

# OC - Projet 3 : Aidez MacGyver à s'échapper !

(création d'un jeu de labyrinthe en 2D)

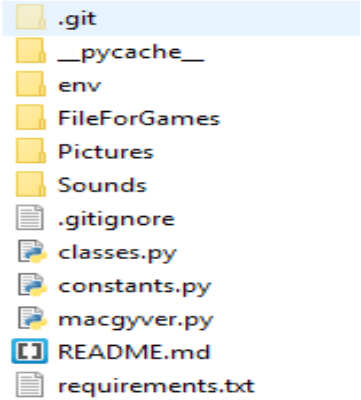
Lien vers Github: <https://github.com/Dyanne06/MacGyver>

## Introduction:

Le projet consiste à créer un jeu de labyrinthe en 2D en utilisant le module pygame.

**But du jeu :** un personnage (ici Mac Gyver) est enfermé dans un labyrinthe en 2D. Pour en sortir, il doit réunir une aiguille, un tube en plastique et de l'éther (dispersés dans le labyrinthe) afin d'endormir le garde (ici Murdoc) qui garde la sortie, sinon il meurt.

## Structure du programme :

	<p>Mon projet se compose de 3 fichiers .py :</p> <ul style="list-style-type: none"><li>• macgyver.py (à exécuter pour lancer le jeu),</li><li>• classes.py (rassemble toutes les classes définies dans le jeu)</li><li>• constantes.py (rassemble toutes les constantes du programme utilisées dans les 2 autres fichiers).</li></ul> <p>Les images sont stockées dans le répertoire /Pictures</p> <p>Les « fichiers sons », au format .wav, sont dans le répertoire /Sounds</p> <p>Le fichier fourni pour construire le labyrinthe sera mis dans le répertoire FileForGame, il sera facilement modifiable.</p>
--	---

## Mise en place du programme :

On utilisera le module pygame pour dessiner l'interface graphique et utiliser la gestion des événements suite à l'utilisation des touches directionnelles du clavier par le joueur.

Dans un premier temps, j'ai créé 2 classes :

1. La classe Labyrinthe : elle permet de créer l'objet labyrinthe, de l'afficher, et de le modifier (comme par exemple, y rajouter les objets permettant à Mac Gyver de sortir)

- Le constructeur `__init__(self, laby_file)` prend en paramètre le fichier fourni pour créer le labyrinthe.

Ce fichier contient des caractères représentant le départ, l'emplacement des murs et l'arrivée (dans notre exemple 'm' = mur, 'a' = arrivée, 'd' = départ).

Cette représentation est stockée dans un dictionnaire `CHAR_LABY`.

- Un objet de type Labyrinthe aura 3 attributs :
  - `start` (coordonnée de la case « départ » ou « d » ici)
  - `struct[]` : une liste de listes représentant le labyrinthe (Chaque ligne sera représentée par une liste, et toutes les listes générées seront stockées dans une liste). Ainsi pour accéder à un élément du labyrinthe, on fera `struct[N°ligne][N°colonne]`
  - `struct_OK[]` : une liste représentant uniquement les emplacements où Mac Gyver peut aller.

- La méthode `show(board_game)`: prend en paramètre en objet du module `pygame` (type `surface`) (`pygame.display.set_mode((BOARD_SIZE + BORDER, BOARD_SIZE)` défini dans les constantes) et nous permet d'afficher le labyrinthe sur cet objet.
  - La méthode `modify_struct(coord, c_char)`, qui prend en paramètre des coordonnées et un caractère, permet de modifier le labyrinthe.
  - La méthode `add_kit_survey()`: permet de rajouter les éléments nécessaires à Mac Gyver pour sortir du labyrinthe (j'appellerai cela le Kit de survie) ?
2. Création de la classe `Perso`: elle va nous permettre de créer l'objet représentant le personnage Mac Gyver, de le mouvoir et de lui faire attraper les objets du kit.
- Le constructeur `__init__(col,row)` prend en paramètre sa position dans le labyrinthe.
  - Une instance `Perso` aura entre autres, l'attribut `KIT` qui permettra de comptabiliser les éléments ramassés par Mac Gyver.
  - La méthode `move(direction, struct)` : permet de déplacer le personnage et d'attraper un objet s'il tombe dessus
  - la méthode `catch_obj(struct)`: permet de ramasser un objet (on le rajoute dans le kit du personnage) et un son est émis.

Remarque : dans cette version de projet, le personnage `Murdoc` correspond simplement à la case « Arrivée » dans la structure du labyrinthe. Mais on pourra bien-sûr faire évoluer ce personnage en le créant comme objet de la classe `Perso`.

Puis, j'ai construit l'algorithme principal :

Une boucle d'événement est créée : pour laisser le programme ouvert tant que le joueur n'a pas fini la partie ou tant qu'il ne quitte pas volontairement le jeu.

Puis on parcourt la liste des événements, pour chaque événement créé, Mac Gyver va dans la direction souhaitée sauf si c'est sur un mur (d'où la liste des emplacements autorisés sauvegardée dans `struct_OK`) ou en dehors du cadre.

A chaque fois que Mac Gyver ramasse un objet, ce dernier sera affiché dans la bordure à gauche du labyrinthe.

Un son final est exécuté lorsqu' `MacGyver` arrive sur la case « `Murdoc` » (son différent s'il gagne ou s'il perd)

Un petit sprite « `You Win` » sera affiché lorsque `Mac Gyver` gagne et « `You lose` » lorsqu'il perd.

Le programme s'arrête automatiquement lorsque `Mac Gyver` est arrivé à la case « `Murdoc` ».

J'ai rajouté une temporisation de 5 seconde avant la fermeture du jeu.

### **Difficultés rencontrées au cours de ce projet :**

Tout d'abord la prise en main de tous les outils et modules fournis par python (`pygame`, `pylint`, `venv`) mis à notre disposition n'est pas évidente. De nombreuses recherches sur internet ont été salvatrices (sans parler de [translate.google.fr](http://translate.google.fr/)!)

Ensuite, le concept orienté objet est une notion importante qu'il faut arriver à maîtriser. Pour y arriver, j'ai suivi les cours proposés par `OpenClassrooms` et j'ai complété par d'autres informations trouvées sur internet.

Pour finir, la finalisation du code en respectant les bonnes pratiques de la `PEP 8`, notamment la prise en compte des messages de `pylint` a été ardue.

Les solutions pour contrer les difficultés rencontrées ont donc été apportées par mes recherches sur les sites informatiques spécialisés ([stackoverflow.com](http://stackoverflow.com), [pylint.readthedocs.io](http://pylint.readthedocs.io), [www.python.org](http://www.python.org), <http://www.pygame.org/docs/>, etc...)