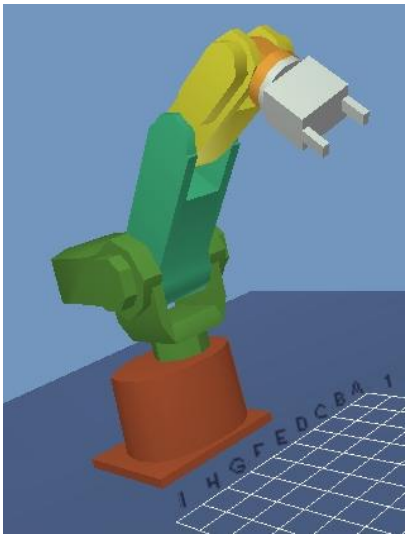


Tout simplement JRobot

Etudiant :
Vladimir Calderón

CYBERNÉTIQUE APPLIQUÉE
2002

Table des matières

JRobot

1 But du Projet et Introduction

Le but du projet était de compléter un logiciel (une interface graphique) pour manipuler le robot RVM1 avec plusieurs possibilités. Les objectifs de base étaient les suivants :

- Utilisation de librairies standard.
- Facil, convivial et très intuitif à utiliser.
- Logiciel complet, c.a.d. tous les boutons servent à quelque chose, les menus aussi, etc.
- Installation facile, le but est d'avoir la suite : **download**, **compile** et **run**.

Une fois ces objectifs en tête, voici le travail qui devait être fait. Mais d'abord un état du logiciel au début du travail :

1. Très souple interface graphique faite en Java, utilisant des composants Swing et les librairies : RVM1[1] et une librairie pour la connection du robot avec le port série COM2.
2. Possibilité de manipuler le robot avec des 'sliders'.
3. Programmation possible avec un éditeur complet de programmes. Ensuite ces programmes peuvent être facilement exécutés.
4. Une petite fénêtre avec une aide.

Enfin, voici le travail qui devait être fait pour compléter ce logiciel, que dorénavant on appellera JRobot :

1. Possibilité de voir l'exécution des commandes en temps réel via une caméra pointée sur le robot. La fénêtre doit être dans le logiciel disponible à tout moment.
2. Une représentation 3D virtuelle du robot, le plus proche de la réalité pour pouvoir faire de la programmation avec un RVM1 virtuelle.
3. A part la manipulation avec les 'sliders', prévoir la possiblité de manipuler le robot avec un joystick.

On a donc trois grandes tâches à accomplir. Les descriptions et les choix pour la programmation de chaqu'une d'elles seront exposées dans les sections suivantes.

On a voulu donner le plus de descriptions graphiques pour les nouvelles capacités de JRobot. On expliquera juste le choix de programmation, librairies, etc et une très brève description de la programmation en soi, pour le développeur on donne à la fin les diagrammes de classe expliquant les trois parties du travail.

A la fin on présente une section avec des possibles améliorations du logiciel. Le produit est maintenant très puissant mais, comme toujours, on peut mieux faire.

Pour finaliser l'introduction on voudrait remercier Patrick Roth pour les conseils qui font la manipulation de JRobot très intuitive.

2 Caméras et Protocole RTP

Java s'améliore d'année en année, ou devrait-on dire qu'il s'agrandit d'année en année. Tout soit pour dire que la librairie standard de Java pour des composants multimedia JMF est maintenant dans sa version 2.1.1 et elle est très complète et assez puissante.

[1] Excellente librairie faite à l'université pour la manipulation de base du robot

C'est pour cette raison que l'on a choisit de travailler avec cette librairie qui se trouve disponible gratuitement dans la site de Java.

Cette librairie va nous permettre d'afficher une petite fenêtre avec l'image captée par notre caméra.

De plus, cette librairie nous a fait connaître le protocole RTP qui nous a permis de nous connecter à une caméra à distance. Ceci en vue d'utiliser le robot à distance.

