Le Middleware MOOS-IvP

Robotique/UV5.7 - Session MOOS-IvP #2 - Décembre 2018

Supports de cours disponibles sur
`www.simon-rohou.fr/cours/moos-ivp`

Simulation d'un bateau pilotable dans l'anse du Moulin Blanc

Dans cet exercice, l'objectif est de réutiliser des MOOSApp déjà existantes et de les intégrer au sein d'une communauté MOOS simulant un robot de surface. Le bateau, piloté par un joystick disponible pendant ce TP, sera affiché dans une vue 2D. Nous verrons lors du prochain TP comment rendre ce bateau autonome.

Les applications suivantes sont à utiliser :

- `pMarineViewer` pour l'affichage du robot sur une carte
- `uSimMarine` pour la simulation du robot
- `uJoystick` pour l'utilisation d'un joystick
- `pNodeReporter` pour faire une synthèse de l'état du robot

L'architecture suivante est proposée :

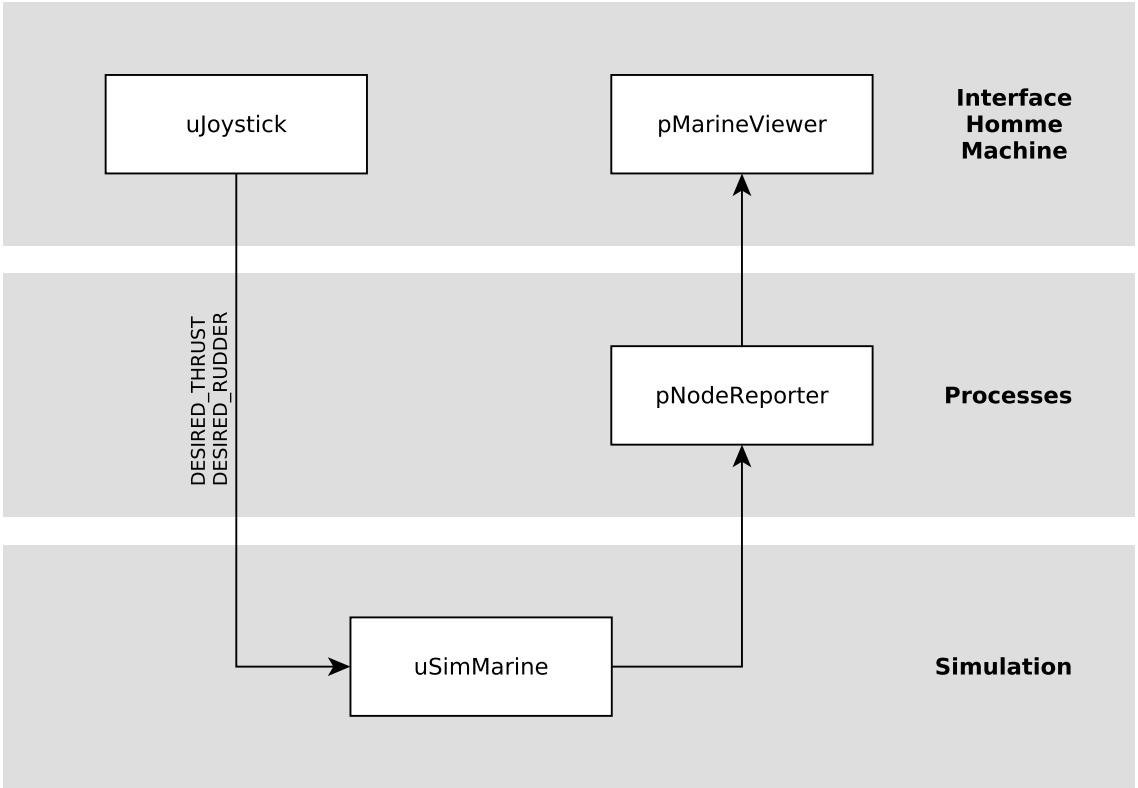

Figure 1: Organisation de la communauté MOOS — les échanges de variables MOOS entre les applications (représentées par des flèches) se font en réalité *via* la MOOSDB, non représentée ici.

1. **Architecture**
   Avant toute chose, il convient de comprendre le fonctionnement de la communauté proposée en Fig. 1. En s'aidant des documentations des applications[1], compléter la Figure 1 en inscrivant les variables MOOS échangées entre les applications.
   - on ne s'intéressera qu'aux variables qui nous semblent pertinentes dans ce projet
   - les variables ayant par défaut le préfixe `USM` seront préfixées par `NAV`

2. **Nouvelle mission**
   Créer un nouveau répertoire dans `moos-ivp-extend/missions/`. Y ajouter un fichier `.moos` contenant un bloc de configuration `ANTLER` lançant les MOOSApp précédemment listées.
   On choisit de définir un point de référence géographique pour la mission[2] (avant le bloc `ANTLER`). Les coordonnées retenues pour cette origine sont $(48.3909425, -4.4346433)$.

3. **Affichage**
   Ajouter un bloc de configuration pour `pMarineViewer`. Afin de visualiser les rapports[3] de chaque application MOOS dans cette interface, configurer `pMarineViewer` pour intégrer l'affichage des `AppCast`. Enfin, configurer l'application pour afficher la carte du Moulin Blanc avec un zoom de 1. La carte est téléchargeable ici : `www.simon-rohou.fr/cours/moos-ivp/doc/tp2_complements.zip`

4. **Synthèse du robot**
   Ajouter un bloc de configuration pour `pNodeReporter`. En suivant la documentation de l'application, configurer un robot de type `ship` et d'une longueur de $10m$.

5. **Simulation du robot**
   Ajouter un bloc de configuration pour `uSimMarine`. En suivant la documentation de l'application, configurer la position initiale du robot en $x = 500$, $y = 0$. Les variables d'état publiées par cette application devront être préfixées par `NAV`. Pour cela, deux solutions : utiliser `pEchoVar` ou bien configurer `uSimMarine`.

6. **Pilotage**

   - **Avec un joystick :**
     Ajouter un bloc de configuration pour `uJoystick`. Dans le répertoire `moos-ivp-extend/src/`, ajouter l'application `uJoystick` téléchargeable sur `www.simon-rohou.fr/cours/moos-ivp/doc/tp2_complements.zip`. Mettre à jour le fichier `moos-ivp-extend/src/CMakeLists.txt` puis compiler.
     Le paramètre `DEVICE_NAME` devra peut-être être mis à jour.

   - **Sans joystick :**
     Dans le fichier `.moos`, ajouter le bloc de configuration suivant :

```
ProcessConfig = uTimerScript
{
  AppTick = 0.1 // le robot va changer de direction toutes les 10s

  // à chaque appel du script, une commande aléatoire est envoyée
  randvar = varname=RANDOM_RUDDER, min=-50, max=50, key=at_reset
  event = var=DESIRED_RUDDER, val=$[RANDOM_RUDDER]
  event = var=DESIRED_THRUST, val=30

  reset_max   = unlimited
  reset_time  = end
}
```

[1] par exemple, la documentation de `pMarineViewer` est accessible en se rendant sur `http://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=Tools.Cover` ou en entrant l'une des lignes de commande :
- `pMarineViewer -h` (pour un résumé)
- `pMarineViewer -i` (pour l'interface)
- `pMarineViewer -e` (pour un exemple de `.moos`)

[2] ce point est défini par des coordonnées géographiques (latitude, longitude) qui correspondront à l'origine du repère cartésien du robot. Ce point se trouve sur la Place de l'Atlantide de Brest.

[3] les rapports sont générés dans la méthode `buildReport()` vue pendant le TP 1.