Projet 3 : The hero of MacGyver !!!

Introduction projet :

Dans le parcours de développeur Python sur OpenClassroom, il nous est demandé dans le Projet 3 de développer un jeu de labyrinthe sur interface 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.

L'algorithme du labyrinthe :

a . Pour la création du code de mon labyrinthe, j'ai commencé par coder un labyrinthe en mode console. La question qui vient est pourquoi avoir choisir cette façon de coder ? La réponse est que : Le labyrinthe en mode console va me permettre en premier lieu de faire toutes les classes nécessaires au fonctionnement du labyrinthe que je détaillerais les difficultés que j'ai rencontrées plus en bas.

b . J'ai répartie le fonctionnement de mon labyrinthe en six fichiers :
- labyrinthe.py = Ce fichier contient toutes les fonctions principales de la class maze.
- MacGyver.py = La class Macgyver contient ses coordonnées(x,y).
- Guardien.py = Cette class est celle du guardien.
- Objet.py = Cette class est celle des objets.
- constante.py = Ce fichier contient toutes les images et la taille de la fenêtre au fonctionnement de l'interface du jeu.
- Interface.py = Ce fichier gère l'affichage et les déplacements du labyrinthe en mode pygame.

Explication des importations :

Ma manière d'importer les différents modules est très simple :
- import pygame = Consiste à importer toute la bibliothèque pygame dans mon script
- from macgyver import Macgyver = Ce script contient la class Macgyver, avec l'attribut self.coordinates qui contient les coordonnées (x,y) de Macgyver.
- from constante import constante = Ce script importe toutes les variables qui contiennent toutes les images des personnages, murs, objets du labyrinthe.
- from labyrinthe import Maze = Ce script labyrinthe, contient une class que j'ai nommée : class Maze. class Maze a 12 méthodes, chacun de ses 12 méthodes gère le labyrinthe avec leur rôle respectif.

Explication des class :

Les class que j'ai placer dans les différents scriptes sont :

- class Maze = Se trouve dans le fichier labyrinthe.py, cette class gère toutes les fonctionnalités du labyrinthe, notamment les images, le dessin du labyrinthe, la répartition des objets de façons aléatoire, les déplacements etc …

- class Macgyver = Dans cette class, ont lui affecte un attribut de class nommer 'self.cordinnates', cette attribut nous permettra de faire une relation d'appartenance avec l'attribut 'self.macgyver', qui

au final, deviendra self.macgyver.coordinnates. Cette de relation d'appartenance gère les coordonnées (x,y) de macgyver depuis le fichier labyrinthe.py.

- class InterMaze = Cette class appartient au fichier interface.py, elle a en elle différents attributs, variables, fonctions pygame pour gérer les affichages, les déplacements etc …

Les différentes fonctions de ma class Maze :

a . def parse_file = Cette fonction que j'ai créer est une grille fictive avec les coordonnées x et y que j'ai nommer case_x et case_y. Cette fonction me permet de gérer les différents coordonnées de Macgyver, du guardien, mur, chemins et des objets.

b . def get_keys = Elle me permet de récupérer les coordonnées de tous les chemins contenus dans le dictionnaire.

c . def random_coordinates = La fonction random_coordinates sert à choisir aléatoirements 3 coordonnées de chemins qui nous permettrons plus-tard de déplacer les objets aléatoirement dans le labyrinthe.

d . def get_items = Cette fonction récupère la fonction random_coordinates, pour renommer les 3 coordonnées des chemins en objet, c'est-à-dire : que les clés des trois coordonnées qui sont nommer avec le caractère c, mais dans la fonction get_items ses 3 coordonnées s'appelleront t, s, e qui représenteront les 3 objets.

e . def bottom, def top, def left, def right = Ses différentes fonctions me permettent de gérer les déplacements de Macgyver.

f . def check_move = Cette fonction est comme un feu tricolore, elle nous indique si on peut avancer, s'arreter, ramasser les objets et vérifie les collisions.

