Projet 3 : Aidez MacGyver à s'échapper

Pour ce projet, on nous demande de créer un jeu où le personnage MacGyver se trouve dans un labyrinthe et doit trouver la sortie mais avant de trouver la sortie, il doit récupérer 3 objets qui seront positionnés aléatoirement sur le labyrinthe.
Une fois ces 3 objets récupérés, il pourra se présenter à la sortie et endormir le gardien.
Je vais donc présenter comment j'ai solutionné mon projet, expliquer les choix de mon algorithme ainsi que les difficultés que j'ai pu avoir pour développer mon jeu.

Algorithme :

Présentation :

Au vu de mon manque d'expérience et de ma méconnaissance totale concernant le développement, j'ai tout de suite rencontré d'énormes difficultés à démarrer le projet, je ne savais vraiment pas par quoi commencer.
C'est pour cela que mon mentor m'a beaucoup orienté au départ afin de me faire comprendre que je devais impérativement décomposer mon problème en plusieurs sous problèmes.
Et surtout la solution pour moi a été de créer le jeu du début jusqu'à la fin sans interface graphique et je dois avouer qu'en procédant de cette manière, cela m'a beaucoup aidé à y voir clair.

Disposition :

Mon jeu est divisé en plusieurs fichiers distincts :

- Un fichier par classe (classe labyrinthe, classe position)
- Un fichier qui permet de lancer le jeu et peut être considéré comme la fonction principale
- Un fichier sous format .txt qui contient le plan du labyrinthe
- Les fichiers images indispensables pour l'interface graphique
- Le fichier « requirements.txt » qui précise quel module installer pour le bon fonctionnement du jeu

Mon programme est composé de deux classes :

Une classe labyrinthe :
Cette classe permet entre autre d'ouvrir le fichier txt qui contient le plan du labyrinthe et d'enregistrer par coordonnées la localisation des murs et tous les autres éléments du labyrinthe sous forme de labyrinthe.
En effet, les coordonnés correspondent à un tuple qui correspond à un clé du dictionnaire qui sera associé à « m » pour le mur par exemple.
Une fois, la disposition du labyrinthe enregistré, il est relativement facile de manipuler le dictionnaire en toute liberté pour y placer de manière aléatoire les objets.
Grâce au module « random », on peut obtenir des coordonnées aléatoirement et grâce au dictionnaire on peut facilement vérifier si les coordonnées peuvent recevoir l'objet ou non (si il y un mur ou autre)
Une fois le labyrinthe complété avec les objets, on peut passer à l'interface graphique pour placer les images correspondantes
Cette classe est doté ensuite de la méthode « playing_game » qui représente la boucle principale du jeu et qui capture les évènements du clavier grâce au module « Pygame », c'est dans cette méthode

également que l'on va appeler d'autres fonctions dans le but de tester si le déplacement est possible et où l'on va vérifier également si le déplacement s'effectuera sur un objet et dans ce cas, il faudra ajouter l'objet à notre liste « object_picked_up »
Tous ces tests sont possible grâce à notre classe utilitaire « position »

Classe position :

Cette classe contient uniquement que des méthodes de classes car nous instancierons aucun objet de la classe position. Cette classe est plus une classe utilitaire très pratique pour effectuer les tests de cases et d'éventuels déplacement.
C'est donc cette classe qui déplacera le personnage et qui communiquera directement avec la classe labyrinthe.
Elle vérifie si le déplacement se fait bien dans le cadre du labyrinthe, c'est-à-dire si les coordonnées sont bien comprises entre 0 et 14. Si les nouvelles coordonnées sont bien sur une case « 0 » qui correspond à un passage libre.
Une fois, toutes les vérifications faites , on appelle la méthode « moving_gyver » de la classe labyrinthe avec comme argument les nouvelles coordonnées du personnage Macgyver afin de le déplacer sur la nouvelle case.
Enfin, on réutilise les méthodes de Pygame afin de faire apparaitre le personnage sur la nouvelle case à l'aide de la méthode « draw_laby » , on va réitérer les boucles et réanalyser les placements des éléments puis à l'aide de « blit » on recolle les images au bon endroit puis à la fin on fait « pygame.display.flip() » ce qui va faire une mise à jour de cette interface graphique.

Fin du jeu :

Au fur et à mesure que MacGyver ramasse les objets, on affiche à l'écran sous forme de texte l'objet qui a été ramassé, on peut également vérifier ce que contient le sac de MacGyver en appuyant sur la touche « b » pour bag.
Une fois que MacGyver arrive sur la case de sortie c'est-à-dire « e » dans le labyrinthe, on va tester grâce à la méthode « ending_game » si la liste « object_picked_up » a bien 3 éléments à l'aide de la méthode « len » si c'est le cas on peut afficher que le joueur a gagné le jeu, à l'écran.
Si la liste possède moins de trois éléments, cela signifiera que MacGyver se retrouve devant le gardien sans les trois objets et dans ce cas, on pourra afficher que le joueur a perdu le jeu, et qu'il peut fermer la page grâce à la touche « q »