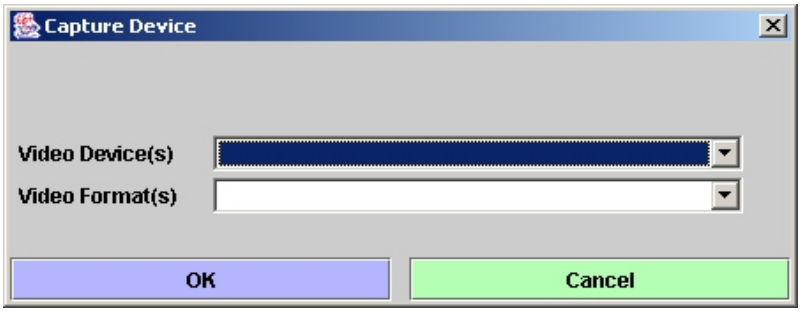


FIG. 1 – Choix du device

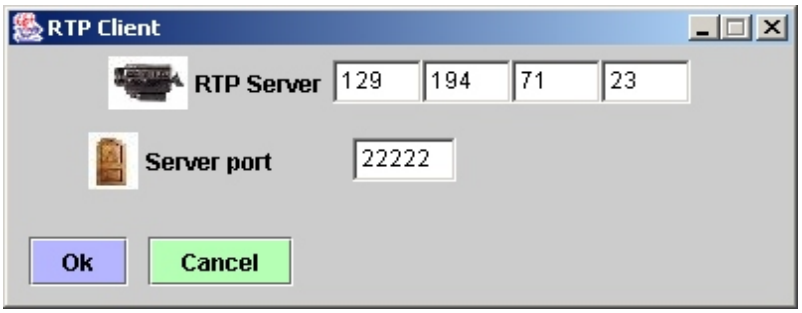


FIG. 2 – Remote capture via RTP

Pour le logiciel on a donc développé l'option de la façon suivante :

- Paneau avec toutes les fonctions de caméra.
- Bouton pour se connecter à une caméra branché sur l'ordinateur. La fénêtre de dialogue cherche tous les devices qui ont une caméra. De plus on peut choisir le format de sortie, voir image 1.
- Bouton pour se connecter à une caméra à distance, on a juste besoin de savoir le numéro IP du serveur qui tient la caméra et le numéro de port sur lequel se trouve ce service, voir image 2.
- Des boutons pour afficher et enlever l'image de notre logiciel.

2.1 Serveur RTP

Puisque l'on avait devéloppé le client pour une connection RTP, il nous a fallu aussi devélopper la partie serveur. Heureusement, dans la documentation de JMF il existe des exemples assez détaillés qui nous ont facilité la tâche.

Le serveur deevéloppé déploit une petite fénêtre qui donne à l'utilisateur le choix du device à exporter, la plage d'adresses qui peuvent y accéder et le port de service.

3 Robot RVM1 Virtuel

Le problème maintenant est de pouvoir afficher une représentation 3D le plus réaliste possible de notre robot RVM1. Pour ce faire on a donc décide d'utiliser la librairie standard de Sun : Java3D.

Cette librairie utilise soit OpenGL, soit DirectX, pour les opérations de base en 3D. Ceci permet à l'application d'être très rapide avec ces calculs. On dira donc que Java3D est en fait une interface pour ces fonctions.

Le grand plus de cette interface est qu'elle a été conçue avec des objets. On a donc partout la facilité de la programmation objet (merci Java). D'autre part le devéloppeurs de Java3D ont opté pour une représentation hiérarchical pour toute scène que l'on veuille rendre.