g . def parse_laby = Grâce à la fonction d'origine qui est self.labyrinthe qui a été initialisé dans la méthode def __init__ , la fonction parse_laby dessine un labyrinthe en mode console, en parcourant le dictionnaire à l'aide de la boucle for, ensuite avec le (if value == (ex : 'c' or 'd' or 's' or 't' or 'e' or 'm' , qui sont les différents), en suite juste en bas de la condition, on change la valeur du caractère, ex : comme ceci : value = caracter ou ont créer une variable qui contient une chaine de caractère, comme ça, line = '' '', et ensuite juste en bas de n'importe qu'elle condition citer ci-dessus, ont ajoutent ceci : line = caractère:).

h . def picture_maze = Elle dessine également le labyrinthe en mode pygame, grâce à différentes variables, boucles et instructions :

1. *Au début j'insère comme paramètre self (par défaut) et fenêtre, fenêtre est une variable qui vient du fichier interface.py, elle sert à afficher les cotées de l'interface, si nous la mettons comme paramètre dans cette fonction, c'est parce qu'elle nous permettra d'intégrer les images dans le dictionnaire.*
2. *Après je créer les variables qui nous permettront d'insérer les ressources d'images.*
3. *Je parcours le dictionnaire self.labyrinthe avec la boucle for key, value …., ensuite je créer deux variables coo_x et coo_y, à l'intérieur de ses variables je calcule à l'aide des clés key[0]*

et key[1] (l'abscysse et l'ordoonnée) avec la TAILLE_SPRITE, qui est la taille des caracteres du dictionnaire.

4. *Pour finir, à l'aide des conditions if value == caracter : et de leurs instructions, qui seront (fenetre.blit(coo_x, coo_y), le caracter se transformera en image.*

i . def move = Dans cette fonction, il y a quatre direction de déplacements, qui nous servent à déplacer Macgyver, si on regarde bien la structure de la fonction, on a fait un réfactorielle des directions, ce qui nous permettra en premier lieu de faire une vérification de chaque direction, pour vérifier si on est autoriser à avancer ou non, ensuite ont remplacent les coordonnées de Macgyver à un chemin, en fonction des directions ont incrémente pour pouvoir avancer ou ont décrémente ses coordonnées. Et pour finir ont redéfinir Macgyver à son caractère initiale (d).

Les détails de ma class InterMaze :

1. *Au début je fais une docstring pour les trois importations différentes, ensuite j'importe ce que j'ai besoin.*
2. *Je créer une class qui se nomme InterMaze, avec la méthode init. Dans la méthode init, je créer un objet qui contient la class Maze, ce qui me permettra de faire le réfactorielle des fonctions qui sont dans mon fichier labyrinthe.py à mon fichier interface.py.*
3. *Ne pas oublier d'initialiser l'interface, grâce à la fonction pygame.display.init(),l'ouverture de la fenètre à l'aide la fonction pygame.display.set_mode, je créer un intitulé avec la fonction pygame.display.set_caption, ensuite j'intègre le fond de mon image grâce à la fonction pygame.image.load, pour clore le tout je fais un chargement de la fenêtre pour afficher correctement les images du labyrinthe.*
4. *Je créer un booléen, pour la boucle du jeu, qui se nomme continuer = 1 (vrai).*
5. *J'insère une fonction grâce à la fonction pygame.time.*
6. *Je parcours les évènnements de pygame, à travers la boucle for, ensuite dans la boucle je pointe la première condition aux touches pygame, ce qui va me permettre d'activer les touches directionelles et de faire un réfactorielle de mes déplacements qui sont dans ma class Maze juste en bas des 4 directions, ex : if event.key == K_UP : m.move(''bottom'').*
7. *Avant la fin de la boucle, je remets le booléen (continuer = 0).*
8. *Pour finir, je recharge l'image du labyrinthe.*

Les détails du fichier constante.py :

1. Au début du script, j'écris la docstring des trois variables que j'ai créé :
   - La première variable elle sert à compter le nombre de cotée qu'à un caractère
   - La deuxième variable est la taille du caractère même.
   - La troisième variable contient les deux premières variables, qui seront calculer pour obtenir la taille finale des caractères du labyrinthe.
2. La quatrième variable, contient l'image du mur du labyrinthe.
3. La cinquième variable, contient l'image de MacGyver.
4. La sixième variable, contient l'image du gardien.
5. La septième variable, contient une seringue.
6. La huitième variable, contient un tube.
7. La neuvième variable, contient une aiguille.