8INF846 - Intelligence Artificielle



**Préalables :**

-On considère qu'une poussière et un bijou ne peuvent pas apparaître sur l’aspirateur (au même endroit où se trouve l’aspirateur).

-Pour implémenter le fait que l’environnement et l’agent fonctionnent sur 2 fils d’exécution différents nous avons utilisé les Threads, cependant il s’agit de la première fois que nous les utilisons concrètement, nous n’avons donc peut-être pas utilisé la méthode la plus optimale.

-Nous avons documenté notre code pour avoir plus de détails sur les fonctions créées et aider à la compréhension.

-Au début de l’application, différents paramètres sont à saisir pour voir comment évolue l’agent dans différentes conditions.

-La fin de la simulation peut être provoquée par un bouton stop, par un manque d’énergie de l’agent ou bien quand l’environnement se retrouve dans un état entièrement propre.

-Concernant l’exploration, la greedy search n’est pas totalement fonctionnelle.

-Nous n’avons pas pu implémenter la fréquence d’exploration

**Propriétés de l’environnement :**

-Complètement observable : l’aspirateur connaît en tout temps son environnement et ou sont les poussières et les bijoux.

-Stochastique : il y a un certain pourcentage de chance qu'une poussière ou un bijou apparaisse à un endroit aléatoire pendant que l’aspirateur se déplace

-Épisodique : l’aspirateur n’a pas besoin de raisonner sur le futur

-Dynamique: l’agent et l’environnement s'exécutent sur deux fils d'exécution différent

par conséquent l’environnement peut changer quand l’agent réfléchis.

-Discret: l’agent dispose d’une énergie maximum donc il a forcément un nombre d’actions limité.

-Agent: pour le moment il n’y a qu'un agent dans l’environnement.

*Pour l’environnement, on a une classe environnement qui contient une carte (qui représente le manoir), une mesure de performance (pour que l’agent se rende compte de sa performance), et différentes fonctions qui permettent de générer l’environnement de manière sporadique.*

*Une carte est représentée par une liste de noeuds et ce sont les noeuds qui détiennent les poussières et les bijoux.*

**Propriétés de l’agent :**

-Autonome: notre agent n’a besoin d’aucune aide pour fonctionner. Il perçoit l’environnement grâce à ses capteurs et agit grâce à ses effecteurs donc il n’y a aucune aide extérieure.

-Habileté sociale : pour le moment il n’en a pas mais à l’avenir il pourra discuter avec par exemple une maison connectée, voir même reconnaître les ordres donnés par un humain…

-Réaction : l’agent perçoit le manoir et agit en conséquence (déplace, aspire, ramasse)

-Pro-action: l’agent agit en fonction de l’état de l’environnement mais aussi en fonction de son but qui est d’atteindre un état propre.

Notre agent est un agent basé sur les buts. Il choisit ses actions en fonction de l'état de son environnement et de son but est d’obtenir un manoir propre.

*Notre classe agent est donc composée d’une classe capteur (qui lui donne les informations sur l’environnement), d’une classe effecteur (qui lui permet de faire les différentes actions possibles), et de toutes ses variables pour avoir ses statistiques.*

**Monde de l’agent Aspirateur :**

 Mesure de performance : nombre de bijoux aspirés.

 Environnement : manoir.

 Effecteurs : roue (déplacement), outils pour aspirer/ramasser.

 Capteurs : caméras, sonar, odomètre, indicateur de vitesse, capteurs du moteur, etc.

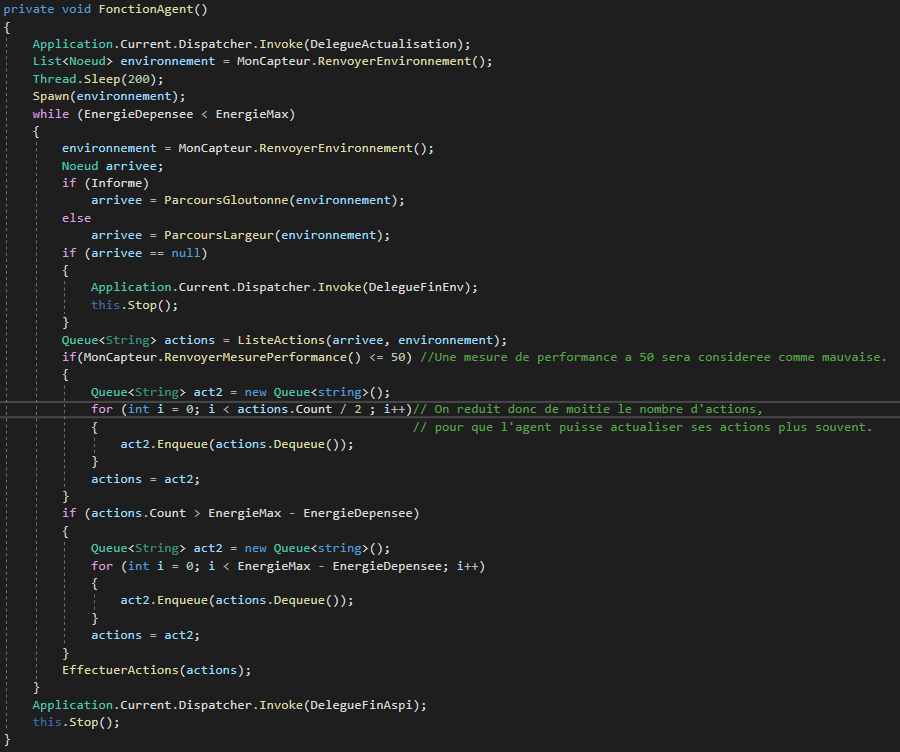
**État mental BDI :**

-Belief: l’agent perçoit l’environnement grâce à ces capteurs et il possède aussi d’autres connaissances comme par exemple l'énergie qu’il dépense par action, l’energie maximum …

-Desire: grâce à la fonction d'exploration il connaît la case la plus proche où il y a de la poussière à un instant T.

-Intention: il définit une liste d’actions à accomplir pour atteindre le but recherché; exemple: (droite, droite, aspirer).

Voici la fonction de agent qui implémente l’état mental BDI :



-Belief: environnement = MonCapteur.RenvoyerEnvironnement();

récupère l’environnement par la classe capteur.

-Desire: arrivee = ParcoursLargeur(environnement);

récupère le noeud d’arrivé qui contient la poussière.

-Intention: Queue<String> actions = ListeActions(arrivee, environnement);

définit la liste des actions à accomplir pour arriver au “Desire”.

**Exploration :**

-Pour ce qui de l’exploration non-informée nous avons choisi l’exploration Breadth-first search (largeur).



-Pour ce qui est de l’exploration informée nous avons choisi la Greedy search (gloutonne) et pour l’heuristique plus une case est proche d’une poussière et plus elle possède une heuristique faible.