Pour donner l'exemple de notre projet et nous mettre tout de suite dans JRobot, voici le graphe responsable du rendu de l'image virtuelle d'un robot RVM1 :

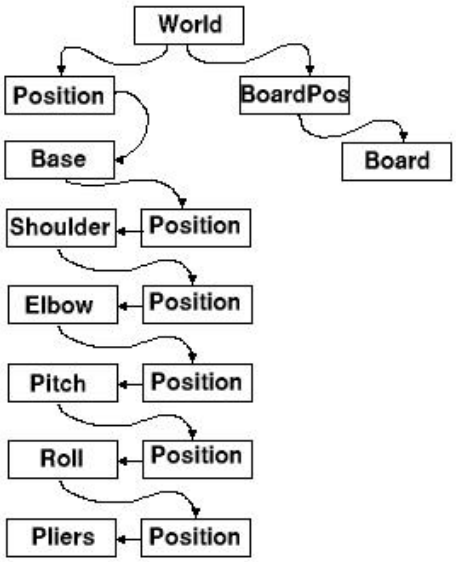


FIG. 3 – Graphe pour la scène avec robot RVM1

3.1 Les objets

Comme on peut s'imaginer, les objets qui composent le robot devaient être le plus réaliste possible, ou du moins, les mesures des extremités devaient être proches de la réalité.

Pour bien accomplir la tâche, on a procédé comme suit :

- Des mesures à la milimètre près d'après une fiche technique du robot RVM1.
- Décomposition des différentes parties du robot en essayant de simplifier les formes tout en gardant un certain détail sur les pièces.
- Implantation des objets dans JRobot par leur nom. Il faut donc juste changer une ligne pour remplacer une pièce pour une autre implantation, peut-être avec plus de détails.

Dans la figure 4 on peut voir le résultat que l'on a obtenu. On a limité le rendu des objets à un maillage pour pouvoir apprecier les détails des polygones et arêtes qui forment les parties du robot.

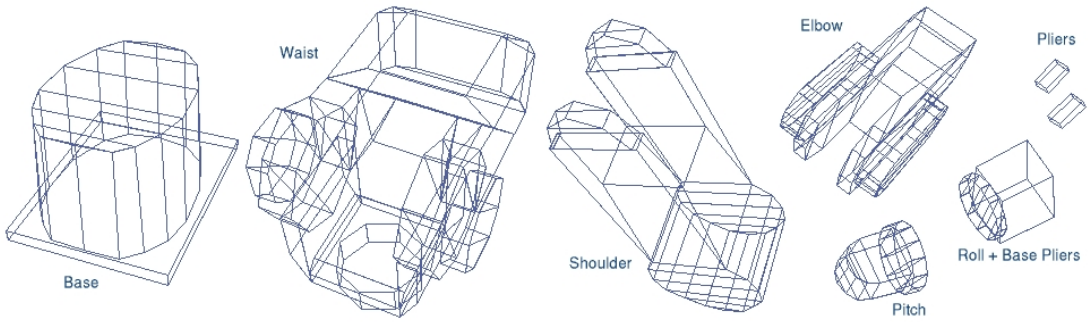


FIG. 4 – Objets du RVM1 rendu juste avec maillage

Le résultat final est donc assez satisfaisant, mais il manque encore des détails au monde de notre robot...

3.2 L'aire de travail du robot

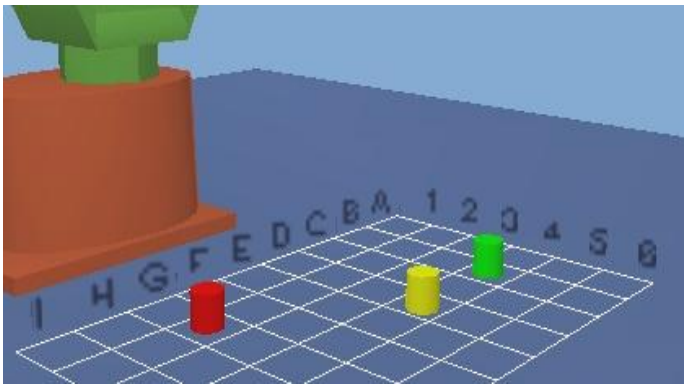
Comme on peut voir à la figure 3, on a ajouté un tableau de travail au monde de notre robot. Ceci pour pouvoir voir le déroulement des commandes avec des cibles virtuelles.

