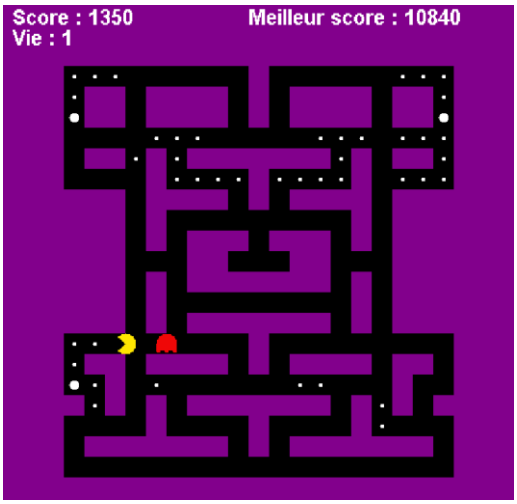
Projet Intelligence Artificielle

L'objectif de ce projet est de créer une intelligence artificielle capable de joueur à une variante du jeu de Pac-Man.

I) Règle du jeu



Vous jouez Pac-Man et votre objectif est d'obtenir le plus grand score possible en mangeant les gommes de la carte sans vous faire dévorer par les fantômes qui peuplent la carte. Chaque fois que le Pac-Man mange une gomme, il obtient 10 points. Certaines gommes, appelées super-gommes, ont le pouvoir d'effrayer les fantômes pendant un certain laps de temps (leurs couleurs virent au bleu). Lorsqu'un fantôme est effrayé, il peut être mangé par Pac-Man. Cela n'a pas pour effet de le tuer réellement, car il retourne simplement à sa position de départ en perdant son statut effrayé. Néanmoins, cela rapporte à Pac-Man un nombre important de points (100 points par fantôme mangé). Le Pac-Man peut se déplacer en haut, en bas, à gauche ou à droite sur la carte, mais ne peut pas traverser les murs. Si le joueur demande un déplacement dans la direction d'un mur, le Pac-Man n'effectuera pas de mouvement.

