M1102: Introduction à l'algorithmique et à la programmation
Projet n°1
Le Cluedo de l'IUT'O

1 Présentation

L'objectif de ce projet est de programmer un jeu inspiré du classique *Cluedo* `https://fr.wikipedia.org/wiki/Cluedo`. A l'origine, le but de ce jeu est de découvrir qui a assassiné le Docteur Lenoir, avec quelle arme et dans quelle pièce. Dans la version que vous allez programmer le sénario est le suivant :

Figure 1 – Le jeu du Cluedo

Les fantômes des pionners de l'informatique hantent le bâtiment informatique de l'IUT'O. Ces fantômes sont bienveillants et souhaitent aider les étudiants de l'IUT'O à obtenir leur diplôme. Pour ce faire, l'un d'entre eux va donner un cours à minuit la nuit prochaine sur l'une des matières informatiques dans l'une des salles du département. Les joueurs auront donc à découvrir la matière, le professeur et le lieu du cours.
Les professeurs sont Edgar Codd (fondateur des bases de données relationnelles), John von Neuman (concepteur des premiers ordinateurs), Allan Turing (fondateur des principes mathématiques à la base de l'informatique), JCR Licklider (fondateur du réseau Internet), Ada Lovelace (conceptrice du premier programme informatique), Grace Hopper (conceptrice des langage Fortran et Cobol). Les matières sont les Systèmes informatiques, l'Algorithmique-Programmation, les Bases de données, la Conception Orientée Objet, les Interfaces Homme-Machine et l'Architecture des réseaux. Les salles sont les différentes pièces du bâtiment Informatique.
Le jeu se compose d'un plateau représentant le plan du bâtiment (ici le département Informatique), un jeu de cartes représentant chacun des personnages, chacune des matières et chacune des salles du bâtiments sauf une, de deux dés et de 6 pions représentant les joueurs de la partie (entre 2 et 6). Au début du jeu, on choisit au hasard, une carte personnage, une carte matière et une carte salle qui seront cachées dans la pièce particulière (pour nous ce sera la salle des archives). Ces trois cartes forment l'énigme à découvrir (appelé plus tard mystère). Ensuite le reste des cartes est distribué à l'ensemble des joueurs et chaque joueur place son pion sur son point de départ. Chaque tour de jeu consiste pour un joueur à lancer les dés. Ce lancé lui permet de savoir de combien de cases il peut se déplacer. Les couloirs du bâtiment sont cadrillés en cases. Un joueur peut entrer ou sortir d'une pièce uniquement par les portes prévues à cet effet. Le déplacement d'un joueur se fait de la manière suivante :

— s'il est dans une pièce, il choisit une porte pour en sortir (cette étape lui coûte 1),
— une fois dans le couloir, il se déplace d'au plus le nombre de points obtenus sur les dés, il ne peut pas passer d'une case à l'autre en diagonale,
— s'il atteint une porte, il peut entrer dans la pièce (cette action lui coûte 1)

Il y a un cas particulier au cas où la pièce possède un passage secret. Dans ce cas le joueur peut choisir (au lieu de lancer les dés), d'accéder directement à la pièce accessible via le passage secret.

Si à la fin de son déplacement un joueur se trouve dans une pièce, il peut émettre une hypothèse en indiquant, le professeur et la matière qu'il pense être dans le mystère. La salle est obligatoirement la pièce où il se trouve. Une fois son hypothèse émise, le joueur qui se trouve après lui doit vérifier s'il possède au moins l'une des trois cartes. Si c'est le cas, il en choisit une et la montre au joueur qui a émis l'hypothèse. S'il n'en a pas il le dit et c'est le joueur d'après qui vérifie ses cartes, ainsi de suite jusqu'à ce qu'un joueur ait montré une carte ou qu'aucun joueur ne possède de carte de l'hypothèse émise. Dans le cas où le joueur termine dans un couloir, il ne peut pas émettre d'hypothèse. Les joueurs jouent chacun leur tour, suivant ce principe. Lorsqu'un joueur pense avoir la solution, il doit se rendre dans la pièce particulière (celle qui n'a pas de carte correspondante), et quand il y est, il peut donner sa solution (professeur, matière et salle). Il vérifie si sa solution est correcte en regardant les cartes cachées au début. S'il a raison, il a gagné, sinon il est éliminé, ce qui signifie qu'il ne peut plus jouer mais il doit quand même continuer à répondre aux hypothèses de ses adversaires.

