Soutenance du projet 3 : Aidez MacGyver à s'échapper !

Julien MILLOT

(Alias « julien millot » sur OC, alias « Jilvo » sur Github)

1 : Créer le cadre de départ :

https://github.com/Jilvo/MacGyver

Après la création du repo Git, j'ai commencé par lire les cours de python pour faire la structure et les déplacements du personnage puis lorsque la plupart du cadre était terminé j'ai commencé par apprendre un peu Pygame.

Initialement j'ai créé le labyrinthe sans interface graphique, j'ai choisi tout d'abord de créer une liste de liste qui générait mon labyrinthe mais je m'orientais dans la mauvaise direction par rapport aux consignes du projet, j'ai donc ensuite créé un niveau dans un fichier texte de 15 lignes avec chacune 15 caractères pour représenter les cases disponibles, les murs, MacGyver et le gardien.

La lecture de labyrinthe se fait par deux imbriquées, une pour les lignes et une pour les colonnes. Cela permet d'avoir une représentation du niveau sous la forme de liste de listes. Cette liste est stockée dans la classe *Labyrinth*. Cette classe contient deux méthodes en plus de l'initialisation, une pour trouver MacGyver, et une pour initialiser les objets.

2 : Animer le personnage :

Les déplacements de MacGyver sont gérés à l'aide des méthodes de la classe *Perso* via une méthode de déplacement qui gère le mouvement du personnage en vérifiant que le personnage ne sorte pas de l'écran ou de dépasser les murs, elle gère aussi l'interaction avec le gardien et avec les objets.

La première version « vraiment » jouable permet de gagner ou perdre en fonction du nombre d'objets ramassées.

3 : Récupérer les objets :

La gestion des objets se fait via la méthode *set_the_object*, qui permet de représenter les objets avec pour chacun les attributs suivants : le nom de l'objet, son initial qui est ajouté dans le labyrinthe, ses coordonnées en abscisses et en ordonnés. La méthode gère aussi si un objet est déjà présent à l'endroit générées aléatoirement, ou s'il y a un mur en réaffectant une nouvelle place libre à l'objet.

Les objets avec leurs coordonnées sont placés dans une liste qui permet d'avoir accès à l'ensemble des objets sur le niveau.

4 : Gagner ! :

Pour gagner le jeu, les conditions à remplir sont qu'il faut ramasser les 3 objets et donc avoir un score de 3, et d'atteindre la case du gardien. A chaque déplacement, la case est vérifiée pour savoir si le gardien y est présent et agir en fonction du score.

Le jeu répond bien au cahier des charges qui était présentait comme sujet du projet.

Cependant, il serait possible d'y apporter des améliorations, notamment sur les graphismes en ayant des animations, pour les directions, transition lorsque l'on ramasse un objet…

On pourrait aussi améliorer le jeu en rajoutant des niveaux, un menu, un tuto, et un écran de fin.

On pourrait créer une IA sur le gardien qui le permettrait de se déplacer.

Problèmes rencontrés :

Trouver comment organiser le programme m'a paru compliquer jusqu'à ce que mon mentor me demande de faire une pseudo code.

J'aurai dû faire des le début un code structuré avec des classes, méthodes, etc… plutôt que de tout mettre dans une main qui à la fin ne ressemblait plus à rien (et en utilisant des variables globales en plus). C'était une source de nombreux bugs et relectures du code lors du déplacement des fonctions dans leurs modules respectifs.

J'ai eu quelques problèmes en appréhendant PyGame car il est assez strict sur certaines pratiques, notamment le fait que l'on ne peut pas « déblité » un objet.