MacGyver Escape – Commentaire

https://github.com/Cecilesg/MacGyver_Escape

Le Labyrinthe :

1. Créer le cadre de départ :

J'ai initialisé le repo Git et l'ai envoyé sur Github, ainsi qu'un dossier VirtualEnv que j'ai appelé p3env sur ma machine. Ensuite j'ai pris le temps d'analyser l'idée du labyrinthe et comment le faire fonctionner avec PyGame.

J'ai commencé par créer les visuels pour le jeu en utilisant les éléments graphiques fournis pour le projet afin que chaque élément mesure précisément 40 pixels x 40 pixels et soit adapté à la taille imposée du labyrinthe de 15 colonnes par 15 lignes en utilisant l'application GIMP2.

Puis, j'ai écrit le script lvl qui représente la carte du labyrinthe dans un fichier à part.

Ensuite, j'ai créé trois fichiers :

- constants.py pour lister les variables correspondant aux images et tous les éléments graphiques.
- classes.py pour lister toutes les classes et leurs méthodes, notamment la définition de la carte du labyrinthe et son affichage ainsi que le placement aléatoire des objets, la définition du personnage et ses mouvements ainsi que le ramassage des objets, la définition de l'apparence des objets et leur apparition dans le labyrinthe.
- macgyverescape.py pour exécuter le jeu sous sa forme algorithmique.

Après cela, dans chaque fichier, j'ai initialisé les librairies et modules nécessaires au bon fonctionnement du jeu. Ceci inclut Pygame pour le jeu lui-même, et Random, pour l'aspect aléatoire des objets en jeu.

C'est dans le fichier exécutable qu'on trouve l'affichage du menu de jeu et que la boucle algorithmique globale du jeu se lance. Je donne le choix à l'utilisateur de commencer à jouer ou de quitter l'application, puis une fois le choix fait on arrive sur le labyrinthe lui-même.

2. Animer le personnage :

L'animation du personnage se fait dans le fichier classes.py dans la classe Character qui définit l'apparence de MacGyver, sa position et sa capacité à bouger dans la méthode move. Dans cette méthode je lui donne aussi la capacité de faire l'action du 'ramassage', c'est à dire qu'en bougeant MacGyver sur les cases qui ont un objet, cela fait 'disparaître' l'affichage de l'objet.

Dans l'exécutable, je load MacGyver, je m'assure que le jeu réponde aux mouvements des flèches du clavier, et en bonus que MacGyver se déplace si on les presse en continu.

3. Récupérer les objets :

Pour la gestion des objets, il a fallu plusieurs étapes dans le fichier classes.py : tout d'abord en s'assurant grâce à une liste, des emplacements libres dans le labyrinthe où Macgyver peut se déplacer. Puis en donnant à Macgyver la capacité de ramasser jusqu'à trois objets, en créant un compteur et en faisant 'disparaître' les objets ramassés. Et enfin en affichant les objets dans les cases libres de notre liste de façon aléatoire.

Dans l'exécutable, je load les objets et leurs images, puis je les fais disparaître en fonction de la position de MacGyver et pour toujours s'il est 'passé' dessus.

4. Gagner ! :

Le code de la victoire se trouve dans l'exécutable, dans la dernière boucle du jeu qui a deux coLe code de la victoire se trouve dans l'exécutable, dans la dernière boucle du jeu qui a deux conditions : ce qui se passe si MacGyver se présente sur la case de fin avec 3 objets ramassés, soit le retour de l'utilisateur au menu du jeu et l'affichage dans la console du message de victoire. Ce qui se passe si Macgyver se présente sur la case de fin avec moins de 3 objets, soit le retour de l'utilisateur au menu du jeu et l'affichage dans la console du message de défaite.

Le choix de l'algorithme :

Ayant besoin de répéter plusieurs fois des instructions dans le script de mon labyrinthe ou encore de voir des actions programmées se continuer, j'ai donc choisi de le coder avec Pygame en utilisant l'algorithme pour le faire fonctionner. Ici j'utilise un algorithme principal sous la forme d'une boucle conditionnelle et de sous-boucles sous la forme conditionnelle et inconditionnelle.

En effet, le jeu en lui-même est une grande boucle 'while' ou 'tant que' et des boucles internes en forme de 'while' et de 'for' qui veut dire pour. Donc tant que condition x est remplie, il se passe y, et pour x on obtient y. J'ai utilisé les conditions if, elif, else, en français, si, sinon si, sinon, pour préciser encore plus les actions dans le jeu.

Mon algorithme a un début et une fin et cela permet d'exécuter le jeu à l'intérieur en fonction des conditions que j'ai fixées et de leur accomplissement ou pas. La boucle du jeu se répète différemment en fonction des choix de l'utilisateur, mais ces choix sont scriptés et fermes dans l'algorithme.

Les difficultés rencontrées :

La grosse difficulté pour moi a été l'intégration des objets. Jusqu'à commencer le travail sur l'intégration des objets, j'avais globalement un jeu qui fonctionnait jusqu'à la victoire en déplaçant MacGyver sur la case de fin. Afin d'intégrer une nouvelle dimension au jeu avec les 3 objets placés aléatoirement dans le labyrinthe, il m'a fallu faire des changements à mon code d'origine.

Tout d'abord j'ai travaillé à comment afficher les 3 objets dans mon labyrinthe avec des coordonnées fixes, à ce moment-là j'avais déjà décidé avec mon mentor que l'aspect aléatoire serait le dernier élément de travail. Une fois que cela a fonctionné, j'ai encore buté non pas sur le fait de créer un compteur pour compter jusqu'à 3 objets à l'affichage et au ramassage, puisque je l'avais prévu dès le départ, mais pour le concept de désafficher ou effacer les objets en jeu une fois que MacGyver se déplace dessus et surtout, de les garder effacés continuellement.

En effet, j'ai fini par réussir à conceptualiser l'effacement des objets quand MacGyver est dessus, mais comment continuer à ne plus les affichés une fois ramassés ? La solution m'échappait.

Mon mentor m'a donc solutionné de créer une liste vide qui va compulser toutes les cases libres sur lesquelles les objets peuvent apparaître, puis de créer un compteur qui s'incrémente à chaque fois que MacGyver passe sur un objet mais aussi retransforme la case 'i' dans la liste pas une case '0' ou vide, une fois que MacGyver est passé dessus et que le compteur s'est incrémenté de 1 jusqu'à 3.

Ceci a fait d'une pierre deux coups pour la fonctionnalité de ramasser les objets mais aussi de les effacer une fois ramassés, et surtout grâce à la liste de cases 'i' disponibles, de trouver à chaque fois des cases libres aléatoires pour le placement des objets dans le jeu.

Une fois cela déterminé, il ne restait plus qu'à utiliser le mode choice de la librairie random pour rendre aléatoire la position des objets et les afficher pour de bon dans la grille de mon labyrinthe.