Projet 3 : Aidez MacGyver à s'échapper !

Dans le cadre de ma formation chez Open Classroom en tant que Développeuse d'application -Python, il m'a été demandé de développer un jeu vidéo en 2D mettant en scène MacGyver perdu dans un labyrinthe.

Pour s'en échapper MacGyver doit ramasser trois objets disposés aléatoirement et se rendre sur la case du garde à la sortie du labyrinthe.

I. Introduction

Pour le développement du jeu je me suis appuyé du module Pygame qui m'a grandement aidé pour la construction du labyrinthe et les déplacements du personnage.

L'ensemble de mon code à été rédigé avec IDLE.

II. Structure

Afin de ne pas surcharger le fichier main j'ai décidé de diviser mon code en deux fichiers distincts.

« mac_gyver.py » fichier main dans lequel se trouve la boucle principale du jeu et les conditions de victoire.

« classes.py » fichier dans le lequel se trouve les classes du jeu et ses fonctions.

III. Les classes *« Level, Player, Loot »*

Class Level : cette classe a pour fonction principale de générer et de définir le parcours du labyrinthe.

Pour ce faire j'ai créé deux fonctions « **generate** » et « **display** »

La **fonction generate** qui utilise le fichier « level.txt » dans lequel se trouve la structure du labyrinthe sous la forme de caractères ascii représentant les éléments du jeu (w) pour les murs, (0) pour les vides, (e) pour le garde, (b) et (i) pour l'inventaire. Je demande à python de créer une liste **[ ]** et de lire ce fichier ligne par ligne en lui indiquant de faire des retours à la ligne avec **\n** à chaque fin de ligne et d'en ajouter une nouvelle avec .**append**.

La **fonction display** me permet de remplacer les éléments de cette liste par ses équivalents en image.

Class Player : cette classe a pour fonction de gérer les déplacements de MacGyver dans le labyrinthe.

Une seule fonction « **update** »

La **fonction update** gère les déplacements du personnage de case en case en vérifiant que chaque condition soit vrai avant de permettre le déplacement de MacGyver en comparant les coordonnées en

x et y (**self.case_x, self.case_y**) du personnage avec la structure du labyrinthe si **!= 'w'** il peut avancer d'une case.

Class Loot : cette classe me permet de générer les objets dans la structure du labyrinthe.

Une seule fonction « **display** »

La **fonction display** va répartir les objets aléatoirement avec **random.randint** dans la structure du labyrinthe. La méthode de ma classe contient un booléen **self.loaded** qui est **True** au début de la boucle et qui devient **False** lorsque les coordonnées générées par **random.randint** sont == '0' (les zones vides).

IV. Le fichier main *« mac_gyver.py »*

Boucle principale :

Je limite ma boucle à 30 fps et ajoute des évènements pour les déplacements avec **event.type == KEYDOWN** et demande de générer la fonction de la class **Player** avec **MAC_GYVER.update** et d'afficher une nouvelle image pour chaque déplacement du personnage avec **screen.blit(MAC_GYVER.direction, (MAC_GYVER.x , MAC_GYVER.y)** *au départ quand j'ai créé ma boucle j'avais placé ma constante **MAC_GYVER** dedans et ne comprenais pas pourquoi mon personnage revenait systématiquement à sa position initiale j'ai passé beaucoup de temps à vérifier mon code pensant que j'avais fait une erreur dans sa rédaction. C'est en montrant le code à un ami dev que j'ai réellement compris le fonctionnement d'une boucle.*

Conditions pour les objets :

J'ai créé un booléen pour chaque objet qui est **True** au départ et devient **False** uniquement si le personnage se place dessus

« If (MAC_GYVER.x, MAC_GYVER .y) == (TUBE_POS.x, TUBE_POS.y):

TUBE_NOTPICKED = False »

J'ai rajouté une autre condition pour donner l'illusion d'un compteur **if TUBE_NOTPICKED == False:** alors affiche l'objet à une autre position **screen.blit(TUBE_POS.Loot_Image, (x, y))**

Conditions de victoire :

Quand MacGyver atteint la case du gardien (e) deux variables booléennes **False** au départ sont testées afin de déterminer si l'utilisateur gagne ou perd. Si tous les objets ont été ramassé **GAME_WON = True** il gagne et la fenêtre affiche du texte « YOU WIN », si l'utilisateur n'a pas ramassé tout les objets **GAME_LOOSE = True** il perd la fenêtre affiche « GAME OVER ».