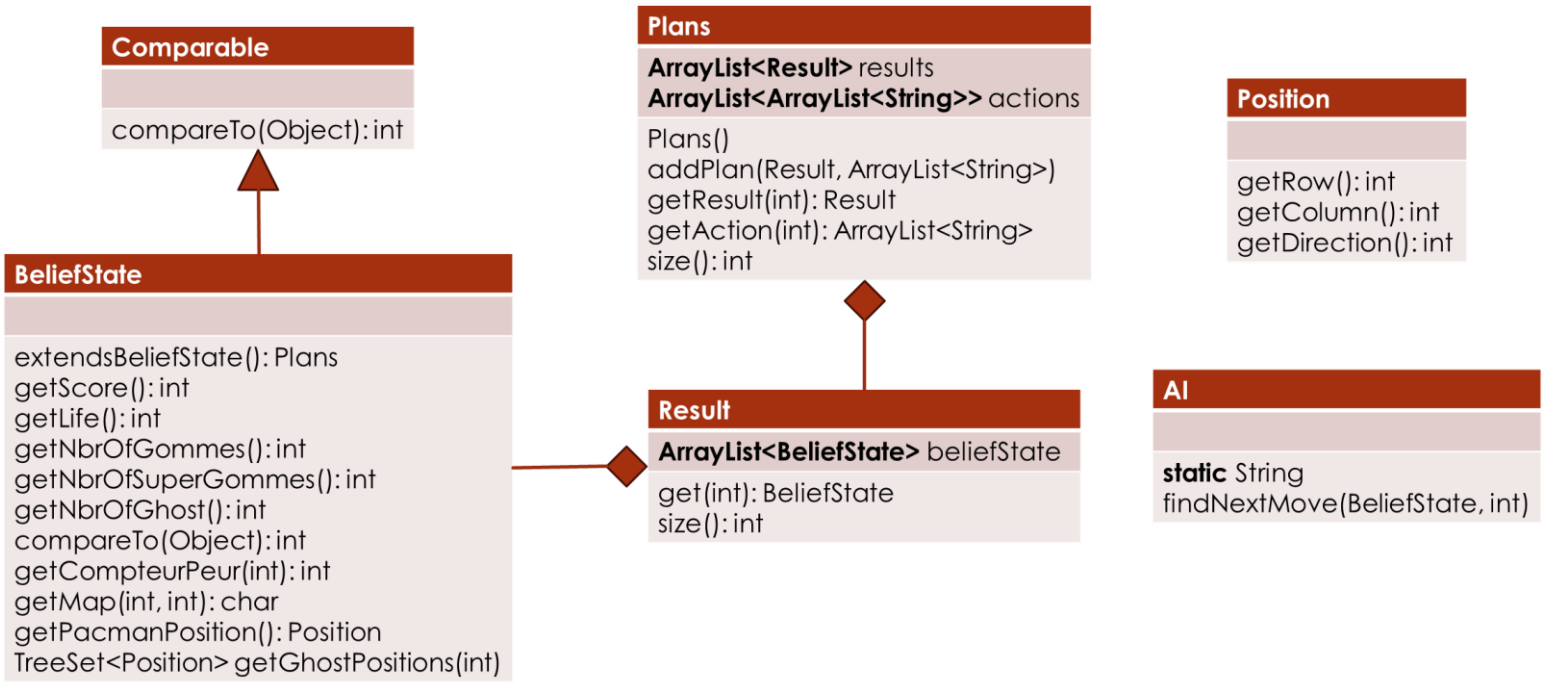
La spécificité de cette version du jeu de Pac-Man est que les fantômes ne sont visibles que lorsque le Pac-Man les a dans son champ de vision. Autrement dit, s'il y a un mur entre le Pac-Man et le fantôme, le Pac-Man ne verra pas le fantôme. Cette version est donc légèrement plus difficile que la version standard car il n'est pas toujours possible de connaitre à l'avance la position des fantômes pour les contourner. Il s'agit d'un environnement partiellement observable.

II) Moteur du jeu et affichage

J'ai utilisé le code de Roland Guillaume et Rémi Freret (et je les en remercie) pour le jeu et l'affichage graphique du jeu. J'ai dû néanmoins beaucoup modifier le code pour l'adapter aux nouvelles règles.

Le code source se trouve sur mon site web (sur la même page que le sujet du projet). Vous allez devoir compléter ce code afin d'implémenter l'IA du Pac-Man.

Voici le diagramme UML simplifié des classes du projet que vous allez utiliser:



a) Classe BeliefState

La classe *BeliefState* modélise un ensemble de croyance qui regroupe les états dans lequel peut se trouver le jeu à un moment donné. Il s'agit d'un ensemble d'état car les fantômes peuvent ne pas être visibles, et leurs positions dans ce cas-là sont inférées de leurs dernières positions connues et de leurs déplacements possibles. Tous les calculs de positions sont effectués par les objets de cette classe et vous n'aurez pas à les effectuer.

Les cinq fonctions suivantes seront utiles pour évaluer un état du jeu. La fonction *getScore* renvoie le score obtenu par Pac-Man dans cet état. La fonction *getLife* retourne le nombre de vie restante au Pac-Man. La fonction *getNbrOfGommes* renvoie le nombre de gommes qui n'ont pas encore été mangées sur le plateau. Le plateau est terminé lorsque cette valeur passe à 0. Notez que les super gommes sont également des gommes et sont comptées par cette fonction. La fonction *getNbrOfSuperGommes* renvoie le nombre de super gommes qui n'ont pas encore été mangées sur le plateau. La fonction *getNbrOfGhost* renvoie le nombre de fantômes sur le plateau. Notez que ce nombre ne change pas du début à la fin de la partie, car lorsqu'un fantôme est mangé il réapparait.

