ROS – Lab 3

Goal
Learn to create your own node starting from the skeleton node.

Content
In this exercise, we simulate the « dog follows master » behavior. In short, the dog moves by always heading at the current position of the master. We assume here that the dog always matches the speed of his master. In the the last part, we assume that the dog has a puppy, and that the puppy follows, not the master, but the dog (say the dog is it's mother, so the puppy is more interested in the dog than in the master), also matching the speed of the dog.

Remark:
Contrary to lab 1, the emphasis is not on optimizing the remappings. First because we will launch very few nodes, and second because there is no simple way to make things fully automatic for the last task of this lab. In short, don't bother! But you will have to do some remapping for the the last task.

Task 1: Setting up the environment.

- Get rid of the packages of lab 1 (store there somewhere, don't delete them permanently). To force a clean start, delete folders «`devel`» and «`build`» of your catkin workspace.
- Prepare your environment. I suggest you use two terminals. In one of them, open a tab to run the `roscore`, a tab to run `rviz` (leave those two running permanently), a tab to compile and run your software.
- Launch the given launch file and configure `rviz` to display markers. You should see a blue cube moving along a circular path.

Task 2: Modifying the « master » node.
Check the code of the « master » node, « `master.cpp` ».
You can ignore the code relative to publishing markers for rviz.
The subscription to `/key_typed` and the corresponding callback function are not used in the initial tasks, so you can ignore them for the time being.
Modify the code of « `master.cpp` » so the node publishes:

- The position of the master (`geometry_msgs/Point` with z=0).
- Its speed (`std_msgs/Float64`).

When you're done, submit to Hippocampus (exercise 3) a zip file containing the full lab (both packages). Name the file "task2.zip". Include in the zip file an image of two terminals showing the echos of the two topics.

Task 3: Creating the « dog » node.
Write the « `dog.cpp` » code so that it implements the above mentioned behavior **for a constant speed of the master and the dog**.
The node should subscribe to the appropriate topic and publish the position of the dog and the speed of the dog3.
Remember that if the master is at A and the dog at B, then the direction of motion of the dog is the unit vector $u = \frac{\vec{BA}}{||\vec{BA}||}$. You just need to multiply u by $v\Delta t$, where v is the speed of the dog and Δt the time elapsed since last update. The code of `master.cpp` shows you how to handle the time elapsed.

In this part, you set the initial position of the dog in the initialization instructions of the main file to (0,0,0). You also set the speed of the dog to 1.0 and the speed of the master will not be altered when the application is run (stays at the initial value 1.0).
Use the example of « `master.cpp` » to publish a new marker for the dog. You will just need to change the color.
Once your application runs, submit to Hippocampus a zip file containing the full lab (both packages). Name the file “task3.zip”. If you have the proper software to record the screen of your PC, include a (short!) video of the result. Preferably record only the relevant region of the screen.

Task 4: Adding support for varying master speed.
Modify the « dog » code so it can adapt to variations of the speed of the master, always matching it. Implement the modification, run the modified application and submit “task4.zip” to Hippocampus, same instructions.

Task 5: Adding parameters to the « dog » node.
Add support for parameters in the “dog” node to allow specifying the dog’s initial position. Submit as “task5.zip”, with the same instructions as earlier.

Task 6: Adding the puppy.
Run an application with an additional puppy which always heads to the dog, at the same speed as the dog. Submit as “task6.zip”.