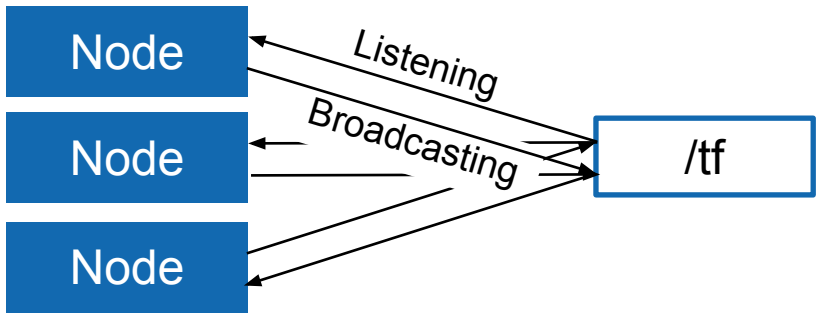
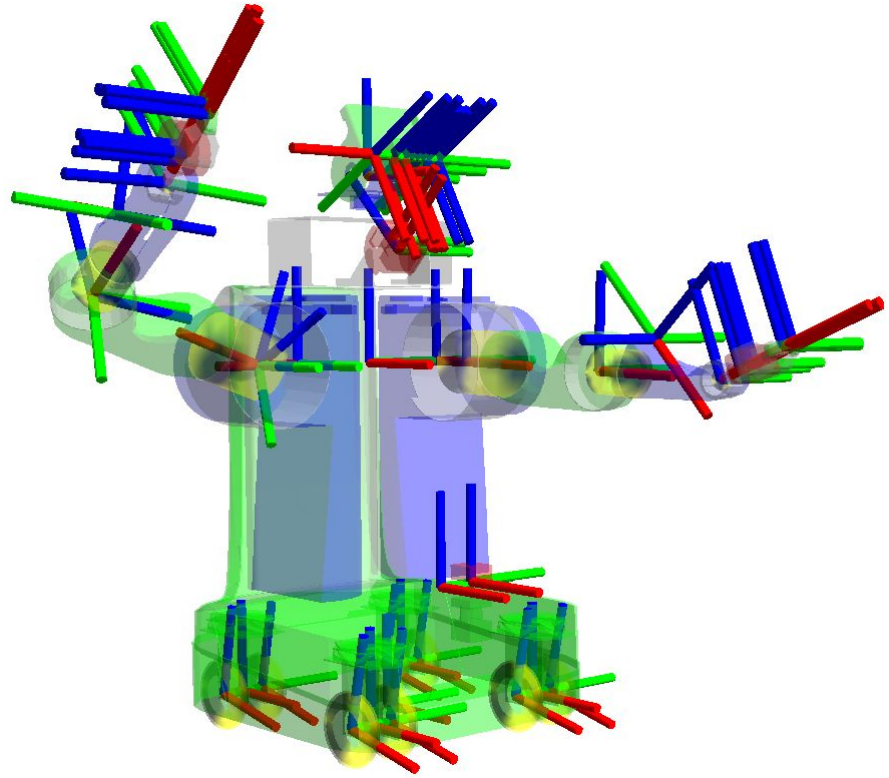
# TF Transformation System

- Tool for keeping track of coordinate frames over time
- Maintains relationship between coordinate frames in a tree structure buffered in time
- Lets the user transform points, vectors, etc. between coordinate frames at desired time
- Implemented as publisher/subscriber model on the topics `/tf` and `/tf_static`



