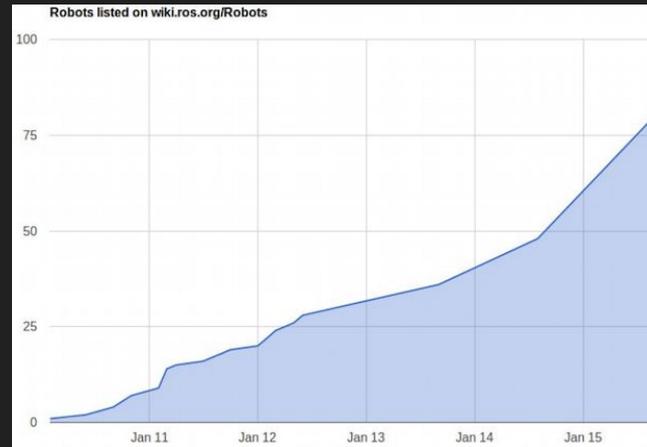


Robotic Manipulation Exercise 1

Introduction to ROS, mujoco, and git

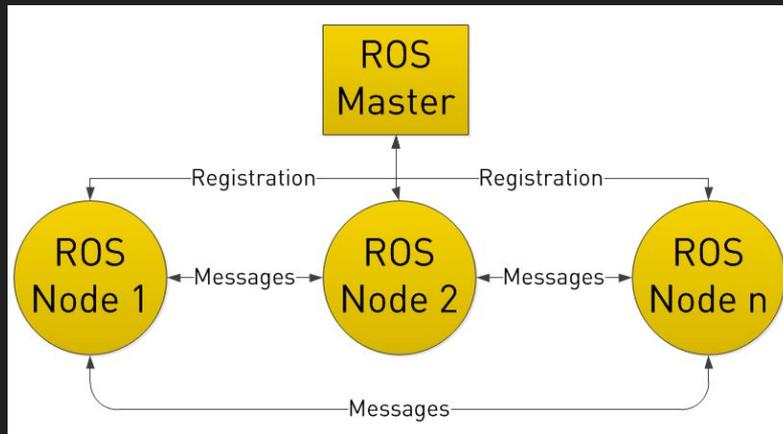
Robotic Operating System

- ROS stand for Robotic Operating System and was released 2007 by a company known as Willow Garage.
- ROS is an open-source, meta-operating system for your robot.
- ROS is designed to be modular at a fine-grained scale.
- ROS is widely used in industry and academic research



ROS concept

- ROS is build up of nodes
- ROS nodes are registered through a ROS Master
- Nodes can communicate with each other via topics
- For more in depth knowledge about ROS you can read, for example, <http://wiki.ros.org/ROS/Introduction>



ROS Workspace Environment

- Default workspace loaded with:

```
$ source /opt/ros/melodic/setup.zsh
```

- Setup ROS workspace (http://wiki.ros.org/catkin/Tutorials/create_a_workspace)

```
$ mkdir -p ~/ros/src
```

```
$ cd ~/ros
```

```
$ catkin_make
```

- Always remember to `source devel/setup.bash` or `devel/setup.zsh` in your workspace after you compiled the code in order to access the newly compiled ROS nodes.

Install MuJoCo

Download MuJoCo simulator from <http://www.mujoco.org> and put the simulator code as well as your MuJoCo key into the folder `~/mujoco/`, i.e.: (you can find the “mjkey.txt” file in “MyCourses > For Aalto users”)

```
$ ls ~/.mujoco
```

