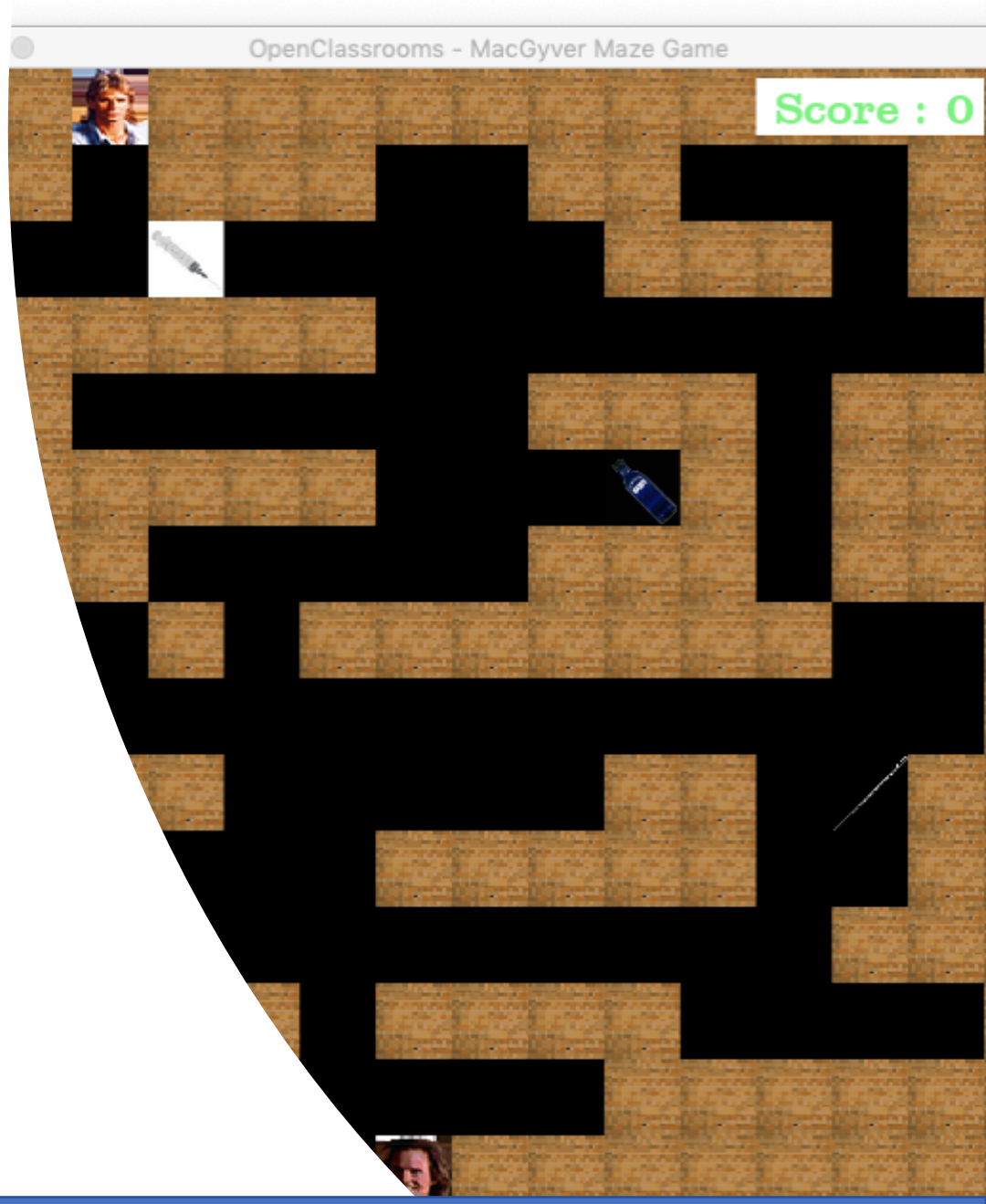
Fabrice Jaouën
Parcours DA Python
Projet 3 OC – Labyrinthe
https://github.com/Fabrice-64/Project3_OC.git



Cahier des Charges

Exigence	Couverte : OK - NOK
Structure labyrinthe dans un fichier	OK
McGyver contrôlé par les touches directionnelles	OK
Objets répartis aléatoirement à chaque ouverture	OK
Fenêtre de jeu, carrée sur 15 sprites	OK
Objet récupéré en passant dessus	OK
Jeu s'arrête si McGyver a récupéré tous les objets et trouvé sortie,	OK
Sinon McGyver meurt	NOK: McGyver ne meurt pas, mais est fait prisonnier (cliffhanger)
Programme en standalone	OK

