PROJET 3 : Aidez Mac Gyver à s'échapper

Principes de l'exercice

Créer un labyrinthe 2D dans lequel Mac Gyver aurait été enfermé. La sortie du labyrinthe est protégée par un garde. Mac Gyver doit trouver dans le labyrinthe 3 objets qui lui permettront de confectionner une seringue avec laquelle il pourra endormir le garde et donc sortir du labyrinthe.
Si il n'a pas ramassé les 3 objets et qu'il se présente devant le garde, la sortie lui est refusée et le jeu est perdu.

Fonctionnalités de l'exercice

- Le labyrinthe ne possède qu'un seul niveau, qui doit être enregistré dans un fichier à part pour le modifier facilement.
- Mac Gyver est dirigé par les touches directionnelles du clavier.
- La fenêtre du jeu doit être un carré de 15 sprites de longueur.
- Mac Gyver se déplace de case en case.
- Mac Gyver récupère un objet en se plaçant dessus.
- Le programme s'arrête si Mac Gyver à trouvé tous les objets et la sortie du labyrinthe, sinon il meurt.
- Le programme est standalone

Choix de l'algorithme

Fichier « n1 » :

Dans ce fichier, je définis la structure de mon labyrinthe grâce à des lettres et des chiffres.
La lettre « d » correspond au point de départ de mon labyrinthe. La lettre « a » correspond à l'arrivée. La lettre « d » correspond au départ. Le chiffre « 0 » correspond aux chemins que peut emprunter Mac Gyver.

Fichier « constantes.py » :

Dans ce fichier, je définis les données nécessaires au bon fonctionnement des fichiers labyrinthe.py et classes.py. On y trouve la taille de ma fenêtre de jeu, en pixel la taille de la bannière en pixel, le nombre de case de chaque coté du labyrinthe. J'y définis également des lettres qui représenteront les objets que Mac Gyver doit trouver dans le labyrinthe. Enfin je crée des variables qui indiquent le le chemin des images nécessaire au jeu.
Ce fichier est appelé en tant que module dans les fichiers labyrinthe.py et classes.py.

Fichier « classes.py » :

Ce fichier contient les classes nécessaire au bon déroulement du jeu. J'y ai défini deux classes :

- **Maze**

Cette classe permet de définir la structure du labyrinthe. Elle possède deux méthodes :

- **Generate**

Cette méthode ouvre le fichier « n1 », lit les différents éléments du jeu ligne par ligne et caractères par caractères et enregistre sa structure dans une liste.
Ensuite cette méthode permet de générer les trois objets de façon aléatoire, grâce au module « randint » de la bibliothèque « random ». Ces objets sont représentés par des lettres définies dans le fichier constantes.py (« T » pour le tube, « E » pour l'ether, « N » pour l'aiguille). Les objets peuvent être générés à partir des listes représentant la structure du labyrinthe, uniquement quand l'élément de la liste est un « 0 » car il représente les chemins que Mac Gyver peut emprunter. Je limite à trois objets. Enfin la structure du labyrinthe est enregistrée dans l'attribut « structure_map ».

- **Display**

Cette méthode permet d'afficher le contenu du labyrinthe à l'écran. Elle va remplacer les éléments de la variable structure_map par leurs images correpondantes. Elle permet également de définir des coordonnées « x_sprite » et « y_sprite » aux éléments en fonction de leur ordre d'apparition dans la liste « structure_map ». Cette méthode affiche donc les murs, le gardien, et les trois objets dans la fenêtre de jeu.

- **Character**

Cette classe permet de définir les mouvements que doit faire Mac Gyver, tenir un inventaire des objets ramassés, et d'afficher cet inventaire. Elle possède quatre méthodes :

- **move**

Cette méthode permet de définir les déplacements que MacGyver doit et peut faire en fonction des choix de l'utilisateur et de la structure du labyrinthe

- **erase**

Cette méthode permet de remplacer l'image d'un objet par celle du chemin une fois que Mac Gyver est passé sur l'objet.

- **add_inventory**

Cette méthode permet d'incrémenter l'inventaire de Mac Gyver. Elle est appelée lorsque celui-ci trouve un objet.

- **display_inventory**

Cette méthode permet d'afficher le nombre d'objets que Mac Gyver doit encore trouver.

Fichier labyrinthe.py :

Au début du fichier, j'importe la bibliothèque Pygame, permettant l'affichage de la fenêtre, des images et le déplacement à l'aide des touches directionnelles du clavier.J'importe aussi les modules « constantes.py » et « classes.py ». J'initialise Pygame, défini la taille de la fenêtre ainsi que les variables qui permettront l'affichage des images du labyrinthe.
Je défini 3 booléens qui serviront à savoir quel objet a été attrapé et quel objet ne l'a pas été.
Ensuite mon fichier est composé de 3 boucles :

- MAIN_GAME : Boucle principale : c'est la boucle du programme, tant qu'elle tourne, le programme est en marche.

- CONTINUE_HOME : Dans cette boucle je charge les images principales du jeu. Tant qu'elle tourne, l'écran d'accueil est visible.Quand l'utilisateur passe l'accueil en appuyant sur entrée, on ferme la boucle d'accueil.On charge le labyrinthe et le personnage à l'aide des classes.

- CONTINUE_GAME : Tant qu'elle tourne, l'utilisateur joue et déplace Mac Gyver. Pour ramasser les objets, le programme va comparer la structure du labyrinthe avec la position de MacGyver. Si le déplacement se fait sur un objet, il appellera les méthodes erase et add_inventory et le booleen définissant si l'objet est attrapé devient True.
  Enfin si le joueur possède les 3 objets en se déplaçant sur le gardien alors il gagne, le programme s'arrête et j'affiche un écran qui lui signifie sa victoire sinon un écran lui signifie sa défaite et l'objet ou les objets qu'il n'a pas trouvé grâce aux booléens.

Difficultés rencontrées

Ma plus grande difficulté a été de comprendre le fonctionnement des classes, j'ai mis pris pas mal de temps à assimiler le concept et à pouvoir l'utiliser. J'ai repris les mêmes cours plusieurs fois.
J'ai aussi eu des difficultés à placer un bandeau en haut de la fenêtre et à y déplacer les images. J'ai testé plusieurs méthodes avant de trouver la bonne. Je me suis aidé de l'exercice « DK labyrinthe », et j'ai effectué beaucoup de recherches sur internet. Une autre difficulté a été de bien comprendre l'anglais notamment sur les sites où la plupart des conseils sont donnés en anglais. Enfin j'ai également eu du mal à faire en sorte que mon code corresponde aux convention PEP8.