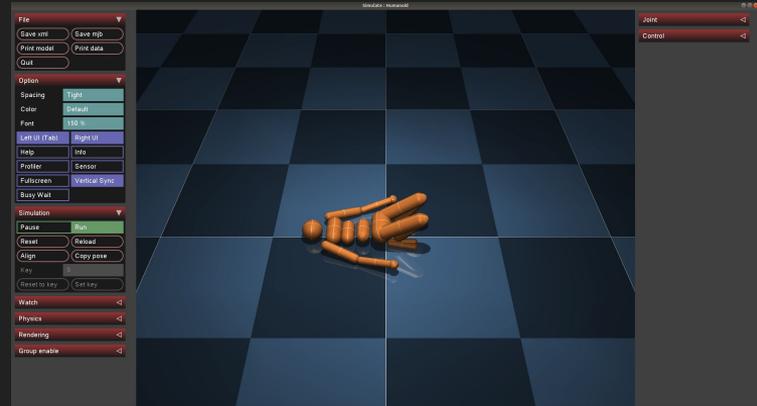
```
  mjkey.txt  mjpro200
```

```
$ ls ~/.mujoco/mjpro200/
```

```
  bin  doc  include  model  sample
```

Inside `~/mujoco` folder run:

```
$ ~/.mujoco/mjpro200/bin/simulate ~/.mujoco/mjpro200/model/humanoid.xml
```



Test ROS

Use three terminals

- First terminal (roscore)

```
$ source ~/ros/devel/setup.zsh
```

```
$ roscore
```

- Second terminal (publisher)

```
$ source ~/ros/devel/setup.zsh
```

```
$ rostopic pub -r 1 /course_name std_msgs/String "data: 'manipulation_course'"
```

- Third terminal (subscriber)

```
$ source ~/ros/devel/setup.zsh
```

```
$ rostopic list #should print three topics
```

```
$ rostopic echo /course_name
```

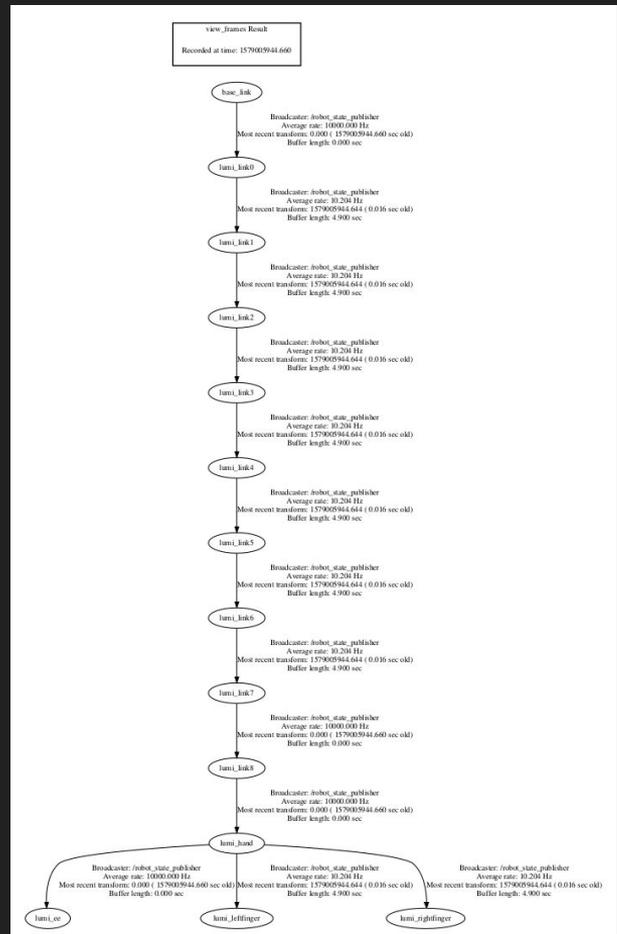
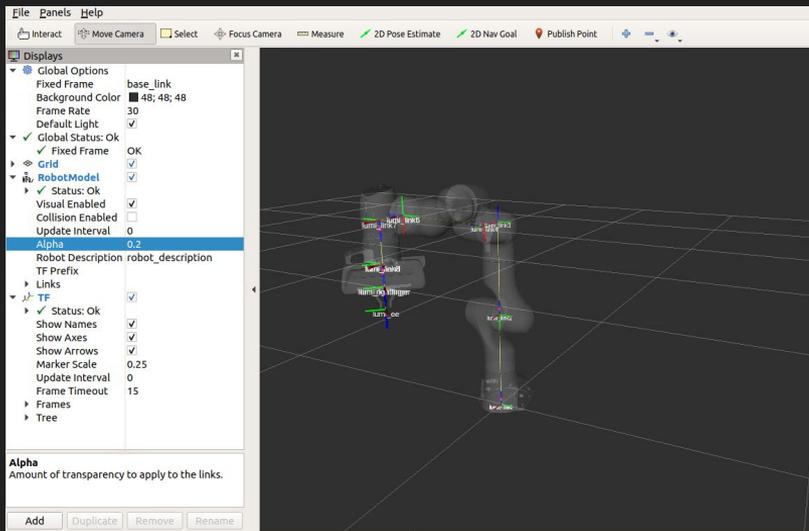
TF

