Assignment 7

The purpose of this assignment is to experiment with *Behavior Based Robotics*. You will learn how to realize intelligent emergent robotic behavior by combining several small, concurrent behaviors using a priority based arbiter.

# Learning goal

This assignment is specifically targeted at following learning goal:

* write a ‘Behavior Based’ robotic application, comprising a priority-based arbiter and various simple behaviors

# Getting ready

Experiment with the online *BSim* simulator which you can find on: <http://www.behaviorbasedprogramming.net/> . To use it you should read the documentation at: <http://www.behaviorbasedprogramming.net/manual.html>.

For your convenience we have prepared a semi-complete *stage\_behavior* package. This package includes the source code of two ready-to-use behavior nodes (*escape\_behavior, cruise\_behavior*) and a ready-to-use launch files (*behavior.launch*). Also the *CMakeLists.txt* file is ready-to-use. The *arbiter* node however is not working yet, although some basic source is provided. Carefully study all source and launch files!

In previous assignment you already cloned the *mines-ros* directory from GitHub into your catkin workspace. To be sure you have the latest version of everything, in particular of the *stage\_behavior* package enter following commands:

* *cd ~/catkin\_ws/src*
* *git pull*
* *cd ~/catkin\_ws*
* *catkin\_make*

# The Tasks

## Arbiter

First task is to write a fully functional, priority based *arbiter* node. When this node is working correctly you will see the simulated robot freely cruising through the simulated Stage world, while escaping from walls.



Following requirements must be met:

* The Arbiter must subscribe to at least 3 input topics, named *cmd\_vel0*, *cmd\_vel1*, *cmd\_vel2.* On these topics it must be able to receive *geometry\_msgs/Twist* messages
* The Arbiter must regularly publish *geometry\_msgs/Twist* messages on the topic *cmd\_vel.* The *Twist* messages which it publishes on *cmd\_vel* must be derived from the input messages received on its input topics.
* The arbiter must use priority based arbitration where the topic *cmd\_vel0* should be treated with highest priority, *cmd\_vel1* with the next highest priority and *cmd\_vel2* with lowest priority, etcetera.
* The arbiter should not know which behavior is connected to which input topic. You must not use any information about the behavior configuration in the arbiter code.

## Goal seeking behavior

Second task is to write a *seek\_behavior* node to extend the cruise and escape behaviors. This node must make the robot seek a goal position and stop whenever this position is reached. How to do this goal seeking is up to you.

The following requirements and constraints must be met:

* The node should be able to receive a new goal as a *geometry\_msgs/PoseStamped* message via */move\_base\_simple/goal*.
* The node is allowed to use the TF system, the actionlib and all available topics
* It is not allowed to use the ROS navigation stack (i.e. the move\_base node).
* It is also not allowed to give this node a predefined path or to give it a map or explicit information about obstacles.
* It is allowed to make modifications to all existing behavior and launch files, but these modifications must be clearly explained.

Hints:

* Goal can be set via the *2D Nav Goal button* of rviz or via *rostopic pub* command: <http://answers.ros.org/question/47973/publishing-to-move_base_simplegoal/>
* A nice example goal position to test is: (5.0, 6.0, 0.0).
* For some goal seeking idea’s look for example here:
  + <http://www.reocities.com/pentagon/barracks/9722/downloads/ControlAUV.pdf>

# What to Submit

Your submission for this assignment will have two parts:

1. A zip file containing your package.
2. A document which explains how everything works and describes how to run it. The document should also describe your observations.