La partie se termine donc lorsqu'un joueur a gagné ou lorsque tous les joueurs sont éliminés. Pour s'aider dans leur tâche, les joueurs tiennent à jour une *fiche indices* qui va leur permettre de noter les informations qu'ils ont récoltées au cours de la partie.

L'objectif de ce projet est donc d'implémenter le jeu du Cluedo où un joueur est un être humain et les autres sont joués par l'ordinateur.

2 Travail à faire

Vous devrez développer, par groupe de trois ou quatre personnes, un programme qui permet à des étudiants de l'IUT (entre 2 et 6) de découvrir quel professeur enseignera quelle matière dans quelle salle. La structuration du projet vous est imposée. Elle s'organise en plusieurs grandes parties qui vous permettront de proche en proche d'arriver à l'implémentation finale du jeu. Pour chaque structure de données, vous êtes en grande partie libres de la représentation mémoire de cette structure. Par contre l'interface de programmation vous est imposée. Si vous respectez bien les spécifications, vous pourrez utiliser les scripts permettant de jouer en mode texte et en mode graphique au Clued'IUT'O.

Nous allons décrire les différentes structures de données que vous aurez à implémenter dans les fichiers Python qui vous sont fournis. Les spécifications plus précises de chaque fonction sont données en commentaires dans les fichiers Python.

Le schéma figure 2 donne une vue de l'architecture générale de l'application.

Dans cette architecture on peut distinguer deux grandes parties. La première partie concerne la gestion du jeu de cartes associé au jeu de cluedo. Cette partie comprend les structures de données suivantes.

— `carte.py` qui contient les informations concernant une carte
— `jeucarte.py` qui gère l'ensemble des cartes du jeu
— `mystere.py` qui gère les cartes qui ont été choisies au hasard et qu'il faut deviner pour gagner

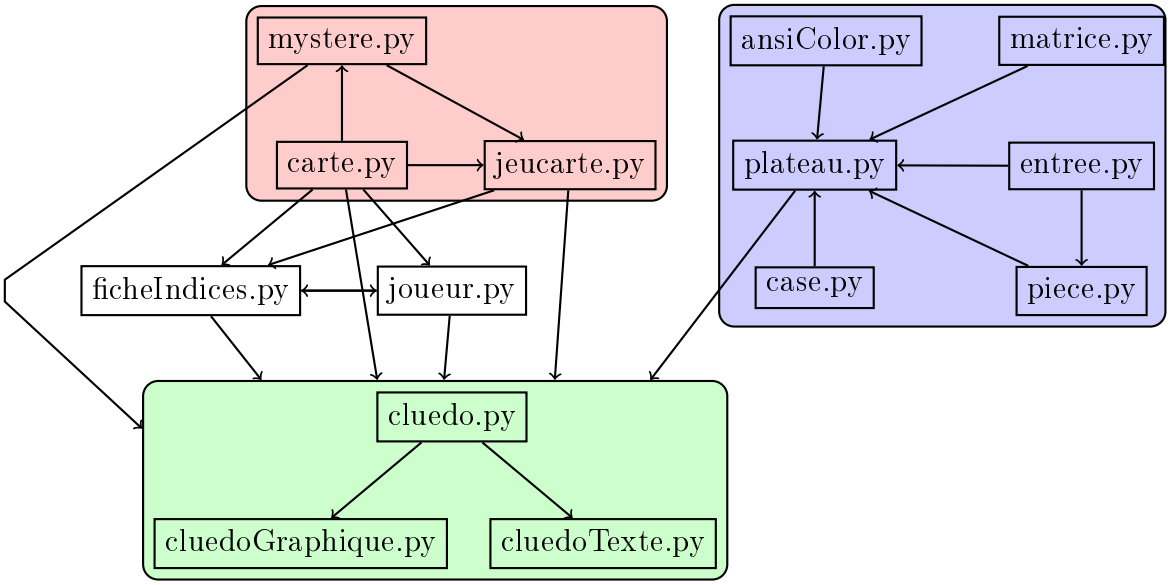

FIGURE 2 – Architecture de l'application

La seconde concerne la gestion du plateau de jeu, elle comprend les structures de données suivantes :

