Projet 3 : Save MacGyver !

1. Introduction du projet

Dans le parcours développeur Python sur Open Classroom, il nous est demandé dans le Projet 3 de développer un jeu vidéo mettant en scène MacGyver perdu dans de labyrinthe. Il doit s'en échapper et pour ce faire MacGyver doit récupérer trois objets disposés de manière aléatoire dans le jeu pour fabriquer une seringue en vue d'endormir le garde et de pouvoir s'évader. Ce document relate mes périples lors de la réalisation de ce jeu.

2. L'algorithme

1. Introduction (Configuration initiale)

Pour la création du code je me suis appuyé sur les module Pygame et Random et me suis servi de l'éditeur sublime que j'ai couplé avec le package Anaconda en fin de projet. Si dans un premier temps j'avais commencé à coder sur l'OS Windows, j'ai rapidement rencontré des difficultés quant à l'aspect peu pratique et fluide d'IDLE qui m'ont poussé à adopter le système Linux afin de passer moins de temps sur la configuration système et m'attaquer à l'essentiel du travail de manière plus sereine.

2. Le code du jeu

Sous les conseils de mon mentor, j'ai fais le choix de divisé mon code en trois fichiers .py distincts : Un fichier Main qui comporte les fonctions principales (Maze.py) qui initialise la fenêtre du jeu, un fichier Class.py qui regroupe les classes et enfin un fichier Constantes.py qui définie les variables liées principalement au éléments graphiques et visuels du jeu. En répartissant ainsi de manière rationnelle et logique mon code en trois parties, il devient plus simple a lire et a modifier.

- Importation des librairies et des fichiers de constantes :

Au début de mon fichier Maze.py j'importe la librairie Pygame , le module Constantes.py, Classes.py et Random. Puis ensuite viens l'initialisation du module PyGame et de la fenêtre du jeu. La taille de celle-ci découle d'une variable " Window_size " pour chacun de ses cotés que je définie dans le fichier Constantes.py comme le résultat de multiplication du nombre de sprites (15) et de leurs tailles en pixels (30). Je reviendrais plus tard sur cette formule pour fixé un valeur fixe en Int de 480 pour la hauteur de la fenêtre afin de laisser une marge noire au dessus.

Dans la fenêtre je me sert ensuite de la fonction " blit " pour afficher un fond visuel de couleur sable, zone que j'étire sur l'ensemble de la zone de jeu.

Au même niveau je situe l'initialisation des visuels comme les murs, personnages et objet avec l'aide de la fonction display. L'emplacement de ces images étant définie dans des variables du fichier " Constantes.py "

- Les Classes :

J'ai créer trois classes pour la réalisation de mon jeu :

La classe Level : cette classe a pour fonction principale de définir le parcours du labyrinthe. Pour ce faire j'ai créer deux fonctions " generate " et " display ", la fonction generate avec la fonction " with open " va aller puiser dans un fichier " Level.text " dans lequel se trouve le plan du labyrinthe sous la forme de caractère ascii représentants les éléments du jeu comme les murs (m),

les vide (0) et l'emplacement initial de MacGyver (d) et du gardien (a). Je demande à Python de lire ce fichier ligne par ligne, sprite par sprite tout en lui indiquant de procéder à la ligne suivante en bout de ligne (15 caractère par ligne, 16 lignes). Le programme stockera sous forme de liste les éléments puisés dans le fichier Level.txt dans une variable " structure "

La fonction display va prendre le relais pour remplacer les éléments de la variable structure créer précédemment par leur équivalent visuel. Le script va ensuite attribuer des coordonnées X et Y aux éléments en fonction de l'ordre d'apparition des caractères dans la liste " structure " et affichera les images des murs et du gardien dans la fenêtre du jeu sur à ces même coordonnées. C'est la classe qui m'as posé le plus de difficultés à définir. Je me suis aidé d'un tutoriel d'Open Classroom sur le module Pygame (DK Labyrinthe) dont j'ai réussis à décortiqué le code après plusieurs heures d'observations et de tâtonnements.

La classe Char : La classe du personnage de MacGyver dispose d'une unique fonction " mooving " qui gère les déplacement du personnage de case en case avec les touches du clavier à l'aide de la fonction " pygame.event.get() ". Elle à l'apparence d'une structure conditionnelle en « if » qui vérifie que plusieurs conditions sont remplis avant d'autoriser le mouvement. Elle compare les données en X et Y (self.case x, self.case.y) de MacGyver avec les coordonnées des éléments de level.structure pour voir si les nouvelles coordonnées de MacGyver correspondent avec un zone libre (soit un o dans le fichier Level.txt) et non un mur (m). Elle dispose également d'une condition relative à la direction voulue interdisant le déplacement hors de la zone de jeu. Exemple pour la gauche : if self.case_x < (Nbr_Sprite_Side – 1)

La classe Loot : La classe des objets à une fonction display qui va répartir les objets au sein du labyrinthe dans les zones libres de level.structure (les 0), l'aspect aléatoire du process est gérer par la fonction randint du module random qui génère un chiffre entre 0 et 14 pour définir les variable case_x et case_y qui seront transformés en coordonnées dans les variables self.y et self.x avec la formule self.case_y * Sprite_Size. La méthode de ma classe Loot contient également un boléens self.loaded qui est True au début de la boucle est qui devient False lorsque que les coordonnées auto-générées par randint équivalent à celles d'un espace vide dans le niveau.

- Compteur d'objet et Endgame :

Pour donner l'illusion que Macgyver ramasse les objets lorsqu'il marche dessus, je compare dans mon fichier Maze.py au sein de la boucle principale les coordonnées de celui-ci avec celles des objets réparties précédemment dans le labyrinthe, chaque objet étant une instance de la classe Loot. L'affichage visuel des objets à leur position dans le labyrinthe et dépendante d'un boléens (Nom de l'objet-NotPicked)- Si Macgyver marche sur un objet, ce boléens devient False et la position de l'objet est modifié pour être déplacé en haut de l'écran dans la bordure supérieure noire.

Quand MacGyver (Mac.x, Mac.y) atteint la case du gardien (a) deux variables booléennes sont testées pour savoir si il gagne ou perd : Si toutes les variables d'affichage des objets dépendante de la classe loot sont passée de True à False, alors tout les objet sont ramassés, la variable GAME_WON devient True et l'écran Win s'affiche, dans toute autre cas, il perd et c'est la variable GAME_LOOSE qui devient True et l'écran de Game Over qui s'affiche.