Projet n°3 : Aidez MacGyver à s'échapper !

Description :

Créer un jeu de labyrinthe en python dans lequel MacGyver aurait été enfermé. La sortie est surveillée par un garde du corps. MacGyver doit ramasser trois objets puis endormir le gardien et trouver la sortie.

Gestion du projet

Le projet a été réalisé en deux partie, la version terminale du Labyrinthe et la partie graphique avec le module pygame. La première partie du projet m'a pris beaucoup de temps car il fallait revoir le langage python. Cette première partie a également été divisée en plusieurs sous parties pour créer les différentes classes et méthodes.

Etapes du projet :

Les étapes pour réaliser le projet :

1. Créer le labyrinth sans l'interface graphique et puis l'afficher dans le terminal
2. Créer les méthodes pour déplacer MacGyver dans le labyrinthe
3. Récupérer le choix de l'utilisateur pour le mouvement de MacGyver
4. Intégrer MacGyver dans le labyrinthe en fonction de commandes utilisateurs.
5. Créer une méthode : check collision pour l'évitement des murs
6. Placer les 3 objets dans le labyrinthe et vérifier si l'objet est bien placé dans une case vide.
7. Ajouter le ramassage des objets par MacGyver avec la méthode check collision
8. Implémenter la partie graphique avec le module pygame

Les classes :

Les deux classes utilisées dans le programme sont stockées dans des fichiers différents.

Class Labyrinth

J'ai choisi une classe Labyrinth pour me permettre de gérer le Labyrinthe et avoir la structure de mon jeu (carte, objet, macyver). Dans cette classe il y'a 3 méthodes :

1. La **méthode load_labyrinth** qui peut permet de charger le fichier txt (map)
2. La **méhode show_labyrinth** qui affiche mon labyrinth
3. La **méthode position_object** qui place mes 3 objets en vérifiant si l'emplacement est vide (pas de mur, pas d'objets).

Class Macgyver

Cette classe permet de gérer le personnage principale (MacGyver).

4 méthodes ont été mises en places :

1. La **méthode search_m** : qui me permet de retrouver la position x et y de MG.
2. La **méthode move_m** permet de mettre un item à la position de mac_gyver
3. La **méhode move_to** gère facilement le déplacement de MG
4. La **méhode check_collision** vérifie si MG peut se déplacer (autorisé ou interdit).

Fichier main.py

Ce fichier contient le coeur de mon projet pour lancer mon jeu on peut également retrouver le module pygame, l'import de mon fichier Labyrinth et les constantes.

Difficultés rencontrées :

J'avais du mal à imaginer le labyrinthe avec la version terminale et comment structurer les instructions du programme.

Solutions trouvées :

Mon mentor m'a demandé de suivre le cours apprendre à programmer en python de Vincent Le Goff sur openclassroom pour être plus à l'aise avec le langage et les algorithmes. Ce qui m'a beaucoup aidé pour la suite du projet. J'ai également suivi des cours de python sur fun-mooc.fr et obtenu une attestion de suivi avec succès.

Points positif

- J'ai appris à utiliser un algorithme pour résoudre un besoin technique.
- J'ai mieux compris la programmation orientée objet.
- J'ai appris à découper le problème en fonctions.
- J'ai appris les bonnes pratiques de programmation en Python 3

Points à améliorer

- Faire beaucoup d'exercices sur les algorithmes en python.
- Créer des jeux en python pour apprendre d'avantage sur les instructions.
- Revoir la programmation orientée objet.

Contraintes

- Versionner le code en utilisant Git et le publier sur Github.
- Respecter les bonnes pratiques de la PEP 8 et développer dans un environnement virtuel utilisant Python 3
- Ecrire en anglais : nom des variables, commentaires, fonctions...

Souleymane Diallo