- `ansiColor.py` qui n'est pas à implémenter et qui gère un code des couleurs en mode texte
- `case.py` qui représente une case du plateau
- `entree.py` qui représente une entrée de pièce
- `piece.py` qui représente les pièces sur le plateau
- `matrice.py` qui est la structure de données permettant de représenter une matrice en 2D
- `plateau.py` qui représente le plateau de jeu principalement constitué d'une matrice de cases.

Deux structures de données se basent sur les jeux de cartes :

- `ficheIndices.py` qui gère une fiche indices d'un joueur. Cette fiche a besoin de connaitre le jeu de cartes ainsi que la liste des joueurs pour fonctionner
- `joueur.py` qui gère un joueur de la partie. Ce joueur aura des cartes en main ainsi qu'une fiche indices qui lui permettra de résoudre le mystère.

Enfin la structure, `cluedo.py` gère l'ensemble du jeu du Cluedo et utilise donc le plateau, les joueurs et le jeu de cartes.
Les deux scripts `cluedoTexte.py` et `cluedoGraphique.py` gèrent les interactions avec l'utilisateur pour jouer au Cluedo.
Voici quelques détails sur chacune des structures de données

2.1 La gestion du jeu de cartes

2.1.1 carte.py

Cette structure de données représente une carte du jeu. Les cartes sont réparties en trois catégories : les *professeurs*, les *matières* et les *salles*. Chaque carte possède donc une catégorie, un numéro, un nom et une description. Par exemple, la carte représentant les *bases de données* est de la catégorie *matière* et porte le numéro 2 dans cette catégorie. Son nom est *bases de données* et sa desciption est *Stockage et interrogation des données*. Notez bien que le numéro de la carte est plutôt un numéro d'ordre dans une catégorie. Pour identifier une carte il faut sa

catégorie et son numéro. Contrairement au jeu réel du Cluedo, on a une carte particulière non distribuable qui est la pièce où l'on peut émettre son hypothèse pour tenter de gagner. Par conséquent, il faudra un indicateur pour distinguer cette carte (qui est non distribuable) des autres cartes. Dans le fichier `carte.py` trois constantes permettent d'identifier les numéros de catégorie `PROFESSEUR`, `MATIERE` et `SALLE` et le dictionnaire `nomCategorie` permet de retrouver le nom des catégories.
Les fonctions à implémenter sont décrites dans le fichier.

2.1.2 mystere.py

Cette structure gère le mystère à découvrir, c'est-à-dire les trois cartes cachées au début de la partie. Etant donné qu'un mystère est composé d'une carte de chacune des catégories, le mystère va conserver uniquement le numéro de chacune des trois cartes. Les fonctions sont décrites dans le fichier.

2.1.3 jeucarte.py

Cette structure permet de gérer un jeu de cartes, de choisir un mystère, de récuperer les cartes distribuables, de mélanger le jeu de cartes, etc.
Le constructeur de cette structure de données fabrique un jeu de carte vide. La fonction `lireJeuCarte()` permet de recupérer un jeu de cartes qui se trouve dans un fichier texte comme `Data/cartes.csv`. Cette fonction nécessite l'implémentation de la fonction `addCarte()` qui ajoute une carte à un jeu de carte. La fonction `cartesDistribuables()` permet de construire un jeu de cartes ne contenant que les cartes distribuables du jeu passé en paramètre. La fonction `listeCartes()` permet de transformer le jeu de cartes en une liste de cartes. Cette fonctionnalité sera utilisée dans la fiche indice. Les autres fonctions sont décrites dans le fichier.

2.2 La gestion du plateau

2.2.1 ansiColor.py

Vous n'avez rien à implémenter dans ce fichier. Il sert simplement à produit un affichage en couleur du plateau.

