TP 4 Java sur Eclipse

Contrôle direct du mouvement des robots via l'IHM

Objectifs du TP

- Piloter les robots en complétant l'interface utilisateur ...

Travail à réaliser

Il s'agit de reprendre un des programmes de robots footballeurs/cueilleurs précédent avec un pilotage manuel à la souris dans un premier temps...

Création d'un nouveau projet :

- Choisir un des projets robots footballeurs ou cueilleurs et faire une copie du projet car on va modifier certaines classes sources de Simbad.
- L'idée ici est d'ajouter dans l'interface existante de Simbad, une programmation de l'interface qui permette de contrôler/piloter un seul robot.

I – Ajout de boutons de direction

Les classes de l'IHM Simbad sont regroupées dans le package **simbad.gui,** notamment les classes qui nous intéressent ici (GUI signifie Graphical User Interface) : **SimulatorControlGUI.java**, **WorldControlGUI.java** et **ControlWindow.java**

Analyser ces deux classes et comprendre leurs liens avec l'interface.

A faire :

1. En vous inspirant de la classe **WorldControlGUI.java**, créer une nouvelle classe **AgentControlGUI.java** dans le même package, qui implémente deux boutons Swing (classe JButton) **left** et **right** qui font tourner le robot sans modifier sa vitesse respectivement à gauche ou à droite d'un angle de 0.5 radian (la méthode **rotateY** d'un agent prend en paramètre des radians pour les angles. Rappel : 180 degrés d'angle correspondent à 3,14 radians). Vous pouvez également utiliser la méthode **setRotationalVelocity()** du robot pour avoir un mouvement de rotation plus progressif (de plus on maîtrise clairement ainsi le sens de rotation, contrairement à l'utilisation de **rotateY**).
   NB : l'attribut **agentFollower** sera remplacé par l'attribut **robotFollower**, instance du robot récupérée grâce à la classe **Simulator.** Il détermine et fait le lien avec l'agent robot à contrôler.
2. Modifier la méthode **performBehavior** du robot à contrôler (dans sa classe de définition) de façon à ce qu'il ne tourne pas de façon autonome, mais possède une vitesse constante pour tester le pilotage du robot.
3. Tester le pilotage avec les deux boutons réalisés.

II – Contrôle de la vitesse

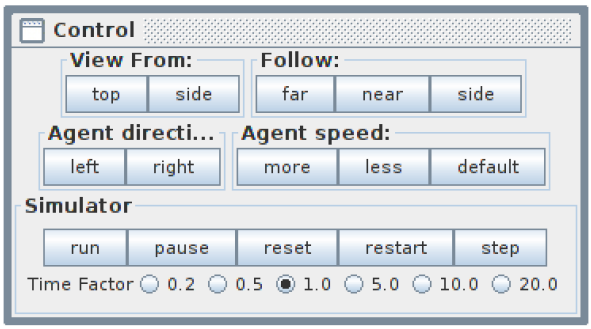
De la même façon que précédemment, faire deux boutons **more** et **less** dans le même cadre (l'instance **JPanel** conteneur des boutons) pour jouer sur la vitesse du robot pour augmenter ou

diminuer de 0.5 m/s la vitesse du robot.

NB : pour cela, il faudra maintenir un attribut de vitesse dans la classe **MyRobot** et deux méthodes **set** et **get** associées pour contrôler la vitesse du robot via l'interface.

L'instance du robot **robotFollower** permettra de contrôler la vitesse.

Voici une représentation de l'interface obtenue :



III – Contrôle de l'orientation et de la vitesse par des sliders

L'utilisation de boutons n'est pas très pratique ici, c'est pourquoi on va les remplacer par des sliders Swing (classe **JSlider**) également contrôlable par le clavier.

Rappels : un **JSlider** réagit à un événement de type **change** et donc doit être associé à un écouteur **ChangeListener** de cet événement.

Ensuite, on doit implémenter la méthode de comportement associée à l'événement :

```
public void stateChanged(ChangeEvent changeEv) { … }
```

On peut/doit également changer la configuration du slider (valeurs max et min, incrément...), par exemple par les méthodes de la classe **Jslider** suivantes :

```
setMaximum, setMinimum, setValue, setMajorTickSpacing, setName, …
```

Pour un exemple complet sur l'usage du composant JSlider, consulter le tutoriel d'Oracle sur la page web :

https://docs.oracle.com/javase/tutorial/uiswing/components/slider.html

S'en inspirer pour adapter le code précédent afin de remplacer les boutons par des sliders. On créera une nouvelle méthode **createSlider(String name, Container container)** dans la classe principale qui attribue un nom au slider et l'associe à un composant conteneur (de type JPanel a priori).

NB : penser à ajouter l'interface **ChangeListener** à votre classe principale.

IV – Contrôle au clavier des sliders

On souhaite maintenant contrôler les sliders par des touches au clavier plutôt qu'avec la souris, et donc d'ajouter en plus un écouteur d'événements clavier à chacun des sliders grâce à la méthode de la classe **JSlider** à compléter lors après la création de chaque slider :

slider1.addKeyListener(new KeyListener(){

public void keyPressed(KeyEvent e){ }

… })

Vous aurez à implémenter une des méthodes associées à un événement du clavier (**keyPressed**, **keyReleased**, **keyTyped**) sur des touches à définir pour l'orientation d'une part et pour la vitesse d'autre part.

NB : vous pouvez également utiliser une classe **KeyAdapter** à la place de **KeyListener** pour éviter d'avoir à mentionner toutes les méthodes de l'interface.

De même, vous pouvez trouver un exemple d'utilisation de **KeyListener** ici :

https://docs.oracle.com/javase/tutorial/uiswing/events/keylistener.html

NB : pour travailler avec une chaîne de caractère pour les touches => Utiliser les méthodes *getKeyCode* puis *getKeyText.*

Tester la nouvelle interface.