Mini projet : Blinky Vs Pac Man

Pac-Man, personnage emblématique de l'histoire du jeu vidéo, est un personnage en forme de rond jaune doté d'une bouche. Il doit manger des pac-gommes et des bonus dans un labyrinthe hanté par quatre fantômes. Chaque fantôme a un comportement qui lui est propre :

- **Blinky** attaque directement Pac Man. Il suit Pac-Man comme son ombre.
- **Pinky** a tendance à se mettre en embuscade. Elle vise l'endroit où va se trouver Pac-Man.
- **Inky** est capricieux. De temps en temps, il part dans la direction opposée à Pac-Man.
- **Clyde** feint l'indifférence. De temps en temps, il choisit une direction au hasard (qui peut être celle de Pac-Man).

1. Objectif du projet

L'objectif du projet est de programmer le comportement de Blinky du jeu Pac Man. Il s'agit donc de trouver un chemin reliant deux cases d'un labyrinthe. Ces deux cases représentent celles occupées par Blinky et PacMan

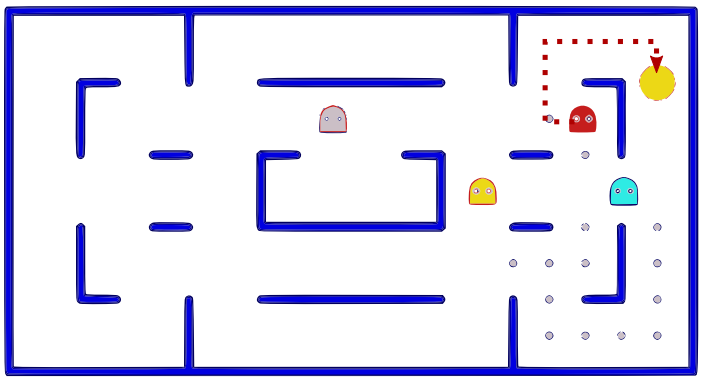

Figure 1: Exemple de chemin reliant les deux personnages

2.Organisation

Le projet sera réalisé par équipe de 2 élèves avec 1 seul poste informatique. Chaque élève alternera le rôle de pilote et de copilote de programmation. Le pilote est chargé de programmer sur ordinateur et le copilote doit relire et corriger en direct les fautes.

3.Travail attendu

- ✗ Imaginer une ou des solutions et réaliser tout ou partie de la solution choisie ;
- ✗ Présenter à l'oral la solution choisie (

4.Documents fournis

Génération du labyrinthe

Un labyrinthe est une surface connexe. De telles surfaces peuvent avoir des topologies différentes : simple, ou comportant des anneaux ou des îlots.

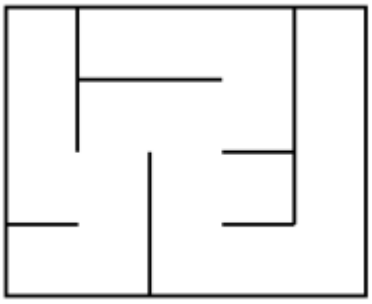

Figure 2: Labyrinthe parfait

On peut distinguer deux catégories de labyrinthe :

- les labyrinthes « parfaits » (figure 2) : chaque cellule est reliée à toutes les autres et, ce, de manière unique.

- les labyrinthes « **imparfaits** » (tous les labyrinthes qui ne sont pas parfaits) : ils peuvent donc contenir des *boucles*, des *îlots* ou des *cellules inaccessibles*.

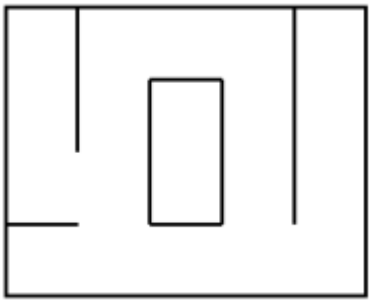

Figure 3: Labyrinthe imparfait

La classe Labyrinthe fournie dans labyrinthe.py est une classe permettant de générer et afficher des labyrinthes parfaits.

Le labyrinthe généré est représenté par un graphe, dont les sommets représentent des **cellules** (chaîne de caractères comportant les coordonnées 'ligne_colonne') et les arêtes des **passages** entre deux cellules (l'absence d'arête équivaut alors à un **mur**).

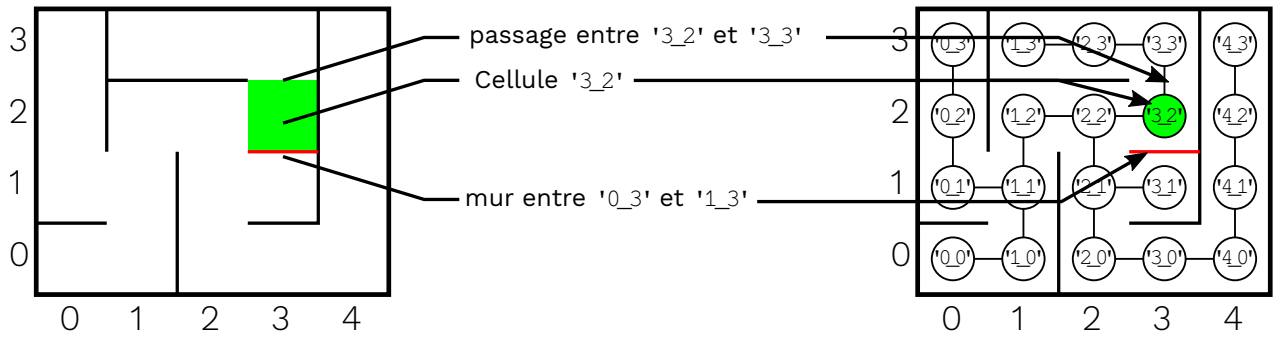

Figure 4: Exemple de labyrinthe et sa représentation avec un graphe

Le graphe est modélisé par une liste de successeur stockée dans un dictionnaire. Chaque nœud peut posséder au plus quatre arrêtes. Ces arêtes sont nommées en suivant le codage suivant :

Nom de l'arête	Localisation sur le nœud
'N'	Arête nord située en haut du nœud
'S'	Arête sud située en bas du nœud
'E'	Arête est située à droite du nœud
'O'	Arête est située à gauche du nœud

Les voisins d'un nœud du dictionnaire principal sont stockés dans un dictionnaire secondaire comportant 4 clés aux noms des différentes arêtes et 4 valeurs correspondant aux noms des voisins (None pour un mur). Par exemple la liste de successeur du graphe figure 4 est la suivante :

```
>>> laby = Labyrinthe(4,3)
>>> laby.creer()
>>> laby.lire_graphe()
{'0_0': {'N': None, 'S': None, 'E': '1_0', 'O': None}, '1_0': {'N': '1_1',
'S': None, 'E': None, 'O': '0_0'}, '2_0': {'N': '2_1', 'S': None, 'E': '3_0',
'O': None},  … , '2_1': {'N': '2_2', 'S': '2_0', 'E': '3_1', 'O': None},
'3_2': {'N': '3_3', 'S': None, 'E': None, 'O': '2_2'}}
```

5. Mots clé d'aide à la recherche de solutions :

- ✗ Parcours en profondeur d'un graphe utilisant une pile
- ✗ Algorithme de Lee / Parcours en largeur d'un graphe utilisant une file
- ✗ Parcours main gauche