2.2.2 case.py

Une case est l'élément de base du plateau. Une case peut être à `None` signifiant que cet emplacement n'est pas accessible. Si la case n'est pas à `None` elle peut faire partie d'un des trois catégories suivantes :

- un couloir (catégorie 0),
- une pièce (catégorie >0) dans ce cas la catégorie est le numéro de la pièce à laquelle appartient la case,
- un point de départ de joueur (catégorie<0) dans ce cas la catégorie est l'opposé du numéro du joueur.

Par ailleurs une case peut contenir ou non le pion d'un joueur. Si elle contient un pion, il faut être capable de donner le numéro du joueur à qui appartient ce pion. Le reste des fonctions est décrit dans le fichier.

2.2.3 matrice.py

C'est simplement la structure de matrice vue en TD.

2.2.4 entree.py

Cette structure gère les portes d'entrée des pièces. En effet pour chaque entrée de pièce, nous avons besoin de connaitre son emplacement (lig,col) sur le plateau ainsi que la direction par laquelle on peut entrer dans la pièce. Cette structure n'implémente qu'un constructeur et les accesseurs aux trois informations de l'entrée.

2.2.5 piece.py

Cette structure gère les pièces (ou salles) du plateau. Les pièces s'étalent sur plusieurs cases du plateau et ont des caractéristiques qui leur sont propres comme un certain nombre de portes d'entrée et éventuellement un passage secret. De plus une pièce peut contenir plusieurs pions de joueur.
Par conséquent une pièce doit avoir un numéro (qui est le même que celui qui se trouve sur les cartes), une liste d'entrées, un contenu qui est une liste de pions (c-à-d des entiers correspondant aux numéros de joueur) et une information sur son passage secret éventuel. S'il y a un passage secret celui-ci indique le numéro de la pièce destination du passage et on doit connaitre la position du passage sur le plateau.
Les fonctions de cette structure de données permettent de créer une pièce (cela se fera grâce à la fonction de lecture d'un plateau), de consulter ces informations mais aussi d'ajouter et d'enlever des pions de la pièce. Par exemple, on peut retrouver la liste des entrées d'une pièce pour connaitre les points de départ des déplacements d'un joueur se trouvant dans cette pièce.

2.2.6 plateau.py

Cette structure gère le plateau de jeu. Il s'agit d'une matrice de cases avec une structure qui gère la liste des pièces présentes sur le plateau.
Beaucoup des fonctions de cette structure sont de simples appels au fonctions correspondantes de `matrice.py`.
La construction du plateau se fait à partir d'un fichier texte comme `Data/plan1.csv`. Cette fonction a besoin que les structures de données `piece.py` et `case.py` soient implémentées ainsi que les fonction `setCase()` et `mettrePion()`. L'affichage d'un plateau se fait par la fonction `afficherPlateau()` qui vous est donnée.
Voici quelques éléments sur certaines fonctions. Concernant `mettrePion()` et `enleverPion()` il faut gérer le cas où la case considérée appartient à une pièce et donc mettre à jour la pièce correspondante.
Les fonctions `passageXXX()` permet de savoir si il y a la possibilité de se diriger dans la direction `XXX` quand on se trouve sur une certaine case. Ces fonctions vont servir pour rechercher les déplacements possibles pour un joueur.
Un certain nombre de fonctions servent à calculer les cases accessibles à un joueur en fonction des points qu'il a obtenus aux dés. Ces fonctions doivent s'inspirer de ce que vous aurez fait en TD sur le labyrinthe. Il faut bien tenir compte du fait qu'un joueur qui est dans une pièce peut en sortir par n'importe quelle porte de cette pièce. De même, il faut prendre en compte que si un joueur entre dans une pièce par l'une des portes, son déplacement s'arrête là.

2.3 Les joueurs et les fiches indices

2.3.1 joueur.py

Cette structure de données gère les joueurs qui participent à la partie de Cluedo. Notez que contrairement à la version classique du jeu, les joueurs ne sont pas des personnages du jeu.

