P3_01_lienGithub : [Julien-Ama/McGyverLabyGame: Create a 2d maze game (github.com)](https://github.com/Julien-Ama/McGyverLabyGame)
P3_02_présentation du projet

MAC GYVER et le LABYRINTHE

création d'un jeu en 2D en utilisant le langage Python et de la bibliothèque Pygame. Les fonctionnalités sont mises en œuvre en suivant les instructions du projet 3 de Openclassrooms.

Le projet est défini par plusieurs packages importables avec leur fichier "**__init__.py**" pour faciliter la gestion des modules:

Dans le dossier "**configuration**", nous trouverons la structure de la carte du jeu dans un fichier à part "**map.config**" afin que celui-ci soit facilement modifiable, ainsi que le fichier "**configuration**" permettant d'interagir avec la carte grâce aux éléments de base définis dans ce fichier.

Dans les dossiers suivants, une Class à était créer dans chaque fichier afin de faciliter les interactions et les importations.

Dans le dossier "**BackSide**" nous trouverons donc la class "**Character**" avec les caractéristiques du personnage ainsi que la class "**Map**", qui sera la jonction principale des caractéristiques de classes, créant ainsi une colonne vertébrale de l'ensemble de notre code:

"**def setCharacter**" permet de puiser dans la class "Character" (def setCoordinates) pour établir la position de notre personnage.

"**def addObjects**" représente la répartition de nos objets apparaissant aléatoirement sur les zones potentielles de notre carte dites "letterForSpace". Nous utiliserons ici la méthode **random** (import random).

"**def setMouvement**" représente les mouvements de notre personnage. Celui-ci puisera donc dans la class "Character" et adapte la capacité de mouvement grâce aux touches directionnelles du clavier que nous retrouverons dans l'import de Pygame dans la class "Visuel".

"**def doMoovement**" organise les conséquences et interactions des mouvements de notre personnage, si celui-ci rencontre un obstacle, ramasse un objet ou bien arrive à la sortie du labyrinthe étant bien ou non équipé.

"**def checkCoordinates**" représente le pourtour de notre carte afin que notre personnage ne puisse aller au-delà de notre structure.

Dans le dossier "**Visuel**" se trouve la partie graphique de notre ensemble:

en interaction avec notre class "Map", dans un premier temps nous verrons la création de notre **surface** avec une résolution de 750 sur 800. "750" pour établir avec justesse la répartition de nos sprites de 15x15 (15 x **50** = 750). Nous apercevons également notre stock de couleurs et l'adaptation de nos images puisants dans notre dossier "**ressource**". Ainsi, notre palette d'images et de couleurs sont prêtes à être utilisées. (ici, la méthode **pygame.transform.scale** est utilisée pour ajuster la taille de nos images).

"**def update**" adapte pygame dans notre "def doMoovement" que nous avons dans notre class "Map" à l'échelle grossie par 50.

"def **addAllObjectInWindowPygame**" celui-ci adaptera l'ensemble de nos éléments sur notre surface. (objets, personnage, murs...) à l'échelle x50. La méthode **rect** et **blind** sera utilisée pour adapter nos images.

"**def lancement**" est la boucle maintenant notre jeu actif jusqu'à que l'une des conditions lui ordonne de se fermer. "def setMouvement" de la classe "Map" est intégré dedans. Le personnage peut donc être déplacé par les touches directionnelles du clavier tant que certaines conditions n'ont pas été remplies.

"**def Victory, Defeat**" représente les fin alternatives de notre héro si celui ci a ramassé tous les objets ou non avant d'aborder notre gardien. " **def zero, one, two** and **max**" représente l'inventaire de notre personnage suivant les objets ramassés. (Nous trouverons la méthode **pygame.font.Sysfont** pour adapter du texte sur notre surface).

Dans notre dossier "**Engine**" se trouve le moteur de notre jeu, celui-ci regroupe tous les éléments majeurs "Map" et "Visuel" afin que notre fichier "**main**" puisse facilement activer cet ensemble.

notre dossier "**Venv**" pour la Création de l'environnement virtuel.

utilisation de "**Flake8**" pour l'optimisation de notre code.

utilisation de "**cx_Freeze**" pour avoir une version standalone de notre jeu.

Julien Inglèse