La fonction *compareTo* permet de comparer deux ensembles de croyance. Elle renvoie 0 si les deux états de croayance sont égaux, et une autre valeur si ce n'est pas le cas. Notez que la classe *BeliefState* implémente l'interface *Comparable*, ce qui veut dire que les objets de cette classe peuvent être utilisés par des structures de données ordonnées comme des *TreeSet* ou des *TreeMap* (utiles pour se souvenir des états de croyance déjà explorés et de leurs associer des valeurs).

On reviendra plus tard sur la fonction *extendsBeliefState*. Les autres fonctions sont expliquées dans l'annexe.

b) Classe Result

Il s'agit d'une classe (de type adaptateur) qui va gérer une liste d'état de croyance. Cette classe sert à stocker un ensemble d'état de croyance qui résultent d'une action effectuée par le Pac-Man à partir d'un état de croyance donné. Le résultat d'une action à partir d'un état de croayance peut donner plusieurs états de croyance car il faut séparer les états qui ne correspondent pas à un même percept. A titre d'exemple, imaginons qu'après un déplacement, le Pac-Man soit dans une position qui lui permet de voir certaines des positions possibles d'un fantôme. Dans ce cas-là, ces différentes positions ne pourront pas appartenir à un même état de croyance, car le Pac-Man ne pourra pas croire qu'un fantôme se trouve à une position visible s'il ne l'est pas, ou il ne pourra pas croire qu'il est à une autre position s'il le voit. Un objet de la classe *Result* correspond en fait au résultat de la fonction *RESULT* du slide 13 du cours 8, qui contient un état de croyance par percept possible. Cette classe ne donne pas une description des percepts car ils ne vous seront pas utiles pour calculer une stratégie (les fonctions de la classe *BeliefState* seront bien suffisantes).

La méthode *get* permet d'accéder à un état de croyance à partir de son indice, et la méthode *size* revoie le nombre d'états de croyance. Ces deux méthodes seront suffisantes pour naviguer dans la liste, et vous n'aurez jamais à construire ce type d'objets, qui seront générés par d'autres méthodes (en particulier la méthode *extendsBeliefState*).

c) Classe Plans

Il s'agit d'une classe qui permet de représenter les résultats possibles d'un ensemble d'actions effectuées à partir d'un même état de croyance. Ce type d'objet va contenir plusieurs objets de la classe *Result*, chacun associé à une liste de chaines de caractères (classe *String*) correspondant à des actions possibles (« UP », « DOWN », « LEFT » et « RIGHT »). Comme on a vu précédemment, les objets de la classe *Result* permettent de modéliser des ensembles d'états de croyances, chaque état de croyance correspondent à un percept possible du Pac-Man. Cela correspond à un résultat possible de la fonction RESULT du slide 13 du cours 18. Un objet de la classe *Plans* permet de stocker le résultat de la fonction RESULT du slide 13 du cours 18 pour différentes actions. En particulier, la méthode *extendsBeliefState* de la classe *BeliefState* va renvoyer un objet de la classe *Plans* qui contient le résultat de chaque action possible de l'agent à partir d'un état de croyance donné.

Les objets de cette classe pourront éventuellement être utilisés pour stocker des stratégies trouvées. Le constructeur permet de créer un objet vide, qui pourra être rempli à l'aide de la fonction *addPlan*, qui va ajouter un ensemble d'états de croyance (objet de la classe *Result*) associé à un ensemble d'actions qui donne ce même résultat (objet de la classe *ArrayList<String>*). Les trois autres fonctions permettent d'accéder au contenu d'un objet de type *Plans* (en particulier celui renvoyé par la méthode *extendsBeliefState* de la classe *BeliefState*). La fonction *getResult* renvoie l'objet de type *Result* dont l'indice est donné en paramètre. La fonction *getAction* renvoie l'ensemble des actions associées au *Result* dont l'indice est donné en paramètre. Enfin, la fonction *size* renvoie le nombre de *Result* stockés. Notez qu'il devrait y avoir au plus quatre *Result* stockés, correspondant aux 4 actions possibles qui sont de monter, descendre, aller à gauche et aller à droite. Lorsque le Pac-Man se trouvera dans un couloir (ou seules deux directions sont possibles), les deux actions qui mènent à un mur correspondront à un même résultat pour le Pac-Man (qui ne bougera pas) et seront stockées ensemble dans le plan (la liste de chaine de caractères associée à cette action contiendra deux actions comme par exemple {UP,DOWN} ou {LEFT,RIGHT}).