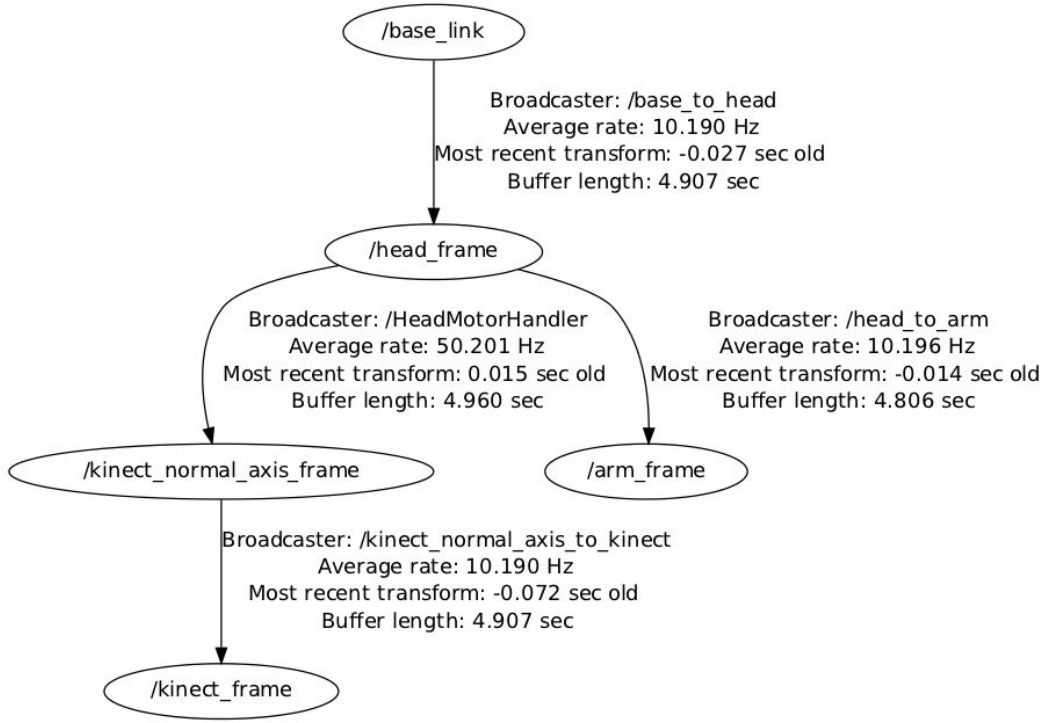
**More info**
http://wiki.ros.org/tf2

# TF Transformation System
## Transform Tree

- TF listeners use a buffer to listen to all broadcasted transforms
- Query for specific transforms from the transform tree

*tf2_msgs/TFMessage.msg*

```
geometry_msgs/TransformStamped[] transforms
  std_msgs/Header header
    uint32 seqtime stamp
    string frame_id
  string child_frame_id
  geometry_msgs/Transform transform
    geometry_msgs/Vector3 translation
    geometry_msgs/Quaternion rotation
```



# TF Transformation System

## Tools

### Command line

Print information about the current transform tree

```
> rosrun tf tf_monitor
```

Print information about the transform between two frames

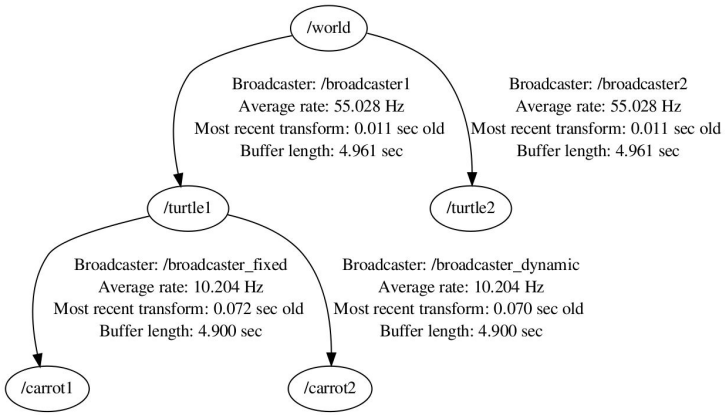
```
> rosrun tf tf_echo
    source_frame target_frame
```

