Parcours Développeur.se d'application Python Rédaction : Fatou SINDJEU

Projet 3 – Aidez MacGyver à s'échapper !

Lien vers le projet

https://github.com/FatSin/oc-gethimout

Démarche

Dans le cadre de la formation de Développeur.se d'application Python, il a été demandé de réaliser un jeu vidéo dont les fonctionnalités sont les suivantes : un personnage mobile (Macgyver) doit se déplacer à l'aides des touches directionnelles du clavier au sein d'un labyrinthe afin de récupérer 3 objets fixes (éther, aiguille, tube) mais dont la position est aléatoirement calculée à chaque ouverture du programme. Le joueur gagne s'il se présente devant un Gardien (statique) ayant récupéré les 3 objets exactement. Sinon, il perd.

Le pré-requis à la programmation a été de se documenter sur le module **Pygame**, qui permet de réaliser des jeux en Python. J'ai donc commencé par suivre le cours *Interface graphique Pygame pour Python*, proposé par Open Classrooms.

La 1ère étape a consisté à coder la génération du labyrinthe, à partir d'un fichier .txt stocké dans le répertoire du projet.
Le fichier contient des symboles, censés désigner les différents éléments du labyrinthe. La translation choisie a été la suivante :
w : mur, 0 : case vide, d : case départ pour Macgyver, f: case du Gardien, n : vide non occupé
J'ai choisi de distinguer les cases vides des cases de remplissage, afin de ne pas générer d'objets dans des cases impossible à atteindre pour Macgyver.
Les cases vides sont des positions possibles à la fois pour les objets, et pour les déplacements de Macgyver.

Ces positions seront par la suite stockées dans une liste dont les éléments seront les seules positions possibles pour Macgyver. Cette approche résoudra 2 problèmes : la génération aléatoire de la position des objets, et le contrôle de la validité des déplacements de Macgyver.

Il a ensuite fallu gérer l'affichage du labyrinthe dans une fenêtre pygame. Il a fallu être particulièrement vigilante quant au bon dimensionnement des images.
La génération aléatoire des positions des objets a été l'étape suivante. Grâce à la fonction **randomint** du module random, il a été aisé de sélectionner au hasard des éléments de la liste des cases vides occupables. A ce stade, le programme affichait le labyrinthe, Macgyver, les objets et le Gardien, tous statiques.

L'étape suivante a consisté à créer la classe Macgyver, afin de pouvoir l'animer. Jusque là, la boucle de jeu était dans la fonction d'affichage du labyrinthe. Ce qui a fini par poser problème. En effet, il fallait à la fois pour la classe récupérer la position initiale de Macgyver , et distinguer phases d'actions et phase d'affichage. L'architecture du code a donc été modifiée. D'un côté, les fonctions de génération et d'affichage du labyrinthe et du Gardien, de l'autre la classe Macgyver qui gère les déplacements et l'affichage du personnage et des objets. Enfin, à part, la boucle de jeu qui rappellera les fonctions d'affichage jusqu'à la fermeture du jeu.
La fonction d'affichage du labyrinthe retourne finalement un objet liste « elements » composé de : un objet Macgyver de type « **rect** » afin de mieux gérer les déplacements du personnage, la position initiale du Macgyver, la liste des positions vides occupables et la position du Gardien.

Ensuite, la gestion des collisions MacGyver/objets puis MacGyver/mur et enfin MacGyver/Gardien a été codée.
Pour la collision MacGyver/mur, il a suffi de n'autoriser MacGyver à se déplacer avec les touches directionnelles du clavier que si la position de destination était dans la liste des positions occupables.
Pour la collision Macgyver/objets, il a fallu convertir en liste les coordonnées en tuple du rect, afin de les comparer aux positions des objets. Un compteur a été ajouté afin de gérer le nombre d'objets récupérés.

Une fois les collisions et les conditions de victoires et de défaite programmées, des images de fin de partie ainsi qu'un menu ont été ajoutés, afin d'améliorer l'environnement pour le joueur.

Enfin, le module **Pylint** a été installé afin de mesurer la conformité du code avec la PEP8. Le score maximal obtenu a été de 5,88.