De cette façon on est de plus en plus rapprochés de la réalité. Pour l'instant cette aire de travail est rudimentaire. Elle ne nous permet que :

- La positionner par rapport au robot.
- Placer des objets cibles en forme de petits cylindres qui rappellent les cibles en bois qui existent au laboratoire de cybernétique.

Les caractéristiques le plus importantes pour cette aire de travail sont notamment la facilité dans la programmation pour placer d'autres objets à l'avenir et l'affichage des indexes pour la grille posé sur l'aire de travail.

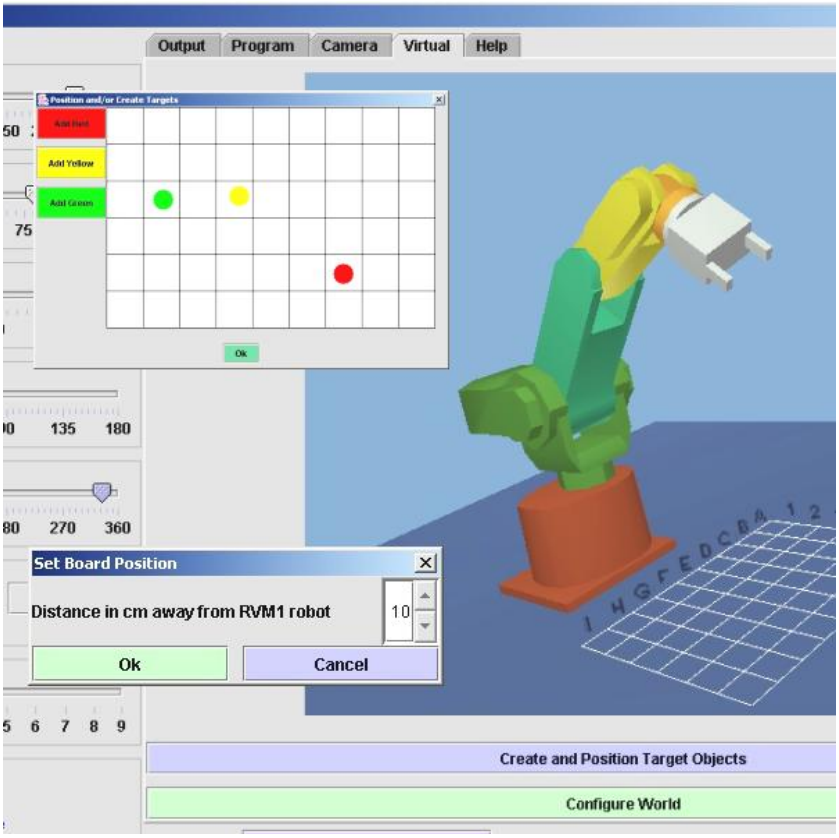
FIG. 5 – Effet de billboard pour les indexes de la grille



C'est une russe assez souvent utilisé en synthèse d'images pour avoir des objets qui semblent être en 3D mais en fait ce n'est qu'un plan en 2D qui reste perpendiculaire à l'axe de vision de l'utilisateur. De cette façon on voit toujours l'index (voir figure 5).

Voici donc le résultat visuelle de toute cette partie virtuelle, on peut voir une fénêtre de dialogue pour placer des cibles, une autre pour la position de la grille ; et finalement une autre qui montre le robot virtuelle en action.

FIG. 6 – Fénêtres devéloppés pour le monde virtuel de JRobot



4 Manipulation du RVM1 avec un Joystick

L'ajout de la possiblité de manipulation avec un joystick signifiait plusieurs problèmes. Le plus important était de trouver une bonne relation, intuitive, entre le joystick et le mouvement d'un robot avec 5 degrés de liberté.