### View Frames

Creates a visual graph (PDF) of the transform tree. Broken at the moment!!!!!
https://github.com/ros/geometry/pull/222

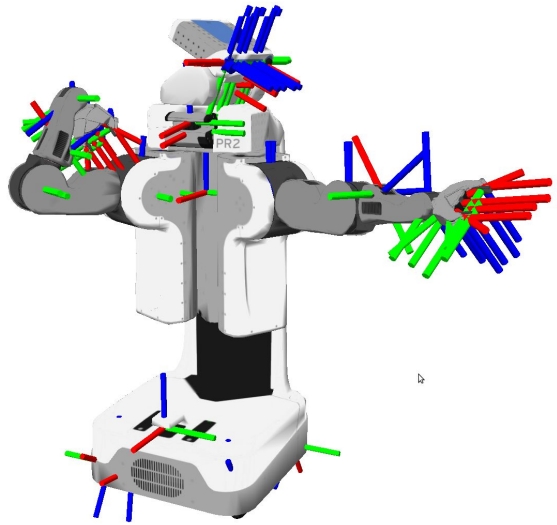
```
rosrun tf view_frames
```

```
rosrun tf2_tools view_frames.py
```



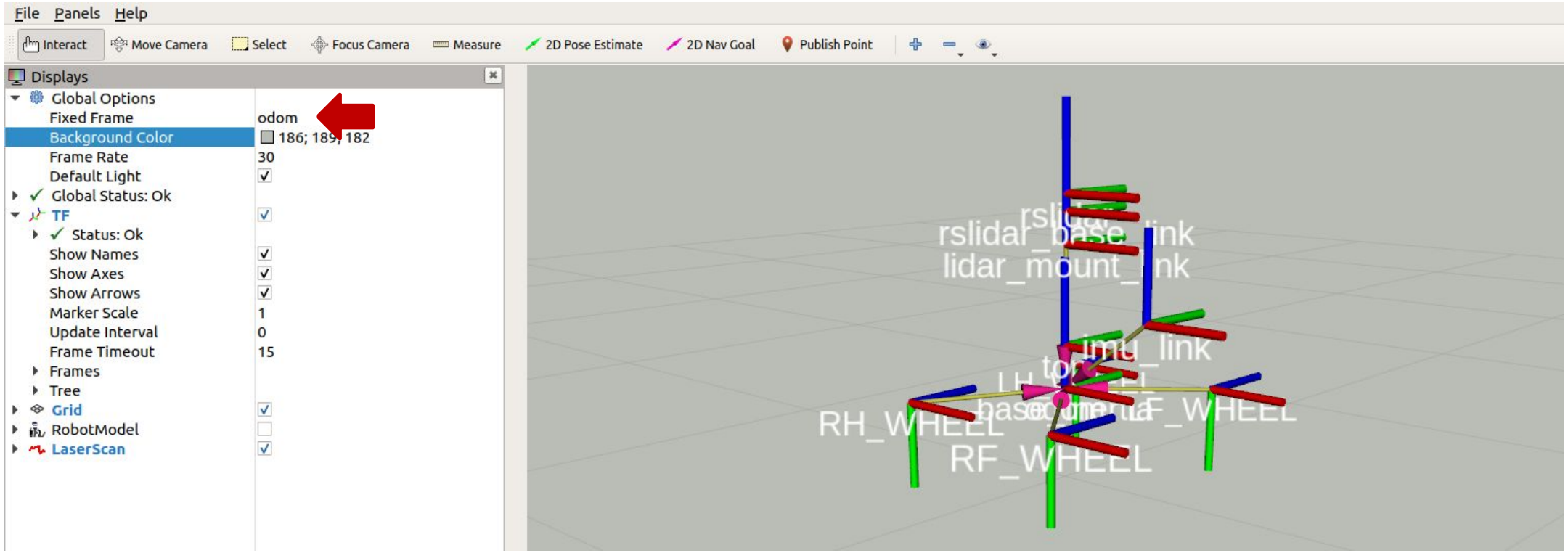

### RViz

3D visualization of the transforms



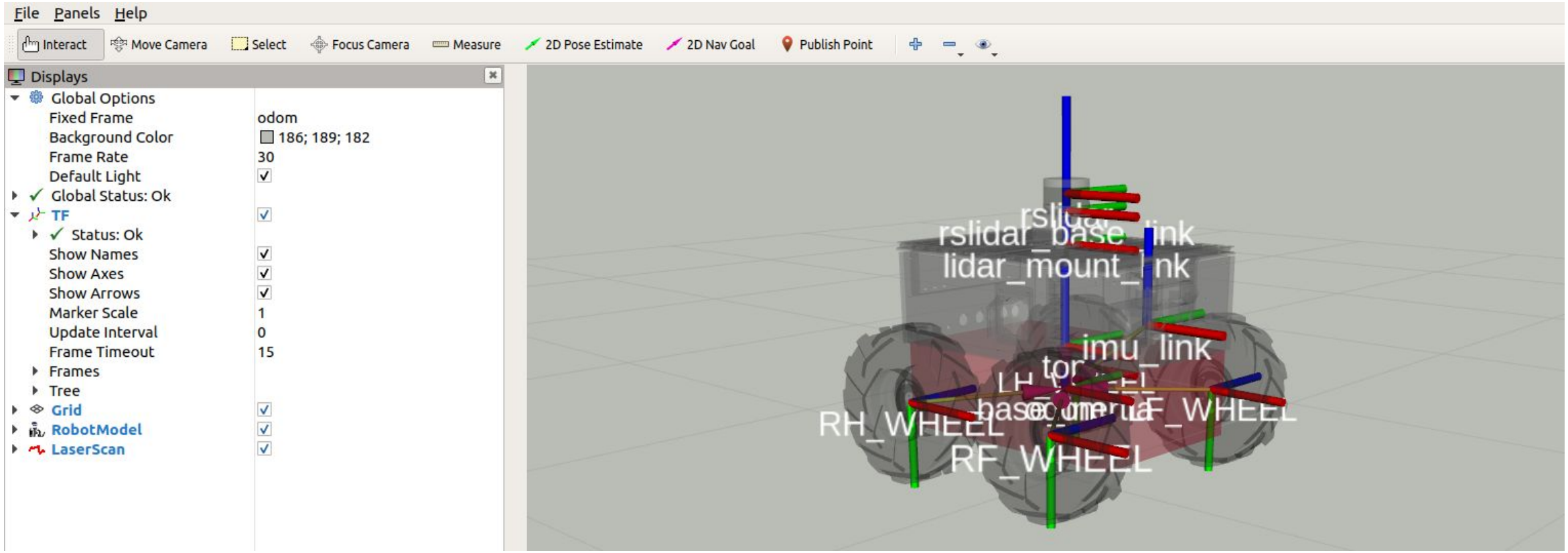
# TF Transformation System

## RViz Plugin



# TF Transformation System

## RViz Plugin



# TF Transformation System
## Transform Listener C++ API

- Create a TF listener to fill up a buffer

```
tf2_ros::Buffer tfBuffer;
tf2_ros::TransformListener tfListener(tfBuffer);
```

- Make sure, that the `listener` does not run out of scope!
- To lookup transformations, use

```
geometry_msgs::TransformStamped transformStamped =
tfBuffer.lookupTransform(target_frame_id,
                         source_frame_id, time);
```

