Développement d'Applications Python – Projet 3
« Aidez MacGyver à s'échapper ! »

1. Reformulation succinte du sujet

L'objectif de ce projet est de concevoir un jeu de labyrinthe en deux dimensions avec le langage Python en utilisant le paquet PyGame. La structure du labyrinthe doit être enregistrée dans un fichier annexe au programme, et elle doit pouvoir être modifiée facilement par l'utilisateur. Le labyrinthe est un carré de 15 cases par côté (i.e. 15^2 = 225 cases).

2. Réflexions de démarrage de projet

2.1. Représentation du labyrinthe

Concernant la représentation du labyrinthe dans le code source, je pense à deux possibilités :
- La plus naturelle est de constituer un tableau à deux dimensions (de type liste de listes par exemple) dont les éléments seraient les éléments constituant le labyrinthe (murs, outils, personnages), rangés dans l'ordre des coordonnées de la carte de jeu
- Une deuxième possibilité consiste à représenter le labyrinthe à l'aide d'un dictionnaire dont :
    - les clés seraient les éléments constituant le labyrinthe (murs, objets, personnages)
    - les valeurs associées seraient les coordonnées de leurs emplacements (simples ou multiples) dans le labyrinthe

2.2. Edition et sauvegarde de la structure du labyrinthe (carte modifiable)

Concernant la sauvegarde de la structure du labyrinthe (i.e. les murs et les personnages, car on ne stockera probablement pas les outils destinés à être ramassés, car ils devront apparaître de façon aléatoire), mon mentor m'incite dès le début du projet à mettre en œuvre une solution plus élaborée qu'un « simple » document texte dans lequel on viendrait lire les caractères.

2.3. Implémentation de la solution technique

La solution devra être mise en œuvre avec le paquet PyGame. Je consulte la documentation :
- PyGame est en quelque sorte une alliance entre la librairie SDL (Simple DirectMedia Layer) et le langage Python
- PyGame comporte plusieurs modules permettant respectivement de gérer les images, l'affichage, le son, le clavier, la souris, le temps, les événements, etc.

2.4. Choix de la structure de base du projet

Il m'est nécessaire de concilier les trois points précédents pour réaliser ce projet.
J'ai précédemment réalisé un programme en langage C avec la librairie SDL sur un sujet proche. J'ai été confronté à des difficultés lorsque j'ai voulu travailler avec une représentation du labyrinthe avec un dictionnaire tel que décrit dans la seconde partie du paragraphe 2.1. Le fait de travailler avec un tableau à deux dimensions m'avait permis de réaliser l'exercice beaucoup plus simplement.

Il me semble que l'énoncé du projet « Aidez MacGyver à s'échapper ! » est assez similaire, donc j'oriente mon choix vers un **labyrinthe représenté par un tableau à deux dimensions**, à l'aide d'objets de type pandas.DataFrame.

De plus, il me semble important de raisonner de façon « orientée client » ou « orientée utilisateur ». Ainsi, j'opte pour une **édition de la structure du labyrinthe de manière graphique dans une fenêtre, à l'aide de la souris**, avec un fichier de sauvegarde au format CSV qui est adapté à des manipulations de tableaux de type pandas.DataFrame.
Mon objectif est de mutualiser la structure de l'interface graphique pour le jeu et l'édition de la carte.

3. Structure du projet en classes

N'ayant pas d'expérience dans la Programmation Orientée Objet (POO), je structure initialement mon application avec deux classes : Labyrinth (pour la logique du jeu) et Interface (pour les graphismes). Ces deux classes contiennent alors beaucoup (trop !) d'attributs et de méthodes.
Mon mentor me réoriente rapidement vers une structure plus « éclatée », avec une classe « Player » et une classe « Tool » pour alimenter la classe « Labyrinth ».
Cette étape remet certes en cause le travail de plusieurs jours, mais me permet surtout de mieux comprendre la logique et l'intérêt de la POO : l'approche est en effet très différente de la programmation fonctionnelle. De plus, en retravaillant la structure de mon application, je comprends que travailler avec davantage de classes facilite la lisibilité et simplifie le code.
Le diagramme de classes structurant mon application est présenté en figure 1 :

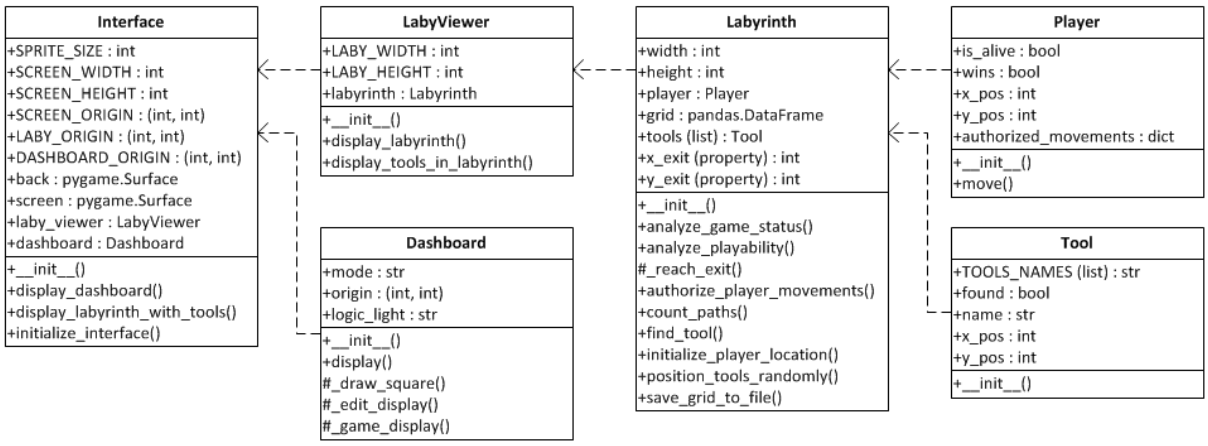

Figure 1 : diagramme de classes de l'application « Mac Gyverinth – by etienne86 »

4. Programmation du projet avec le langage Python

Ayant investi un temps assez important dans la réflexion initiale, puis dans la restructuration de l'application en classes, la programmation avec le langage Python devient plus aisée. Cela n'empêche pas de rencontrer des problèmes, notamment lorsque le jeu ne se comporte pas comme attendu.
Voici un exemple rencontré dans ce projet : le jeu devrait se terminer, mais il est encore possible de déplacer le joueur ! Pour déboguer, j'utilise la fonction « print » pour afficher la valeur de certaines variables à des moments-clés du code. Cela me permet notamment de constater qu'abscisses et ordonnées sont inversées, ou que certaines valeurs ne sont pas de l'ordre de grandeur attendu !

5. Lien vers le code source du projet

https://github.com/etienne86/oc_p3_mac_laby