La librairie choisie fonctionne correctement avec Windows, pour n'importe quel joystick. Il existe la possibilité d'utiliser une version similaire en Linux mais il manque de tests pour cela. L'intérêt de porter une telle librairie en Linux est intéressante pour JRobot dans la mesure où tout le programme fonctionne très bien avec Linux, il ne manque qu'une librairie Linux faite en Java pour le joystick.

Le joystick proposé pour l'expérience était un Joystick type Warrior. Ces types de joystick ont un bouton prevu pour commander des rotations. Le mapping proposé était donc :

- Gauche - Droite de la manette pour la hanche (waist) du robot.
- Haut - Bas de la manette, tout en appuyant sur des boutons différents, pour chacun des degrés de liberté.
- Un bouton pour commander la fermeture et ouverture des pinces.
- Le bouton rotationnel pour le poignet du robot.

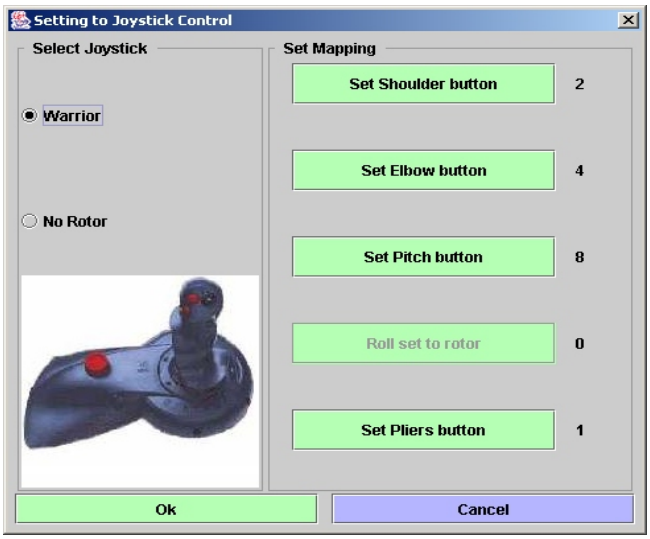


FIG. 7 – Fénêtre de dialogue pour determiner le mapping

Après une conversation avec Patrick Roth, il a été décidé que, puisque le produit était censé être souple et à la porté de tout le monde, il fallait donc donner l'option à l'utilisateur parmi 2 types de joystick (minimum). Et, bien evidement, donner la possiblité de determiner sa propre relation boutons - degrés de liberté (mapping).

La fénêtre de dialogue pour ceci est la figure 7. Celle-ci est affiché quand on appui sur le bouton pour passer à commander avec le joystick. Voici l'aperçu de la fénêtre principal qui change entre la commande avec joystick et des sliders :