- For `time`, use `ros::Time(0)` to get the latest available transform

```
#include <ros/ros.h>
#include <tf2_ros/transform_listener.h>
#include <geometry_msgs/TransformStamped.h>

int main(int argc, char** argv) {
  ros::init(argc, argv, "tf2_listener");
  ros::NodeHandle nodeHandle;
  tf2_ros::Buffer tfBuffer;
  tf2_ros::TransformListener tfListener(tfBuffer);

  ros::Rate rate(10.0);
  while (nodeHandle.ok()) {
    geometry_msgs::TransformStamped transformStamped;
    try {
      transformStamped = tfBuffer.lookupTransform("base",
                                "odom", ros::Time(0));
    } catch (tf2::TransformException &exception) {
      ROS_WARN("%s", exception.what());
      ros::Duration(1.0).sleep();
      continue;
    }
    rate.sleep();
  }
  return 0;
};
```

**More info**

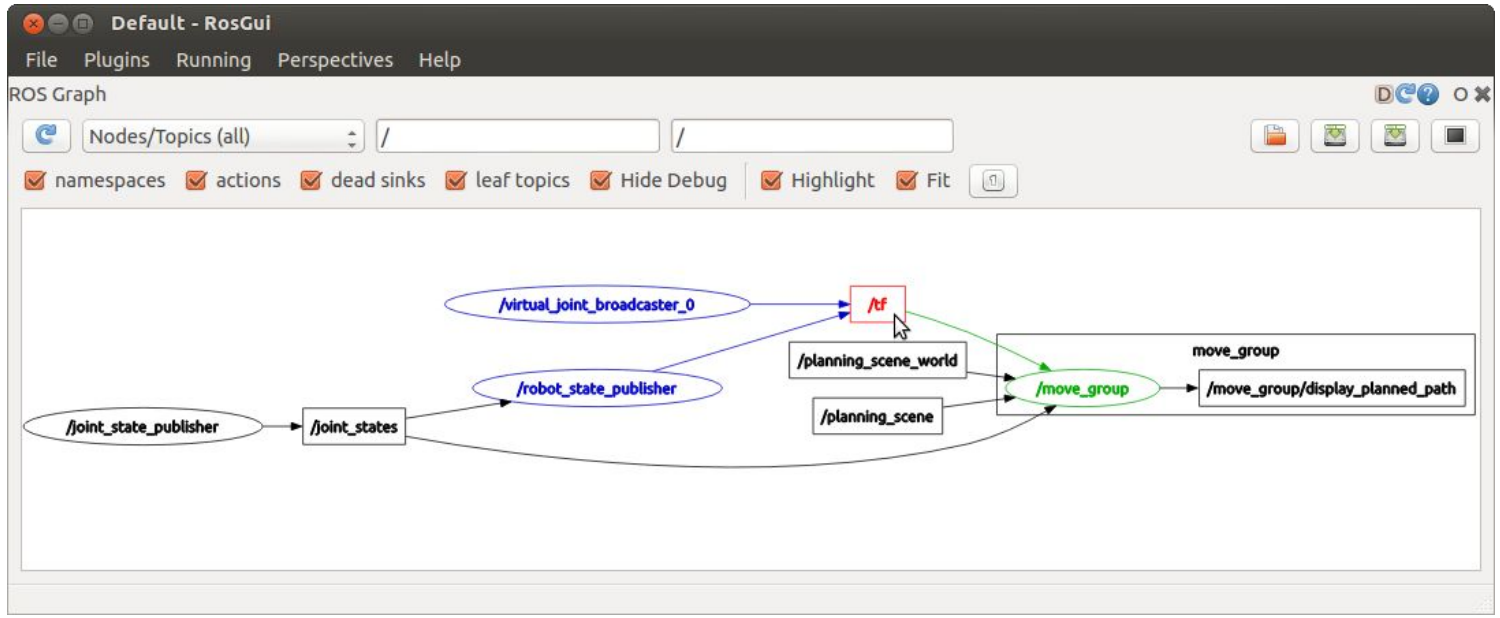
http://wiki.ros.org/tf2/Tutorials/Writing%20a%20tf2%20listener%20%28C%2B%2B%29

