Cahier des charges projet IA04 : Flotte de drones en 2D

Sommaire

1 Présentation globale de l'idée

1.1 Définition d'un drone

Un drone (de l'anglais drone) désigne un aéronef sans pilote à bord. Le drone peut avoir un usage civil ou militaire. [...] Sa taille et masse (de quelques grammes à plusieurs tonnes) dépendent des capacités recherchées. Le pilotage automatique ou à partir du sol permet des vols longs de plusieurs dizaines d'heures (à comparer aux deux heures typiques d'autonomie d'un chasseur).
- Wikipédia : https://fr.wikipedia.org/wiki/Drone

1.2 Modélisation

On suppose qu'un agent représente un drone et on se place dans un plan 2D (xOy). Chaque drone est assimilé à un point qui représente sa position (x, y) dans le plan. L'objectif est de faire en sorte que les agents coexistent dans un périmètre donné. Les coordonnées sont positives, on se place dans $\mathbb{R}_+^2$. L'espace de déplacement (l'environnement) n'est pas torique et est limité par des bornes en x et y. L'espace n'est pas forcément carré.

1.3 Comportement des drones

La coexistence des drones est basée sur plusieurs règles de fonctionnement :

1. Les drones ont tendance à s'organiser en flotte, le nombre de drones maximal dans une flotte est une constante à définir au début du programme.
2. Les drones sont démarrés seuls au départ et ne font pas partie d'une flotte.
3. Chaque drone a un unique identifiant (qui dans JADE je suppose correspondra en fait à son AID).
4. Chaque flotte a un un maître qui guide sa flotte.
5. Les flottes sont en anneau, le diamètre peut varier en fonction des objets ou autres drones sur leur passage, de manière donc à éviter les collisions.
6. La communication des drones se fait par ondes radio à courte portée, ce qui concrètement veut dire qu'un drone ne peut émettre qu'à un certain rayon autour de lui, et de même ne recevoir qu'à une distance inférieure ou égale à ce rayon.
7. Lorsqu'un drone (ou une flotte) rencontre un autre drone (ou une autre flotte), une procédure est activée de manière à ce que les deux ensembles fusionnent (ou non, en fonction du nombre de drones déjà présents dans la flotte).
8. Lorsqu'un drone est détruit (par exemple il s'écrase contre un bâtiment, représenté par un polygone sur le plan), il est retiré du graphique (le point qui le représentait est effacé).
9. Inversement lorsqu'un drone est créé, le point le représentant apparait sur le graphique.
10. La communication utilisée dans une flotte est une communication de proche en proche (topologie en anneau) : https://fr.wikipedia.org/wiki/Topologie_de_reseau#Le_r.C3.A9seau_en_anneau

1.4 Objectif

L'idée est donc de modéliser ce système et d'y incorporer une **interface graphique** (animation) afin de visualiser en temps réel les déplacements et les différents interactions des drones.

Voici une image de ce à quoi cela pourrait ressembler (ce serait donc une capture à un instant donné de la figure graphique) :