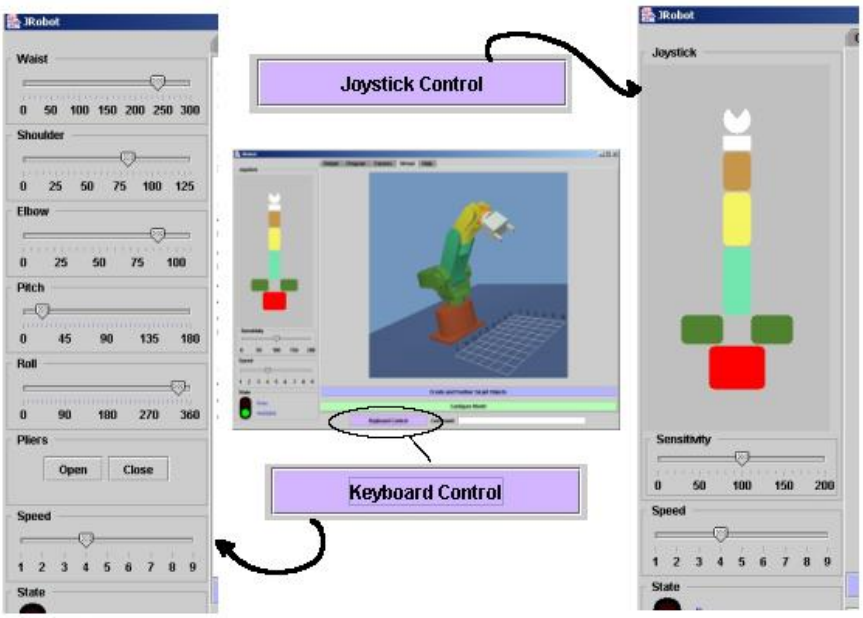


FIG. 8 – Changement d'aspect pour joystick

5 En parlant de l'avenir de JRobot

Comme on l'avait remarqué au début de ce document pour l'instant JRobot est assez puissant et très conviviale et surtout, très poli (toutes les erreurs et messages s'affichent tout de suite au premier plan).

Mais il faut toujours voir un peu plus loin. C'est pour cela que nous avons choisi quelques améliorations possibles dans un avenir très proche :

- Amélioration de l'interactivité entre cibles et robot pour le monde virtuel. Determiner jusqu'à quel niveau d'interactivité veut-on aller.
- Ajout des types de joysticks pour le choix de l'utilisateur.
- Possibilité d'avoir l'aperçu de plusieurs caméras en même temps, au moins deux pour avoir une meilleure vue du robot.

6 Disponibilité de JRobot

Dans ce document on n'a pas expliqué encore la façon dont le programme doit se compiler ou exécuter. Veuillez donc aller au site :

`http ://cui.unige.ch/ caldero6/jrobot/index.htm`

Pour obtenir un fichier zip qui contient tout le logiciel. Dans le site il existe une documentation pour l'installation et exécution du programme.

Juste pour montrer la souplesse de JRobot, voici une session typique une fois dans le site :

1. Décompresser le fichier `jrobot.zip` dans un répertoire choisi.
2. Installer J2SDK v.1.4 (`http ://java.sun.com/produits`)
3. Installer JMF v.2.1.1 (`http ://java.sun.com/produits`)
4. Installer Java3D (`http ://java.sun.com/produits`)
5. Compiler le tout avec **compile.bat**
6. Profitez de JRobot avec **run.bat**

7 Conclusion

Ce travail nous a permis de connaître des librairies de Java très intéressantes comme le sont Java3D et JMF. Et surtout a renforcé notre connaissance en synthèse d'image. La documentation de Java étant très complète il n'aura fallu que lire exactement la description de objets pour trouver la bonne façon de faire.

Les conseils de Patrick Roth ont été très enrichissantes pour ne pas s'arrêter au milieu d'un logiciel qui est prometteur. Le produit final est assez satisfaisant.

Peut-être le seul desavantage de travailler avec Java est que, puisque les librairies reposent elles-mêmes sur des librairies propres au système d'exploitation, des fois il existe des erreurs qui ne se montrent pas au niveau de la JVM.

A Diagramme de classes pour le module caméra

Le module pour la caméra nous demandait d'abord de créer une fénêtre exclusive pour la caméra. Ceci pour pousuivre avec le modèle de programmation que JRobot avait adopté au début.

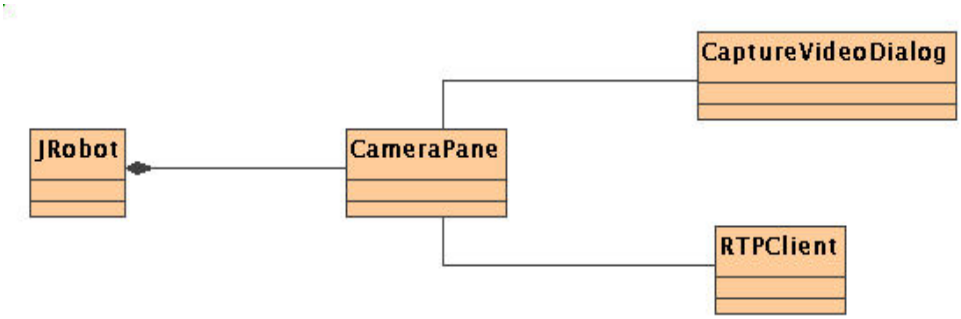


FIG. 9 – Diagramme de classes

Deux actions générées par 2 boutons déclenchent l'affichage de fénêtres de support pour la prise de données. Ensuite deux autres actions déclenchent : le début de l'affichage des images de la caméra et l'arrêt de cette affichage.