# rqt User Interface
## rqt_graph

- Visualizing the ROS computation graph

Run *rqt_graph* with

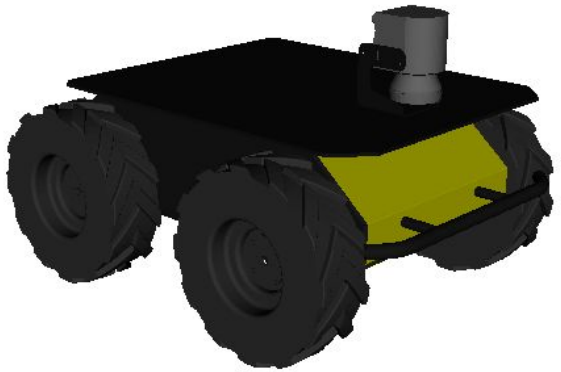
```
> rosrun rqt_graph rqt_graph
```



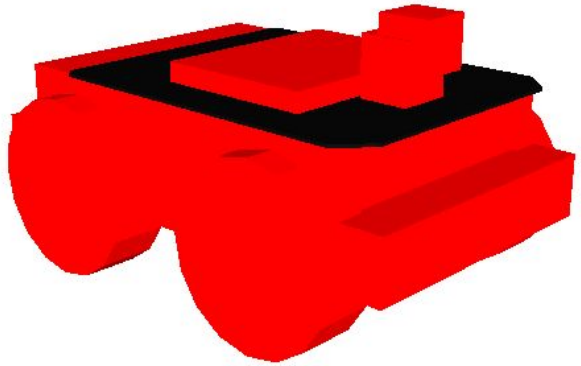

**More info**
http://wiki.ros.org/rqt_graph

# Robot Models
## Unified Robot Description Format (URDF)

- Defines an XML format for representing a robot model
  - Kinematic and dynamic description
  - Visual representation
  - Collision model
- URDF generation can be be scripted with *XACRO*



Mesh for visuals

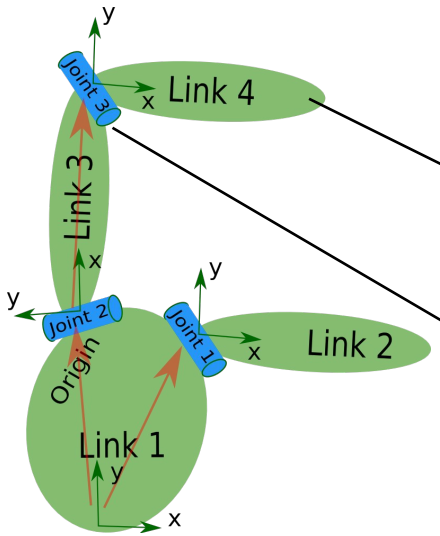


Primitives for collision

**More info**
http://wiki.ros.org/urdf
http://wiki.ros.org/xacro

# Robot Models
## Unified Robot Description Format (URDF)

- Description consists of a set of *link* elements and a set of *joint* elements
- Joints connect the links together




*robot.urdf*

```
<robot name="robot">
  <link> ... </link>
  <link> ... </link>
  <link> ... </link>

  <joint> .... </joint>
  <joint> .... </joint>
  <joint> .... </joint>
</robot>
```

```
<link name="link_name">
  <visual>
    <geometry>
      <mesh filename="mesh.dae"/>
    </geometry>
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="10"/>
    <inertia ixx="0.4" ixy="0.0" .../>
  </inertial>
</link>
```

```
<joint name="joint_name" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort="1000.0" upper="0.548" ... />
  <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
  <parent link="parent_link_name"/>
  <child link="child_link_name"/>
</joint>
```

**More info**
http://wiki.ros.org/urdf/XML/model

# Robot Models
## Usage in ROS

- The robot description (URDF) is stored on the parameter server (typically) under `/robot_description`
- You can visualize the robot model in Rviz with the  *RobotModel* plugin