d) Classe AI

Il s'agit de la classe qui implémente l'intelligence artificielle qui va jouer au Pac-Man. Elle contient une unique fonction *findNextMove* qui va calculer la prochaine action à effectuer à partir de l'état de croyance courant donné en paramètre. Cette fonction retourne une chaine de caractères représentant l'action à effectuer (« UP », « DOWN », « LEFT » ou « RIGHT »). Il s'agit de la fonction que vous devez implémenter.

III) Travail à effectuer

Comme indiqué plus haut, vous devez compléter les méthodes *findNextMove* de la classe *AI*. Pour cela, vous pouvez créer autant de classes que vous le voulez. En revanche, vous ne devez pas modifier les classes du projet déjà existantes.

L'objectif est de faire une IA la plus efficace possible (c'est-à-dire qui obtient le plus grand score possible), mais également la plus rapide possible. Le temps moyen mis pas l'IA pour trouver l'action suivante sera affiché à la fin de jeu. Ce paramètre sera pris en compte dans la note.

Seul le fichier AI.class sera à rendre. Les fichiers rendus passeront par un logiciel de détection automatique de plagiat. Tout plagiat avéré sera sanctionné d'une note de zéro pour l'ensemble des groupes ayant soumis un projet commun.

IV) Annexe

Je décris ici des fonctions de la classes *BeliefState* qui pourraient vous être utiles si vous voulez faire des heuristiques extrêmement sophistiqués, mais elles ne sont pas indispensables.

La fonction *getCompteurPeur* retourne le compteur de peur d'un des fantômes (dont l'indice est donné en paramètre), qui est une valeur entre 0 et 60. Lorsque la valeur vaut 0, le fantôme est dans un état normal et ne peut être mangé. A l'inverse lorsque le compteur est supérieur à 0, le fantôme est dans un état de peur et peut être mangé. Lorsque le Pac-Man mange une super-gomme, la valeur de tous les compteurs de peur prend la valeur 60. Lorsque le fantôme est en état de peur, ce compteur est décrémenté par 2 après chaque action jusqu'à retourner à 0. Enfin, si le fantôme est mangé, il retourne à sa position initiale et le compteur de peur est réinitialisé à 0.

La fonction *getMap* renvoie le contenu d'une des cases du plateau dont les indices de ligne et colonne sont données en paramètre. La valeur retournée peut être '#' si il s'agit d'un mur, '.' si il s'agit d'une gomme, '*' s'il s'agit d'une super-gomme, ou une autre valeur si il s'agit d'une case vide.

La fonction *getPacmanPosition* renvoie la position de Pac-Man sous la forme d'un objet de la classe *Position*. Cet objet a trois fonctions qui permettent d'accéder aux coordonnées du pacman : *getRow* qui renvoie l'indice de ligne, *getColumn* qui renvoie l'indice de colonne et *getDirection* qui renvoie la direction suivie (qui peut être 'U', 'D', 'L', 'R').

La fonction *getGhostPosition* renvoie la liste des positions (objets de la classe *Position*) possibles du fantôme dont l'indice est donné en paramètre. La structure de donné utilisée pour stocker ces positions est un *TreeSet*, qui est un arbre binaire de recherche.