Les fichiers impliqués

La suivante est une liste exhaustive des fichiers qui comprennent cette partie :

1. jrobot/CameraPane.java
2. jrobot/CaptureVideoDialog.java
3. jrobot/RTPClient.java
4. rtpserver/RTPServer.java
5. rtpserver/VideoTransmit.java

B Diagramme de classes pour le robot virtuel

Ce module comprend la mise en programmation du graphe qui représente la scène (voir image 3) et ensuite attacher chaqu'une de ces parties à un élément qui va réagir quand une commande sera donné au robot.

Ces éléments sont, pour Java3D, des 'Behaviors' (ou comportements) qui peuvent être réliés à un groupe de transformation. Ensuite, une fois dans le traitement des évènements sur ce comportement on peut modifier la matrice de transformation courante pour ce groupe de transformation. On peut aussi brancher et débrancher des feuilles ou branches qui se trouvent au dessous de la hiérarchie.

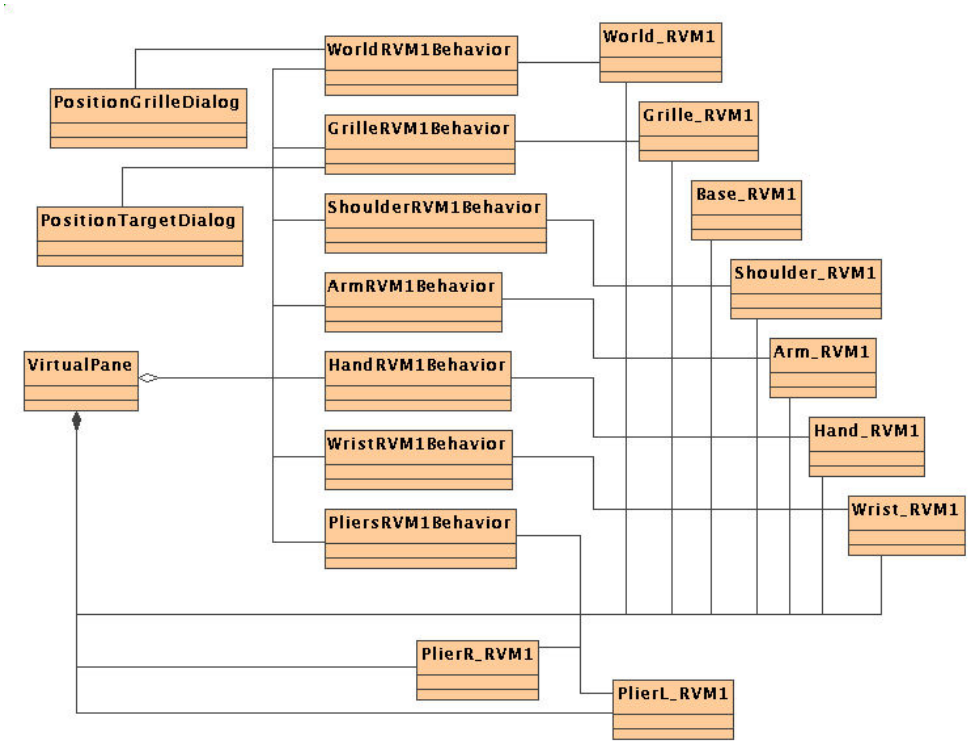


FIG. 10 – Diagramme de classes

Pour notre robot on voit donc que chaque partie de robot à son propre comportement. On signale que le fait que le calcul du rendu se fait en multipliant en cascade toutes les matrices de transformation d'un graphe, simplifie la tâche dans chaque noeud.