Un joueur possède un numéro (entre 1 et 6), un nom. Il peut être humain ou non (dans notre cas au plus un joueur sera un être humain). De plus, le joueur possèdera des cartes du jeu de cartes et pourra être éliminé ou non. On rappelle que si un joueur est éliminé, il ne joue plus mais il doit continuer à répondre aux questions des autres joueurs.
Le constructeur de joueur permet de créer un joueur dont le jeu de cartes est vide. La fonction `lireJoueurs()` permet de charger une liste de joueurs (sans leurs cartes) à partir d'un fichier texte comme `Data/joueurs.csv`.
Les fonctions sur les joueurs sont décrites dans le fichier `joueur.py`. La fonction `reponseHypothese()` est utilisée par les joueurs non humains pour répondre à une hypothèse émise par un autre joueur. Cette fonction peut utiliser de l'aléatoire si vous le souhaitez.

2.3.2 ficheindices.py

Cette structure gère les fiches indices des joueurs. Un exemple de fiche indices est donné figure 3.

```
                                Fiche Indices
Professeurs                                        Matières
                                  1 2 3 4                                          1 2 3 4
                    1. Edgar Codd: - - ? ?            1. Systèmes informatiques: - - + -
                2. John von Neuman: + - - -        2. Algorithmique-Programmation: - ? ? ?
                   3. Allan Turing: + - - -                    3. Base de données: - ? ? ?
                  4. JCR Licklider: - ? ? ?          4. Conception Orientée Objet: - ? ? ?
                   5. Ada Lovelace: + - - -           5. Interface Homme-Machine: - ? ? ?
                   6. Grace Hopper: - ? ? ?           6. Architecture des réseaux: - ? ? ?
Salles
                                  1 2 3 4
                  1. Amphithéâtre: - - ? ?
                      2. Salle TP: - + - -
                      3. Salle TD: - ? ? ?
              4. Bureau Technicien: + - - -
  5. Bureau du Chef de Département: - ? ? ?
                6. Salle des profs: - ? ? ?
                  7. Local INFASSO: - ? ? ?
                     8. Secrétariat: - ? ? ?
 9. Bureau du directeur des études: + - - -
```

FIGURE 3 – Exemple d'une fiche indices

Dans cette fiche on voit qu'il y a 4 joueurs (numérotés de 1 à 4). Pour chaque carte du jeu on a une série de signes `-`, `+`, `?` indiquant ce que l'on a appris depuis le début de la partie. `+` indique que l'on sait que le joueur correspondant possède la carte, `-` indique que l'on sait que le joueur correspondant ne possède pas la carte, enfin `?` indique que l'on ne sait pas si le joueur correspondant possède ou non la carte. Ainsi sur la figure 3 on sait que les joueurs 1 et 2 ne possèdent par la carte *Edgar Codd* mais on ne sait pas si 3 et 4 la possèdent. On voit aussi que le joueur 2 possède la carte *Salle TP* et donc aucun autre joueur ne la possède.
Pour gérer une telle fiche, il faut connaitre la liste des cartes distribuables et la liste des joueurs qui sont les paramètres du constructeur. La structure interne de la fiche indices doit associer à chaque carte du jeu de cartes, une liste de symboles (`-`, `+`, `?`) permettant de noter les indications connues sur la carte. Notez que les symboles sont désignés par des variables globales appelées respectivement `NEPOSSEDEPAS`, `POSSEDE` et `NESAISPAS`.

Le constructeur doit initialiser la structure interne de manière à indiquer qu'on ne sait rien.
La fonction `creerIndications()` permet de créér un dictionnaire associant à chaque numéro de joueur un `?` et peut être utilisée dans le constructeur.
La fonction `connaissance()` permet de mettre à jour la fiche indices en fonction d'une indication faite lors du jeu. Cette indication peut être

- le joueur `numJoueur` possède la carte `nomCarte`,`numCat` : dans ce cas on peut mettre un `+` pour le joueur `numJoueur` dans les indications de la carte. Evidemment il faut aussi mettre des `-` pour les autres joueurs.
- le joueur `numJoueur` ne possède pas la carte `nomCarte`,`numCat` : dans ce cas on peut mettre un `-` pour le joueur `numJoueur` dans les indications de la carte.