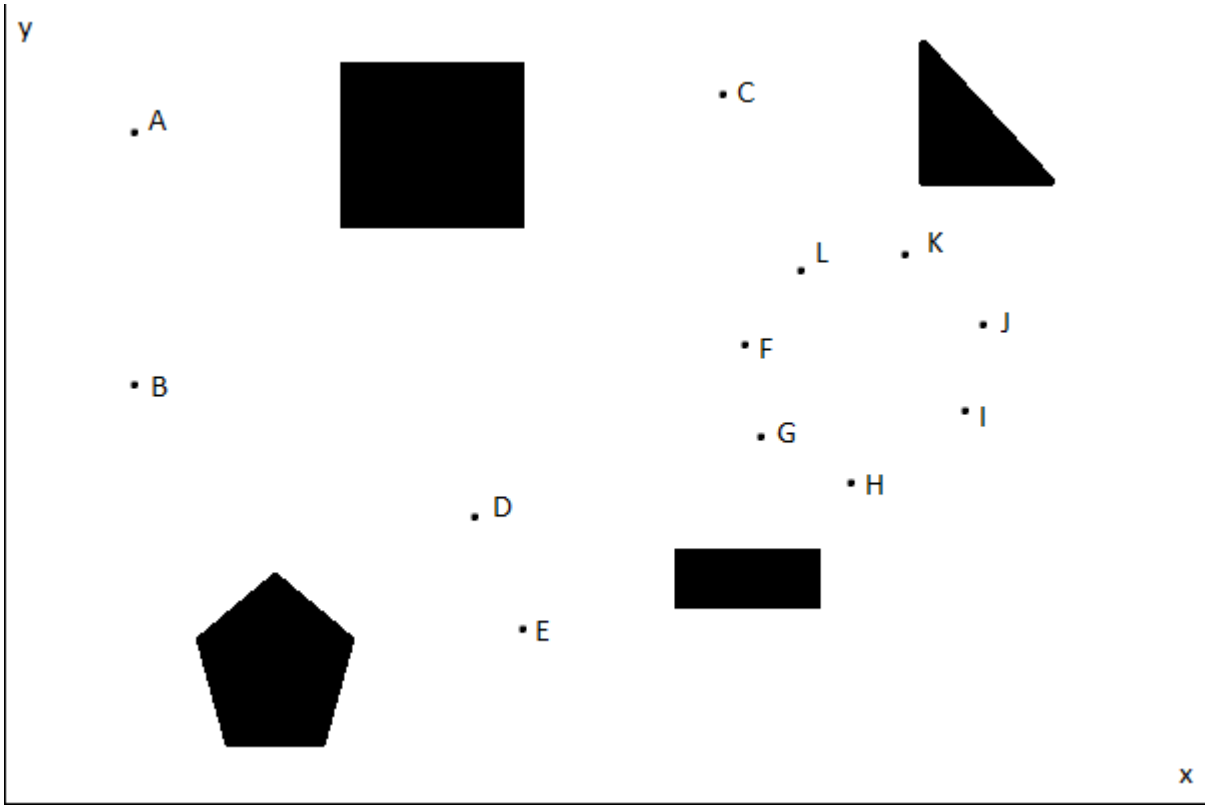

FIGURE 1 – Les points A, B, ... sont des drones et les polygones noirs sont des objets (facultatif)

2 Implémentation

2.1 Architecture JADE

L'architecture se divise en trois parties : les drones qui évoluent dans l'environnement 2D, les objets qui se trouvent sur le terrain, et l'interface graphique qui va représenter tout cela sur l'écran. Ces trois parties sont plus ou moins indépendantes.

2.2 La classe `Drone`

La classe `Drone` est la principale classe à implémenter. En termes de cardinalité, le nombre d'instances de cette classe peut varier de 1 à n.

2.2.1 Attributs

La liste d'attributs de la classe `Drone` :

1. La position actuelle du drone dans le plan $p = (x, y)$ qui est représentée par une instance d'une classe à définir, qui définira un certain nombre de méthodes (par exemple renvoyer la distance entre deux points du plan).
2. Un identifiant unique qui peut éventuellement être l'AID du drone.
3. Un statut à définir par la suite, notamment lors des fusion de flotte, il faudra imposer aux drones de ne plus se déplacer et d'ignorer les messages reçus de l'environnement, donc il leur faudra un état particulier qui les mettra en stand-by.
4. La prochaine position du drone (calculée en fonction de la position actuelle, de la position objectif, et de l'environnement immédiat).
5. La position objectif du drone.
6. La flotte à laquelle il appartient, s'il appartient à une flotte.
7. Son rang dans la flotte et le voisin auquel il doit envoyer un message (anneau unidirectionnel), s'il appartient à une flotte.

2.2.2 Behaviours

Le drone doit pouvoir réagir à son environnement de plusieurs manières, dépendamment des stimuli qui lui parviennent. Voici la liste des behaviours de la classe `Drone` :

1. Un behaviour qui renvoie la position de l'agent à l'agent `Display` quand il le lui demande.
2. Un behaviour qui envoie spontanément un message à l'agent `Display` lors de la mort de l'agent (l'agent `Display` devra dans ce cas supprimer le drone de la liste des drones et mettre à jour l'affichage).
3. Un behaviour qui réagit aux messages provenant de d'autres éléments de la classe `Drone`.
4. Un behaviour qui envoie périodiquement des messages dans l'environnement immédiat de l'agent (il se contente d'émettre aux autres agents, la réception du message se fera par une fonction de filtration sur l'attribut position).
5. Un behaviour qui réagit aux messages provenant des objets (dans cette modélisation, les objets sont des ensembles de points fixes dont les points périphériques envoient périodiquement des messages aux drones).

2.3 La classe `Display`

La classe `Display` est une classe indépendante de la classe `Drone`. Son rôle est de rappatrier les positions des drones et de les afficher sur une interface graphique. L'implémentation de classe passe donc par l'utilisation d'une librairie graphique Java. En termes de cardinalité, le nombre d'instances de cette classe est exactement de 1.

2.3.1 Attributs