# Rapport TP Multi-Agents sous JADE

## Choix du sujet et Scenarios :

Nous avons choisi le sujet 2, qui concerne des agents explorateurs chargés de récupérer des pierres rares sur une planète éloignée. Bien que nous sachions que ces pierres sont regroupées en tas, leur localisation précise demeure inconnue. Les agents n’ont pas accès à une carte de la planète et doivent naviguer sur un terrain très accidenté, parsemé d'obstacles. Ils sont conscients de la position du vaisseau mère, qui leur sert de base opérationnelle.

Les agents disposent d’une batterie qui peut être rechargée au vaisseau mère, mais leur capacité de collecte est limitée (fixée à trois pierres dans notre cas). Ils peuvent communiquer directement entre eux pour coordonner leurs efforts. La communication se fera principalement par le biais de requêtes de demande d’aide, par exemple, lorsqu'un agent est confronté à une quantité de pierres trop importante pour être transportée seul ou lorsqu'il a besoin d'une recharge de batterie. De plus, ils sont en mesure de retourner au vaisseau mère pour décharger les pierres collectées ou recharger leur batterie, ce qui leur permet de poursuivre leur mission tant que le vaisseau mère n’est pas plein.

## Modélisation BDI :

Dans notre système, chaque *SpaceAgent* (agent de collecte) est chargé de localiser, collecter et livrer des pierres au vaisseau mère. Ce dernier, en tant qu’agent central, sert de dépôt pour les pierres et gère la réception des ressources collectées.

### SpaceAgent (Agent de Collecte)

* **Beliefs :**
  + Position (x, y)
  + Nombre de pierres collectées (collected)
  + Capacité de collecte (capacity)
  + Niveau de batterie (batteryLevel)
  + État de la simulation et du vaisseau mère (isRunning, isMothershipFull)
* **Desires** :
  + Collecter des pierres (jusqu'à la capacité maximale)
  + Retourner au vaisseau mère lorsque :
    - Capacité de collecte atteinte
    - Batterie faible
  + Recevoir de l'aide en cas de besoin
* **Intentions** :
  + Explorer et collecter des pierres (exploration comportementale)
  + Communiquer avec d'autres agents pour :
    - Demander de l'aide (requestHelp)
    - Informer d'autres agents (notifyOtherAgents)
  + Se déplacer vers le vaisseau mère pour :
    - Déposer des pierres (returnToMotherShip)
    - Se recharger

### MotherShip (Vaisseau mère) :

* **Beliefs**
  + Position (x, y)
  + Nombre de pierres collectées (collectedStones)
  + Capacité maximale (maxCapacity)
* **Desires**
  + Accepter des pierres des agents (tant que la capacité n'est pas atteinte)
  + Rester opérationnel et à une position fixe
* **Intentions**
  + Accepter les dépôts de pierres des agents via depositStones(int stones)
  + Vérifier sa capacité de stockage via canAccommodate(int stones)

## Modélisation sous Jade :

**1. Preparation de l’environnement**

1. **Téléchargement de JADE :**  
   Nous avons télécharger le fichier JADE-all-4.5.0.zip à l'adresse suivante : [JADE Download](http://jade.tilab.com/download/jade/license/jade-download/).
2. **Extraction des fichiers :**  
   Après avoir téléchargé le fichier JADE-all-4.5.0.zip, on le décompresse et on obtient quatre autres fichiers ZIP :
   * JADE-bin-4.5.0.zip
   * JADE-doc-4.5.0.zip
   * JADE-examples-4.5.0.zip
   * JADE-src-4.5.0.zip  
     Décompressez ces quatre fichiers.
3. **Mise à jour de la variable CLASSPATH :**  
   Il faut mettre à jour la variable d'environnement CLASSPATH.
4. **Configuration de la variable CLASSPATH :**  
   Dans la zone « Variables système », On attribue a la varaible CLASSPATH la valeur suivante, qui est la concaténation des chemins des fichiers jade.jar et jadeExamples.jar situés dans le chemin suivant :

Exemple : C:\JADE-all-4.5.0\JADE-bin-4.5.0\jade\lib\jade.jar;C:\JADE-all-4.5.0\JADE-bin-4.5.0\jade\lib\jadeExamples.jar

1. **Configuration dans IntelliJ :**  
   a. Ouvrir IntelliJ IDEA et accéder à la structure du projet en allant dans File > Project Structure.  
   b. Dans le panneau de gauche, sélectionner l'onglet Project Settings, puis cliquer sur Libraries.  
   c. Cliquer sur le bouton + pour ajouter une nouvelle bibliothèque et choisir l'option Java.  
   d. Naviguez jusqu'au dossier .../JADE-bin-4.5.0/jade/lib.  
   e. Sélectionner le fichier jade.jar et cliquez sur OK pour l'ajouter au projet.

Maintenant que tout est installer on peut lancer Le GUI de Jade pour tester, avec la commande suivante,

java jade.Boot -gui

On obtient alors l'interface graphique de JADE.

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

Figure 1 Interface graphique de JADE

Pour notre projet, nous avons opté pour une interface graphique plus interactive. Nous avons utilisé Java Swing et créé trois classes : une pour les SpaceAgents et une pour le MotherShip (qui est également un agent), avec les différents comportements requis pour répondre à notre sujet.

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquementUne image contenant texte, capture d’écran, carré, mots croisés

Description générée automatiquement

Figure 2 Interface interactive de l'application

Notre interface est assez intuitive et permet de visualiser la simulation. Elle offre la possibilité de choisir la taille de la grille qui représente notre planète, le nombre de tas de pierres qui seront disposés aléatoirement, le nombre d’agents que l’on souhaite envoyer, ainsi que le nombre d’obstacles. Cette interface s’appuie sur JADE. Les deux interfaces sont connectées, ce qui permet de contrôler la simulation depuis les deux.

## Description de l'Architecture et du Code

**1. Classes Principales**

* **SpaceAgent**
  + Représente un agent autonome qui explore l'environnement.
  + **Attributs :**
    - x, y : Position de l'agent sur la grille.
    - batteryLevel : Niveau de batterie de l'agent.
    - collectedStones : Nombre de pierres collectées par l'agent.
  + **Méthodes Principales :**
    - move(): Change la position de l'agent.
    - collectStone(): Ramasse une pierre si présente.
    - depositStones(): Dépose des pierres dans le vaisseau mère.
* **MotherShip**
  + Représente le vaisseau mère qui collecte les pierres.
  + **Attributs :**
    - x, y : Position du vaisseau mère sur la grille.
    - collectedStones : Nombre total de pierres collectées.
    - maxCapacity : Capacité maximale de stockage.
  + **Méthodes Principales :**
    - depositStones(int stones): Dépose un certain nombre de pierres.
    - canAccommodate(int stones): Vérifie si la capacité permet d'ajouter plus de pierres.
* **PlanetVisualizer**
  + Gère l'affichage graphique des agents, des pierres et des obstacles sur la grille.
  + **Attributs :**
    - gridSize : Taille de la grille.
    - agents : Liste des agents actifs.
    - stones : Matrice représentant les pierres.
    - obstacles : Matrice représentant les obstacles.
    - mothership : Instance de MotherShip.
  + **Méthodes Principales :**
    - paintComponent(Graphics g): Dessine la grille et les éléments visuels.
    - initializeStones(int numStones): Initialise les pierres sur la grille.
    - initializeObstacles(int numObstacles): Initialise les obstacles sur la grille.
    - updateAgentPosition(int agentIndex, int x, int y): Met à jour la position d'un agent.

Amelioration potentiel ? Je ne sais pas trop quoi mettre la