Rôle des différents modules

Module	Action
maze_game.py	Pilote le déroulement du jeu, depuis l'initialisation du labyrinthe, jusqu'à la rencontre entre le héros et le gardien
Maze.py	Permet l'initialisation du labyrinthe et la disposition de tous les éléments constitutifs du jeu
Items.py	Contient les éléments nécessaires à l'initialisation des personnages et accessoires, mais aussi les déplacements du héros avec leur effet sur les autres objets, exception faite des murs.
Config.py	Rassemble les ressources graphiques et constantes, pour les mettre à jour plus facilement dans l'hypothèse d'une évolution du jeu

Séquences du jeu depuis maze_game.py

```
# Creates an instance to open a file containing the maze and bu
maze = Maze()
maze.draw_maze(Config.maze_level)
maze.display_objects()

# Updates the maze_window
pygame.display.flip()

# Keeps the window open until intentionally closed
running = True
while running:
    # This loop goes through the 3 main phases of this game: mo
    maze.macgyver.move(maze.walls, maze.window)
    maze.macgyver.collecting_item(
        maze.window, maze.objects_to_collect, maze.my_font)
    maze.macgyver.meeting_warden(
        maze.warden, maze.window, maze.my_font_end_game)
    pygame.display.flip()
```

Initialisation du labyrinthe
Et de l'ensemble des objets

Actualisation de la fenêtre

Séquences de jeu:
1. Déplacement du héros;
2. Contrôle de présence d'un item à collecter
3. Contrôle d'une rencontre avec le gardien

Focus : Initialisation du labyrinthe
Démarche générale :

1. L'objet Labyrinthe est initialisé avec une fenêtre vide, mais renseignée (titre et tableau de score);

2. Dessin du labyrinthe, avec les murs, le héros et le gardien ;
   - Choix: le héros et le gardien sont dans le fichier décrivant le labyrinthe pour éviter que le hasard ne les mette trop proches l'un de l'autre ;
   - Les coordonnées des éléments de mur servent de clé au dictionnaire les rassemblant pour un contrôle plus fluide des capacités de mouvement du héros;

3. Répartition aléatoire des objets à collecter

Focus : Initialisation du labyrinthe

```
class Maze:
    def __init__(self):
        # Creates lists containing the objects of each wall sprite, the coo
        # Including a list containing the objects to be collected by the he
        self.corridors = []
        self.walls = {}
        self.objects_to_collect = []
        # Set up a window for the maze. Length and height are 15 characters
        self.window = pygame.display.set_mode((MAZE_HEIGHT, MAZE_WIDTH))
        pygame.display.set_caption("OpenClassrooms - MacGyver Maze Game")
        self.my_font = pygame.font.Font(SELECTED_FONT, 24)
        self.my_font_end_game = pygame.font.Font(SELECTED_FONT, 48)
        self.text = "Score : "
        self.text_window = self.my_font.render(
            self.text, True, (125, 250, 125))
```

Choix de conception :

Dès son initialisation, le labyrinthe reçoit les attributs nécessaires au déroulement du jeu:

self.corridors : utilisé pour répartir aléatoirement les objets au lancement du jeu ;

self.walls: utilisé pour identifier si le prochain mouvement du héros va dans un mur

self.objects_to_collect: utilisé pour vérifier si un objet est à proximité immédiate du héros

Dès l'initialisation, la fenêtre de jeu, vide, est créée avec les éléments statiques

Focus : Répartition des objets à collecter

```
number_items = len(Config.objects_to_be_collected_pictures)
low = 1
high = len(self.corridors) // (number_items + 2)
# Scatters the objects on the corridors of the maze by slicing the li
# and randomly laying an object in this very slice
for i in range(number_items):
    # Displays the objects in the maze and stores them in a class lis
    location = randrange(low, high)
    # The coordinates of the corridors have been saved as tuples as (
    self.object_to_collect = Items.ToCollect(
        self.corridors[location][0], self.corridors[location][1])
    self.object_to_collect.picture = self.draw_picture(
```

Choix de conception :

- Une liste à usage unique & contenant les x, y des cases vides a été créée lors du dessin du labyrinthe ;
- Cette liste est divisée en segments, dans chacun d'entre eux, un tuple est choisi aléatoirement pour fournir les coordonnées d'un des objets à collecter ;
- Le nombre de segments est supérieur au nombre d'objets pour éviter qu'ils ne soient trop près du gardien ;
- Ces objets, augmentés de leurs coordonnées, sont ensuite placés dans une liste où le héros viendra vérifier à chaque déplacement s'il est près (ou pas de l'un de ces objets à collecter)

