Mentor : Thomas Gaël

Mentor de soutenance : (à définir)

Code source : (mettre le lien github)

Ce document résume le projet 3 et s’inscrit dans le cadre de la formation de développeur d’Application Python d’OpenClassrooms.

Objectif du projet : Aider MacGyver à s’échapper du labyrinthe en récupérant des objets disséminés au hasard dans le labyrinthe afin d’endormir le garde surveillant la sortie.

En complément du fichier Readme.md, ce document décrit plus spécifiquement la démarche adoptée face aux contraintes du projet ainsi que les solutions techniques apportées guidant le choix de l’algorithme telle que présentée dans sa version finale.

A l’image de la construction d’une maison, la démarche adoptée s’articule autour de 2 axes distincts :

I°) Concevoir l’architecture interne du jeu

II°) Afficher la décoration du jeu

Le choix d’une telle démarche permet de se focaliser dans un premier temps sur la logique interne du jeu avant de se concentrer sur ses aspects graphiques.

# Concevoir l’architecture interne du jeu

1. Créer l’environnement de jeu

* Créer le labyrinthe

Le labyrinthe est crée à partir d’un fichier que nous avons nommé map.txt. Cette dernière contient une esquisse de notre carte. Sur cette carte sont déjà affichés plusieurs symboles :

- « M » pour Macgyver

- « # » pour mur

- « \_ » pour une case vide

- « G » pour Guardian

Les objets à ramasser par Macgyver ne sont pas encore représentés à ce stade. En effet, selon les contraintes du projet, ces derniers doivent être disposés de manière aléatoire sur des cases vides du labyrinthe. Afin d’anticiper leur future disposition, nous avons choisi arbitrairement leurs symboles de la façon suivante :

- « E » pour Ether

- « T » pour Tube

- « N » pour Needle (aiguille)

Une fois entré le nom de la carte dans le terminal lors du démarrage du jeu, la carte map.txt est ensuite prise en charge dans un « Sélecteur de carte » (Class MapSelector). Chaque ligne contenue dans carte.txt est ajoutée à une liste initialement vide (self.map). Cette étape permet alors de générer la matrice du labyrinthe.

* Créer les personnages

Macgyver et le gardien sont crées à partir de la classe nommée Character (personnages). Leurs emplacements respectifs sont représentés grâce à leurs coordonnées suivant un plan en deux dimensions : pos\_x indique la position des personnages au niveau des lignes et pos\_y au niveau des colonnes. Ces dernières sont mises alors en paramètre lors de la création des personnages afin de pouvoir identifier leurs positions dans le labyrinthe.

* Créer les objets

Les objets (ether, tube et aiguille) sont crées à partir de la classe ItemMgmt (Item Management : Gestion des objets). Tout comme les personnages, ces objets possèdent des coordonnées (pos\_x, pos\_y).

* Gérer les mouvements du héros dans notre jeu
* Se déplacer (héros/labyrinthe)

Une fois notre personnage Macgyver crée dans notre classe Character, celui-ci peut se déplacer dans le labyrinthe grâce aux méthodes move\_left, move\_right, move\_up et move\_down

1. Gérer les interactions du jeu entre les différents objets

* Interaction héros/labyrinthe

Si le déplacement s’effectue vers un mur, le déplacement n’est pas autorisé et le message "Vous êtes face à un mur, déplacement impossible." s’affiche alors dans le terminal.

* Interactions items/labyrinthe

La disposition aléatoire des objets au sein du labyrinthe s’effectue en deux moments :

- Création d’une boucle while : Tant que la position de chaque élément n’est pas une case vide (et donc un mur), une nouvelle position est générée au hasard grâce à la fonction randint du module random.

- Une fois que la boucle tombe sur une case vide, on sort de la boucle et la position de l’objet au sein du labyrinthe est effectuée grâce à la fonction position\_item avec en paramètres le nom du labyrinthe et le symbole de l’objet choisi (E, T ou N dans notre cas).

* Interaction héros/items : récupérer un objet

Cette action est prise en charge par la fonction pick\_item.

* Interaction héros/gardien : fin du jeu

Lorsque le héros se présente devant le gardien, il remporte la partie s’il possède les trois objets. Sinon, il perd.

# Afficher le jeu graphiquement

1. Afficher les éléments du labyrinthe

Une fois le « squelette » de notre programme terminé, l’affichage s’effectue à l’aide de la librairie Pygame.

Tout ce qui concerne l’affichage est géré via la classe « afficheur » (fichier displayer.py). Plusieurs méthodes de cette classe permettent d’afficher les différents éléments. A chaque méthode est associée à une tâche spécifique :

|  |  |
| --- | --- |
| Nom de la méthode | Tâche associée |
| display\_map | Afficher le labyrinthe sur écran. |
| display\_inventory | Afficher le nombre d’objets récupéré par MacGyver. |
| display\_victory\_text | Afficher un message en cas de victoire. |
| display\_failure\_text | Afficher un message en cas de défaite. |
| display\_stop\_game | Afficher un message pour indiquer comment arrêter le jeu. |

1. Gestion des événements

La gestion des événements s’effectue grâce au fichier keyMgmt (Key Management : Gestion des touches). La méthode select\_key associée permet de gérer les touches saisies par le joueur.

Difficultés rencontrées

Il s’agit du premier projet pour un étudiant complètement débutant en langage Python. L’apprentissage des concepts s’est donc fait de façon progressive et plutôt lente (Cours, lecture de forums spécialisés, tutoriels vidéos). L’assimilation du concept de classes et en général de la programmation orientée objet m’a paru particulièrement ardue pour un débutant comme moi.

A travers ce projet, j’ai également mesuré l’importance de ne pas passer directement dans la rédaction du code : Prendre un papier et un crayon pour esquisser la trame du programme avant de passer à sa transcription en langage Python semble une bonne méthode de travail permettant d’éviter des relectures parfois chronophages, notamment lors du découpage des fonctions et des modules.