DA Python Projet 3

Le labyrinthe de Mc Gyver

I) Lien vers le projet 3 sur le compte herodote2242 sur Github :

https://github.com/herodote2242/projet_3_mc_gyver

II) Introduction :

Me concernant, il s'agit du premier véritable programme informatique que je crée. Je n'ai jamais fait de programmation auparavant. J'ai donc mis beaucoup de temps sur ce projet, plus de 2 mois. Il m'a fallu un long moment au début pour commencer à comprendre la logique de rédaction du code en python. Même en ayant suivi les cours sur la plateforme OC, que j'arrive à comprendre en les reproduisant, la logique de l'appliquer à mon cas pratique me dépassait pendant plusieurs semaines. J'ai recherché d'autres sources d'apprentissage, dont un MOOC qui vient de se terminer sur France Université Numérique, et qui va très loin dans le détail de l'utilisation de Python. J'ai également investi dans le livre Learning Python que je trouve très bien expliqué, mais très exhaustif. Toutes ces sources supplémentaires me font enfin sentir un peu plus à l'aise dans le codage en Python.

III) Rédaction et difficultés rencontrées :

Au tout début, je suis parti de l'hypothèse que mon programme devait contenir autant de modules que de grandes notions, et j'ai construit la structure du labyrinthe dans un fichier json. Le choix du fichier json me paraît pertinent puisque j'ai vu en cours qu'il pouvait être chargé en python par une simple fonction load. Le départ m'a donné quelques difficultés, jusqu'à ce que je comprenne comment écrire chacune des lignes contenant les sprites, et que j'arrive enfin à charger la structure. Le fichier structure_modifiable.json comprend 15 lignes de 15 sprites. Les murs sont symbolisés par des dièses, les chemins par des espaces. On peut tout à fait modifier manuellement dans ce fichier des chemins et des murs pour voir le labyrinthe évoluer en conséquence. On évitera cependant de modifier le gardien et la case EXIT : mon code est construit de telle sorte que Mc Gyver est obligé de se présenter sur la case EXIT devant le gardien pour que l'un des deux soit "assomé" par l'autre. Il s'agit en fait de la case où se déroule l'affrontement. La version précédente voulait que le gardien soit interposé entre Mc Gyver et la sortie, mais dans ce cas l'affrontement avait lieu sur encore une autre case, alourdissant le programme.

J'ai donc défini 4 notions et leurs comportements : le labyrinthe en lui-même avec sa structure et son décor ; les objets avec leur méthode d'apparition, de ramassage et de comptage ; le gardien avec ses 2 réactions possibles (s'endormir ou assomer Mc Gyver) et enfin Mc Gyver, avec sa méthode de déplacement et la construction de la seringue.
J'avais donc 4 modules s'appelant structure.py, objects.py, gardien.py, mc_gyver.py. Après plusieurs sermaines en l'état, le code se concentrait en grande partie sur structure.py et n'était pas gérable. Je devais modifier les fichiers afin de créer un module qui serait la "porte d'entrée" du programme, soit le seul à être lancé et qui lancerait automatiquement les autres. Le programme se composait alors comme suit :

- labyrinthe.py
- decor.py
- classes.py

Là encore toutes les classes étaient intégrées dans le seul fichier classes.py, ce qui rendait compliqué la vérification des modifications apportées au fur et à mesure sur les classes. En effet, il s'est avéré inutile de faire tourner le fichier sous Python pour vérifier une seule modification sur une seule classe, quand le reste du fichier n'est pas entièrement débugé.
A nouveau, j'ai donc découpé davantage les modules pour arriver à la structure actuelle, qui permet de travailler et de tester les modules indépendamment les uns des autres:

- labyrinthe.py,
- config.py,
- syringe.py,
- maze.py,
- mcgyver.py,
- application.py,
- et sans oublier structure_modifiable.json.
- A cela s'ajoutent le dossier des images et la musique de fond.

Une partie du code sur les déplacements de Mc Gyver était délicate à penser. Pour commencer, j'ai mélangé des coordonnées x et y dans le code, ce qui faisait prendre des directions contraires à Mc Gyver selon les touches du clavier, ajouté à un problème de mise à jour des cases qui gardaient le portrait du héros affiché même quand il avait quitté la case. Puis j'ai ajouté une partie permettant de faire revenir la case quittée à son état de chemin, et de changer la case d'arrivée en nouvelle case de présence du héros.

Globalement ce qui m'a donné le plus de problèmes, c'était de jongler entre les différentes classes, et d'importer correctement les fonctions et arguments quand ils étaient définis dans d'autres modules.

Autre difficulté : trouver un algorithme efficace pour disposer de façon aléatoire les 3 objects constituant la seringue à chaque ouverture du jeu. La première solution avait le mérite de marcher en théorie, mais elle pouvait statistiquement faire apparaître les 3 objets sur la même case. Il s'agissait d'un algorithme à base de "if" et de fonctions random.randrange. Finalement, c'est une fonction qui liste les sprites disponibles (les espaces dans la structure) dans un premier temps, et une autre fonction qui prélève au hasard 3 positions différentes dans cette liste, qui ont été implémentées pour résoudre efficacement ce problème.

Dernier problème rencontré : la mise en place de la fin du jeu. Elle dépend de la condition de ramassage des trois objets constitutifs de la seringue, qui permet d'endormir le garde et de gagner la partie. Quelque soit le résultat, la fin du jeu doit faire apparaître un message au joueur l'informant s'il a gagné ou perdu, et l'invitant à frapper n'importe quelle touche pour quitter le jeu. Bien que le code prévu pour la mise en place du message de fin n'est pas compliqué en soi, j'ai rencontré une difficulté à le faire se déclencher correctement quand Mc Gyver arrive sur la case EXIT.

IV) Bon jeu !

PS 1 : La branche master est la branche soumise à la soutenance. La branche encapsulation est un axe d'amélioration suggéré par mon mentor, sur laquelle je n'ai pas encore eu l'occasion d'arriver à mes fins.

PS 2 : en 2018, les aventures de Mc Gyver reviennent à la télévision... malheureusement ce n'est plus avec Richard Dean Anderson, je me demande si le nouveau visage d'Angus aura autant de classe et de charisme :). D'après le teaser de la série, il s'agit d'un blondinet qui présente à peu de choses près la même coiffure que l'original... Coïncidence rigolote avec le parcours Python !