et "isPicked" a pour but d'effacer l'image d'un objet ramassé.

c. Enfin le fichier constants.py regroupe les constantes du programme.

B. **Difficultés rencontrées et solutions retenues**.

La première difficulté concerne le découpage du programme en module puisqu'il fallait faire interagir plusieurs classes entre elles où certains objets sont envoyés en paramètre à d'autre. Pour y arriver, j'ai donc fait le choix de scinder le programme en trois fichier:
un fichier regroupant toutes les classes(pas de programmation javiotique ici), un pour les constantes du programme et le fichier du programme principal.

Le second challenge se trouvait au niveau l'implémentation du code principale: comment faire passer le joueur d'une étape à l'autre ? Ici, les boucles infinies représentent donc une option intéressante. Le script principale en contient jusqu'à trois à inclusion croissante permettant ainsi de gérer les étapes du joueur à travers les évènements de pygame.

Enfin la dernière grande difficulté était la façon de s'y prendre pour faire disparaître un objet une fois que le personnage se positionne dessus. Pour cela, je définis la méthode **pickUp** dans la classe ***Bot*** qui met l'objet trouvé dans le sac à dos du personnage et **isPicked** dans la classe ***Utility*** qui replace le efface l'image après le passage du personnage. Notons que cette dernière méthode est utilisée par **pickUp**.

C. Idées d'amélioration

- Ajouter d'autres obstacles dans le labyrinthe(des gardiens en mouvement).
- Passage secret pour aller dans un autre niveau.

Lien github du projet 3