Création des items du labyrinthe

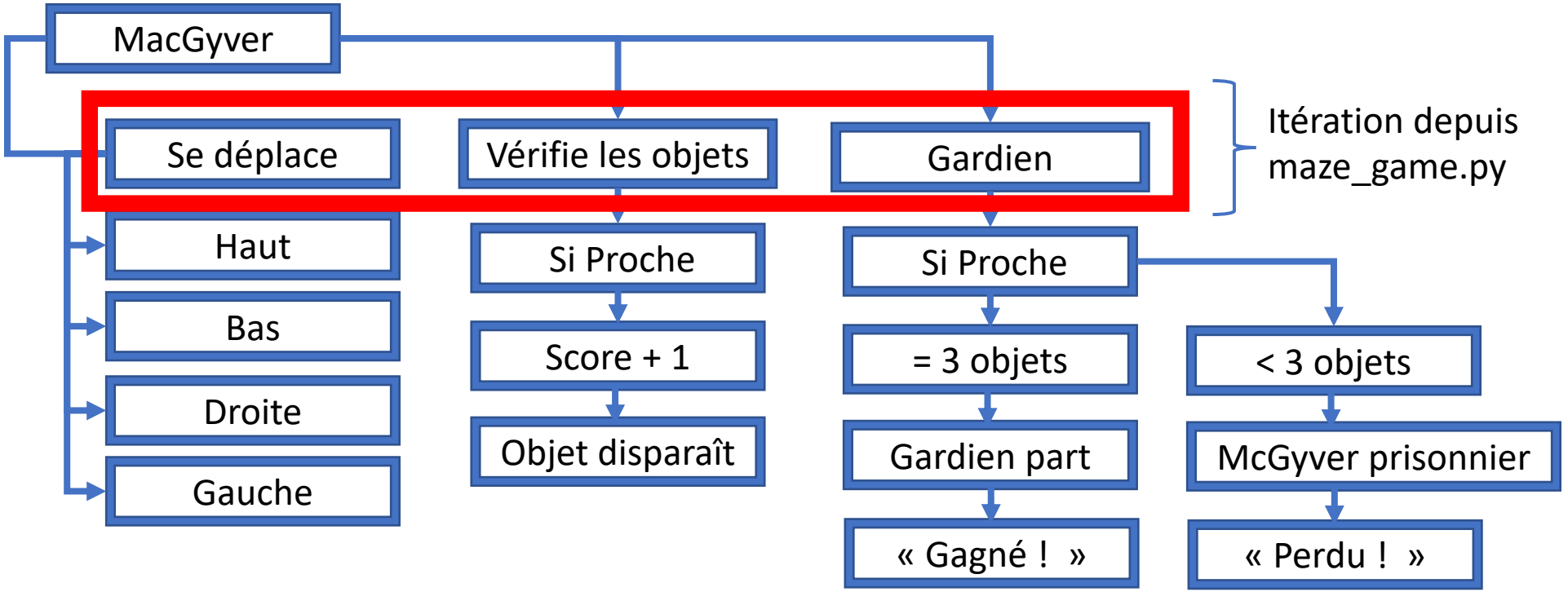
```
class Items:
    def __init__(self, x, y):
        self.x = x
        self.y = y


class Warden(Items):
    def __init__(self, x, y):
        super().__init__(x, y)
        self.pic = Config.warden_picture
```

Choix de conception :

- Création d'une classe parent, Items et de classes enfants qui héritent des coordonnées x, y d'Items. But: si jamais le jeu doit évoluer, il sera plus facile d'attribuer d'autres caractéristiques soit à l'ensemble des objets, soit individuellement ;
- Chaque classe enfant est initialisée avec son icône. But : clarifier la structure de chaque objet, alors que cela aurait pu aussi être fait en direct à la création du labyrinthe.

Classe MacGyver

Il s'agit de la classe la plus riche en fonctionnalités, puisqu'il est le seul à se déplacer, mais aussi à avoir une action directe sur les autres objets.



Evolutions possibles :

- Ajouter une fonctionnalité de création aléatoire de labyrinthes ;
- Rendre le gardien mobile ;
- Faire apparaître le couteau suisse de MacGyver dans sa cellule au bout d'un certain nombre de tentatives d'évasion ;
- Dans les trois objets figure un tuyau en plastique, donc, faire apparaître un cercle autour de MacGyver pour simuler la portée de la sarbacane ;
- Donner un temps limité à MacGyver pour remplir la mission ;

Connaissances acquises :

- Fluidité dans l'usage simple de Git et GitHub ;
- Apprivoisement de VS Code ;
- POO : classes, méthodes, attributs de méthode et de classe, interaction entre modules grâce au self ;
- Intérêt du module Config et du fichier Requirements.txt ;
- Utilisation du main() pour dérouler les grandes étapes d'un programme et s'appuyer sur les modules pour aller dans leur détail.
- Logique informatique : découpage du cahier des charges en multiples petits pas ;