Projet 3 : Macgyver Escape Game.

Github : https://github.com/pierro145/MCgyver

Lors du commencement du projet, j'ai défini l'ensemble des étapes qui me serait nécessaire à l'élaboration du jeu sur papier.

- L'élaboration du labyrinthe.
- Les mouvements du personnage.
- Les objets à disposer de façon aléatoire.
- Les objets à récupérer.
- Le placement du gardien.

J'ai dans un premier temps créé mon espace sur gitub et créer mon environnement virtuel puis j'ai effectué les paramétrages nécessaire de mon environnement de travail sous VScode.

Ensuite, je me suis interrogé sur la façon d'interpréter un labyrinthe sur en python et le rendre visible en mode terminal.

Après quelques jours, je ne parvenais pas à mettre un point de départ et écrire ma première ligne de code, en effet je ne parvenais pas à comprendre comment j'allais pouvoir créer ce labyrinthe.

1. Elaboration du Labyrinthe

Après le parcours de la documentation officiel sur python.org en complément des cours présents sur Openclassroom, j'ai utilisé une méthode permettant de lire un fichier et dans notre cas un fichier Txt ou j'avais préalablement dessiné une structure de 15 par 15 caractères.

Une fois bien assimilé que la fonction « enumerate » vérifiait chaque caractère du fichier, j'ai par la suite créé une classe Position me permettant d'enregistrer les positions de mes caractères dans « l'espace » grâce à une double boucle « For ».

Ainsi lors de l'exécution de cette boucle, j'ai pu lors de son passage, stocker les différents caractères et leurs positions dans des listes que j'avais définies au préalable.

Je disposais ensuite de mes listes contenants les murs, les passages, etc.

Cela m'a permis de faciliter par la suite l'implémentation de la surcouche graphique Pygame.

Une fois, ma méthode « *build* » de mon labyrinthe créé, alimentant l'ensemble des listes, j'ai implémenté la méthode « *show* » en utilisation la fonction « *PRINT* »

Exemple : « *if position in self.free* » cela va « *PRINT* » en terminal tous les passages de mon labyrinthe et ainsi de suites pour les murs et personnages.

2. Gestion des mouvements de MacGyver

Une fois ma classe « *position* » finalisée et mon labyrinthe correctement affiché grâce à la gestion des listes. L'élaboration du mécanisme de déplacement de McGyver fut assez simple à réaliser.

Il me suffisait de mettre en place la méthode suivante : « *si la position demandée est libre dans la liste alors la nouvelle position de macgyver est la suivante* ».

Grâce cette méthode le risque d'erreur est limité car l'ensemble des positions valides sont enregistrées dans une liste.

3. **Ajout des items de façon aléatoire sur le labyrinthe**

Une fois la méthode de déplacement réalisée et fonctionnelle, je me suis attelé à la création de ma méthode sur l'ajout des items de façon aléatoire dans le labyrinthe.

Ici le choix de l'algorithme initial pour la construction de mon labyrinthe m'a facilité les choses car j'ai tout simplement grâce au module « random » créé la méthode suivante : « *return sample(liste et nombre de position voulue) »*

Cette méthode lors de son appel affiche les items à des positions forcément libres.

4. **Attraper les items lors du passage de MacGyver sur l'item**

Pour créer le mécanisme de récupération d'items, j'ai simplement effectuée une condition *« si ma position correspondante au personnage macgyver est égale à une position présente dans ma liste de passage alors j'ajoute l'élément dans une nouvelle liste »*

L'élément ne fait donc plus partie de la liste contenant les passages et fait partir de la liste correspondant à l'item, l'élément disparaît alors du labyrinthe au passage du personnage MacGyver.

4.1 **Calcul de l'inventaire lors du passage au gardien**

Pour cette méthode il m'a suffi de créer une condition et d'utiliser la fonction « *len* » si le nombre d'items est égal à trois lors du passage du garde alors MacGyver a gagné et peu s'échapper sinon il meurt et le jeu s'arrête.

5. **Implémentation de la couche graphique avec le module Pygame**

Je n'ai pas vraiment rencontré de difficulté lors de l'implémentation de la couche graphique, car j'ai fait mon possible pour séparer le « gui » de la partie créé en amont en mode console. J'ai donc créé une nouvelle branche nommée « graphic » sur git puis j'ai commencé à appliquer mes changements pour ajouter une interface graphique. Il m'a suffi ensuite de créer une nouvelle classe que j'ai nommé « Display » d'initialiser mon module pygame et ses différentes variables. J'ai ensuite pu créer mes méthodes en appelant les éléments que j'ai créé au préalable lors de la création de mon labyrinthe en version console.

6. Conclusion

Le plus difficile pour moi a été de trouver le bon mécanisme pour l'élaboration du labyrinthe, une fois mis en place et fonctionnel, j'ai vraiment gagné du temps pour l'implémentation des autres fonctionnalités.Mon jeu est pleinement fonctionnel en mode terminal, une branche dédiée a été créé sur git.