P3 / Macgyver get out of the maze

Github : https://github.com/CoLoDot/Pygame.git

1/ Vue d'ensemble

Pitch : Le projet 3 est un jeu 2D d'un seul niveau dont le personnage principal est Macgyver. Le but de la partie est de ramasser les trois objets répartis dans le labyrinthe afin d'endormir le gardien qui bloque la sortie.

Le projet m'a permis de découvrir et d'apprendre le langage Python ainsi que son module Pygame. De fait, partant de zéro ou presque en programmation, j'ai d'abord passé deux semaines et demi a étudier par le biais des cours proposés dans le parcours de la formation. Suite à ces enseignements, mêlants théorie et travaux pratiques, j'ai commencé à développer le projet dans un seul script, en démarrant par la réalisation du labyrinthe. La réalisation du projet 3 a donc duré 1 mois et demi (cette durée inclue le temps passé à étudier les cours).

2/ Classes et fonctions

CLASSE MAZE

Le script mgclasses.py contient les trois classes du jeu. La première, « Maze » a pour but de créer et afficher à l'écran le labyrinthe dans lequel Macgyver est piégé. Pour ce faire, il a fallu créer un script de texte (level.txt) contenant le labyrinthe en version brute. A partir de là, c'est au sein de la fonction « creating_level » que la structure est appelée en lecture simple. Ceci nous permet donc de lire chaque ligne, puis chaque caractère, afin de générer le labyrinthe et d'en sauvegarder la structure. Enfin, la partie graphique du labyrinthe se fit par le biais du module Pygame. Cette dernière étape est contenue dans la fonction « displaying_level », qui reprend la structure du labyrinthe précédemment créée et converti chaque élément en sprite. Ainsi, les éléments du fichier « level.txt » correspondant à « m » sont couplés avec une image de mur tandis que ceux correspondant à « 0 » deviennent le sol sur lequel le personnage évolue.

CLASSE MACGYVER

A la suite de la classe Maze se trouve la classe Macgyver, qui contient deux fonctions. Dans la fonction « __init__ », l'image/le sprite du personnage est crée par le biais du module Pygame (« pygame.image.load().convert_alpha() ») et sa position est initialisée aux coordonnées « 0, 0 », donc dans le coin supérieur gauche de la fenêtre de jeu. La seconde fonction de la classe est celle qui consiste à faire bouger le personnage dans le labyrinthe. Il fut donc nécessaire de se baser sur la structure du niveau du jeu pour éviter que le personnage ne traverse les murs et qu'il suive le chemin prédéfinie par les sprites correspondants à « 0 » dans le script « level.txt ». Concrètement, c'est par le biais d'une condition « if » que chaque mouvement fonctionne. Dans un premier temps, on vérifie que le personnage ne puisse pas sortir de la fenêtre de jeu. Ensuite on s'assure que le sprite vers lequel le personnage se dirige n'est pas un mur, en lisant la structure du labyrinthe. Si le sprite n'est pas un mur, le personnage peut avancer.

CLASSE PICKITTOWIN

La dernière classe du jeu est celle permettant de créer et d'afficher les objets aléatoirement au sein du labyrinthe. Pour ce faire, le module random fut utilisé. Ainsi, la première étape fut donc de relire à nouveau la structure du niveau créée dans la classe Maze puis, étant donné que la structure est un array, d'aller lire chaque ligne (y) et d'y définir une position aléatoire pour chaque objet. Ensuite, une boucle « while » fut créée pour chaque item, chaque boucle à donc pour but de définir le sprite en x de manière aléatoire et dispose déjà de la ligne (y). Afin d'éviter toute collision, la boucle vérifie que le sprite correspond bien à « 0 » afin que l'objet n'apparaisse pas sur un mur. Si le sprite est correct (égal à « 0 »), le test devient la position de l'objet. Ainsi, à chaque nouvelle partie, une position aléatoire est définie pour chaque objet.

3/ Boucle principale du jeu

La boucle principale du jeu permet de coordonner l'affichage graphique et les évènements clavier effectués par le joueur. Par ce biais on constate que le jeu est bien interactif et ce de plusieurs façons. D'une part, les éléments du jeu nous permettent de constater que le projet s'inscrit dans la programmation orientée objet. D'autre part, l'usage des touches du clavier nous permet de créer une interaction entre le joueur (qui manipule Macgyver) et les items qu'il doit saisir au sein du labyrinthe. Par ailleurs, la boucle principale permet de maintenir la fenêtre de jeu active tant que la partie n'est pas gagnée (par l'usage d'un booléen lié aux items à saisir), mais elle regroupe également l'affichage de tous les éléments (musique, compteur d'objet, etc.)

4/ Difficulté(s) rencontrée(s)

La principale difficulté du projet fut de mettre en place la structure conditionnelle permettant l'affichage aléatoire des objets à ramasser par le personnage pour gagner le jeu. (classe Pickittowin)

5/ Environnement virtuel

L'usage d'un environnement virtuel est capital pour que le jeu puisse être utilisé sur n'importe quelle machine ou système d'exploitation. Ainsi, le fichier « requirements.txt » permet au joueur d'installer les libraires nécessaires, dans leurs bonnes versions, avant d'y jouer.

6/ PEP8

D'un point de vu plus formel, le code des différents scripts suit les recommandations PEP8, qui permettent à n'importe quel développeur ou développeuse de comprendre chacune des fonctionnalités écrites dans les scripts.

Dans le présent projet, le module Pylint a été utilisé pour noter et vérifier le code afin qu'il se rapproche le plus possible d'une forme compréhensible pour tous.

7/ Ajout

Dans le but de rendre le jeu plus dynamique et agréable, une musique de fond a été ajouté au jeu.