La fonction `initFicheIndices()` permet d'initialiser la fiche indice d'un joueur en fonction des cartes qu'il possède en indiquant que le joueur sait qu'il possède les cartes de son jeu.
Les fonctions suivantes sont utilisées pour créer l'IA des joueurs non humains. En effet pour permettre d'émettre des hypothèses pas trop stupides, il faut utiliser les connaissances acquises dans cette fiche.
La fonction `estConnue()` permet de savoir si on a une certitude sur une carte, c'est-à-dire qu'on sait qui la possède ou alors on sait que personne ne la possède.
La fonction `estDansLeMystere()` permet de savoir si on a la certitude que personne ne possède la carte (et donc elle fait partie des cartes cachées au début de la partie).
La fonction `listeCartesInconnues()` permet de récupérer la liste des cartes d'une certaine catégorie pour lesquelles on n'a pas de certitude.
La fonction `listeCartesJoueur()` permet de récupérer la liste des cartes d'une certaine catégorie pour lesquelles on est sûr qu'elles appartiennent à un certain joueur.
La fonction `hypothesesSures()` retourne un dictionnaire représentant les cartes dont on est sûr qu'elle font partie du mystère. Si on n'a pas de certitude pour une catégorie, celle ci n'apparaitra pas dans les clés du dictionnaire résultat.
La fonction `choixCartes()` retourne la liste de cartes d'une certaine catégorie intéressantes à utiliser pour émettre une hypothèse. En général, on choisit des cartes parmi celles dont on ne sait rien et éventuellement parmi celle que l'on possède soi-même. Vous pouvez mettre la stratégie que vous souhaitez dans cette fonction.
La fonction `creerUneHypothese()` crée une hypothèse sachant que la salle est imposée. Elle va donc choisir une carte parmi les professeurs et une parmi les matières. Ces cartes sont choisies en utilisant la fonction `choixCartes()`. Là encore vous pouvez utiliser votre propre stratégie pour choisir les cartes.

2.4 Le Cluedo

Cette partie est constituée de trois fichiers `cluedo.py`, `cluedoTexte.py`, `cluedoGraphique.py`. Vous n'aurez rien à implémenter dans les fichiers `cluedoTexte.py` et `cluedoGraphique.py`.

2.4.1 cluedo.py

Ce fichier implémente la strucuture du jeu du Cluedo avec sont plateau, ses cartes et ses joueurs. L'API contient un certain nombres d'observateurs qui permettent aux scripts `cluedoGraphique.py` et `cluedoTexte.py` d'afficher les informations de Cluedo mais aussi de demander des actions comme déplacer le joueur courant en fonction de l'interaction avec l'utilisateur. Par conséquent, le fichier `cluedo.py` ne comprend pas l'implémentation du déroulement d'une partie mais par contre il contient toutes les fonctions qui permettent de mettre à jour le plateau et les joueurs en fonction des choix fait par les joueurs. Ce fichier contient aussi les fonctions qui permettent à l'ordinateur de prendre des décisions (vers quelle

pièce il veut se déplacer, quelle hypothèse il émet etc...). Le détail des fonctions est donné dans le fichier.

3 Rendus

Les groupes de projets doivent être constitués de **trois ou quatre personnes du même groupe de TD**. Les groupes de projets seront formés par l'équipe enseignante.
Le rendu se fera en trois phases.

1. **le 10 décembre** l'implantation des fichiers gérant le jeu de cartes (voir section 2.1).
2. **le 21 décembre** l'implantation des fichiers `joueur.py` et `ficheindices.py` jusqu'au commentaire indiquant la fin des fonctions à implémenter.
3. **le 17 janvier** l'implantation complète du projet.

Le rendu final se fera sur l'ENT dans une archive `zip` portant le nom des développeurs du programme.
Ce rendu comprendra

— Les sources commentés de vos programmes
— Un petit dossier de programmation au format `pdf` indiquant votre répartition du travail, les méthodes de tests adoptées, le choix des structures de données utilisées, les principaux algorithmes que vous avez implémentés, les bugs connus de votre programme et les extensions que vous avez apportées. En annexe vous devrez insérer pour chaque participant une fiche individuelle indiquant le travail effectué au jour le jour.

Ce rendu se fera sur Celene le **17 janvier 2019** avant 23h55. Le **18 janvier** sera organisée une soutenance de 15 minutes au cours de laquelle vous présenterez votre travail et ferez une petite démonstration.