Ainsi, par example, pour l'épaule du robot on ne doit en fait que changer l'angle dans un des axes pour faire plier le robot. Le fait de trouver cet épaule plus à gauche ou à droite dépend de la hanche du robot qui se trouve, elle, plus haute dans la hiérarchie du graphe.

Les fichiers

La liste de tous les fichiers nécessaire pour la visualisation virtuelle du robot RVM1 est la suivante :

1. jrobot/VirtualRVM1Pane.java
2. jrobot/PositionGrilleDialog.java
3. jrobot/PositionTargetDialog.java
4. jrobot/Object2DGrille.java

5. jrobot/GrilleCanvas.java
6. jrobot/GrilleRVM1Behavior.java
7. jrobot/ArmRVM1Behavior.java
8. jrobot/HandRVM1Behavior.java
9. jrobot/PliersRVM1Behavior.java
10. jrobot/ShoulderRVM1Behavior.java
11. jrobot/WaistRVM1Behavior.java
12. jrobot/WorldRVM1Behavior.java
13. jrobot/WristRVM1Behavior.java
14. jrobot/AppearanceCreator.java
15. jrobot/virtual/objects/Arm_RVM1.java
16. jrobot/virtual/objects/Base_RVM1.java
17. jrobot/virtual/objects/Grille_RVM1.java
18. jrobot/virtual/objects/Hand_RVM1.java
19. jrobot/virtual/objects/LittleCylinder.java
20. jrobot/virtual/objects/PlierL_RVM1.java
21. jrobot/virtual/objects/PlierR_RVM1.java
22. jrobot/virtual/objects/Shoulder_RVM1.java
23. jrobot/virtual/objects/Waist_RVM1.java
24. jrobot/virtual/objects/World_RVM1.java
25. jrobot/virtual/objects/Wrist_RVM1.java

C Diagramme de classes pour la manipulation avec Joystick

JRobot fait appel à un objet `ConfigJoystick` pour acquérir un objet `Mapping` et une instance de `Joystick` pour la manipulation du robot.

C'est dans la configuration du joystick que l'on pourra ajouter de nouveaux types de joystick.

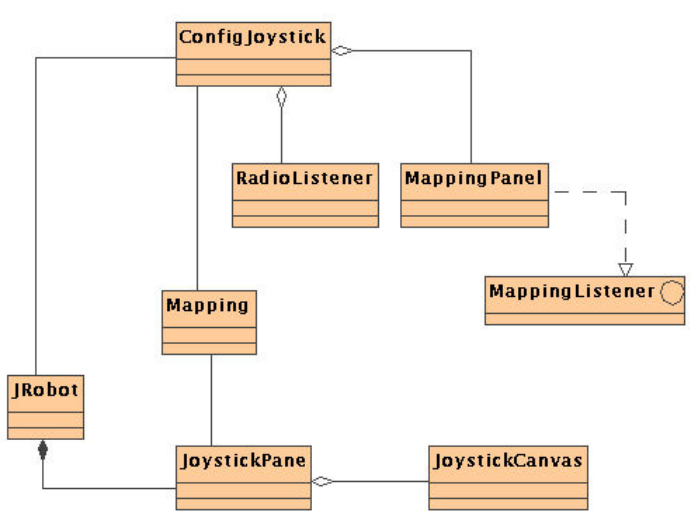


FIG. 11 – Diagramme de classes

Ensuite le JRobot n'a besoin que de cette paire mapping-joystick pour pouvoir commander le robot. L'objet `JoystickCanvas` ne sert que pour faciliter la vue des actions du joystick. Un effort a été fait pour rendre l'intéraction très visuelle.

Les fichiers

La liste de tous les fichiers nécessaire pour la manipulation du robot avec un joystick est :

1. jrobot/joystick/ConfigJoystick.java
2. jrobot/joystick/MappingPanel.java
3. jrobot/joystick/MappingListener.java
4. jrobot/joystick/Mapping.java
5. jrobot/joystick/JoystickPane.java