- A robotic system typically has many 3D coordinate frames that change over time. These coordinate systems are naturally expressed in a transformation (TF) tree → <http://wiki.ros.org/tf2>
- tf maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.

RViz and TF tree

```
$ roslaunch lumi_description show.launch
```

```
$ rosrn tf view_frames && evince frames.pdf
```



Git

- git is a version-control system.
- In this course, gitlab is used for storing all exercises. If you have no previous knowledge of git and/or gitlab then please read up about it online at, e.g. <https://docs.gitlab.com/ee/gitlab-basics/>
- To use Aalto gitlab you need to log in to <https://version.aalto.fi/> and then set up your ssh key <https://docs.gitlab.com/ee/gitlab-basics/create-your-ssh-keys.html>
- Or follow these two links [here](#) and [here](#).

gitlab group, forking the course material, and pushing code

- Interactive session during the exercise session.
- For the gitlab repository, we created one subgroup for each one of you. You can use the following pattern to access that:

https://version.aalto.fi/gitlab/robotic_manipulation_students_projects_2021/<your email address without @aalto.fi>

for example if your email address is eshagh.kargar@aalto.fi use:

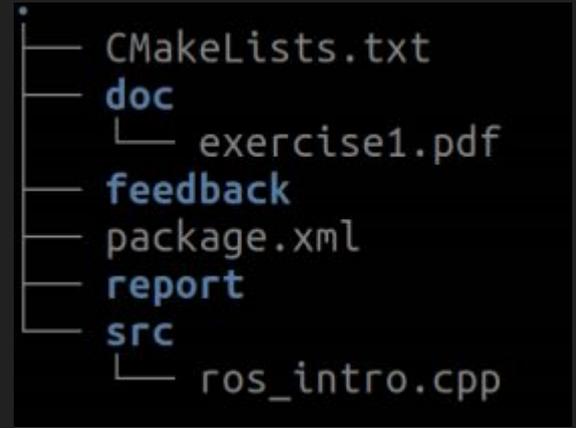
https://version.aalto.fi/gitlab/robotic_manipulation_students_projects_2021/eshagh.kargar

- On your computer, remember to always clone ***your*** newly forked exercise repository into the src directory of your ROS workspace

Exercise file system

The file system for each exercise is visualized in the figure to the right

- The src folder contains the template code you need to fix
- The feedback folder will contain the TA's feedback and points awarded
- In the report folder you will upload the exercise report as a pdf
- The docs folder will contain all necessary information for the current exercise.
- Other files are ROS specific which you do not need to touch.



What did we not cover?

- Specifically to ROS, we did not cover concepts such as:
 - ROS Services <http://wiki.ros.org/Services>,
 - ROS Parameter Server <http://wiki.ros.org/Parameter>,
 - ROS Bags <http://wiki.ros.org/Bags>,
 - and much more <http://wiki.ros.org/ROS/Concepts>.
- With respect to Git we did not cover concepts such as
 - Git Branching and Merging <https://git-scm.com/book/en/v2/>
 - Git-Branching-Basic-Branching-and-Merging
 - git-diff <https://git-scm.com/docs/git-diff>
 - and much more <http://thepilcrow.net/explaining-basic-concepts-git-and-github/>
- You will probably not need to master nor need these concepts during the course, but it is good to know about them.