*control.launch*

```
...
<include file="$(find smb_description)/launch/load.launch">
   <arg name="simulation"       value="$(arg simulation)"/>
   <arg name="description_name" value="$(arg robot_description)"/>
   <arg name="description_file" value="$(arg description_file)"/>
   <arg name="wheel_joint_type" value="continuous"/>
   <arg name="robot_namespace"  value="$(arg robot_namespace)"/>
 </include>
...
```

*load.launch*

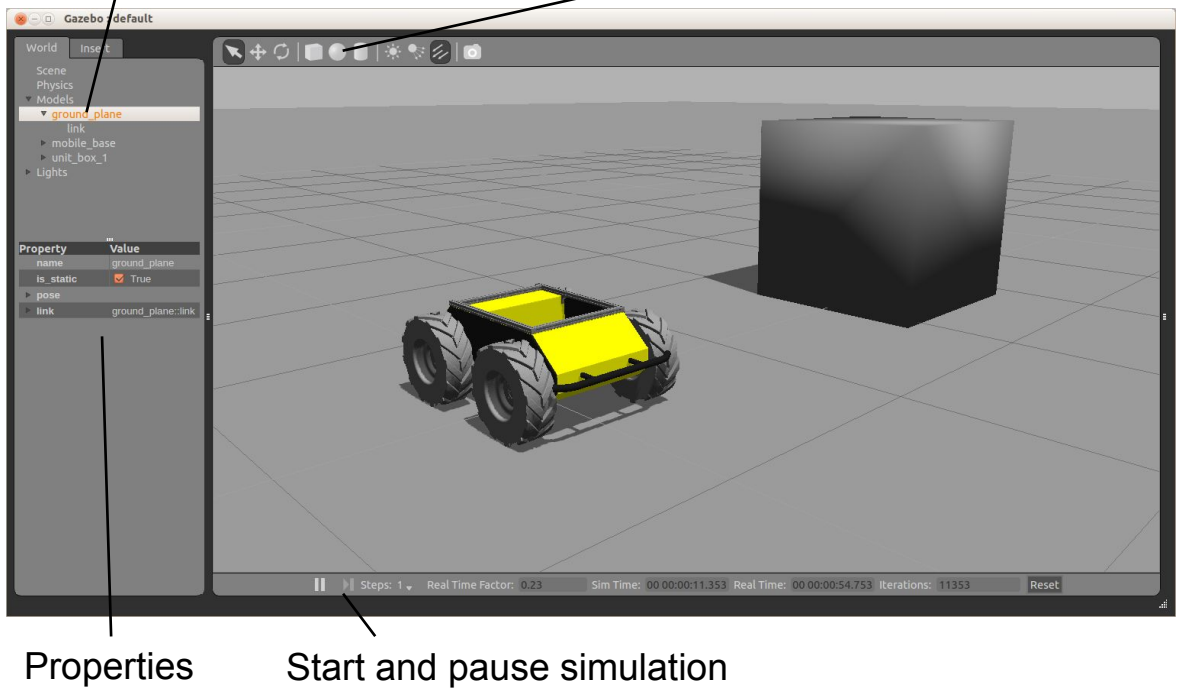
```
...
 <param name="$(arg description_name)" command="$(find xacro)/xacro
   $(arg description_file)
   wheel_joint_type:=$(arg wheel_joint_type)
   simulation:=$(arg simulation)
   robot_namespace:=$(arg robot_namespace)
   lidar:=$(arg lidar)
   description_name_xacro:=$(arg description_name)
   publish_tf:=$(arg publish_tf)"/>
</launch>
...
```

# Gazebo Simulator

- Simulate 3d rigid-body dynamics
- Simulate a variety of sensors including noise
- 3d visualization and user interaction
- Includes a database of many robots and environments (*Gazebo worlds*)
- Provides a ROS interface
- Extensible with plugins

Run Gazebo with

```
> rosrun gazebo_ros gazebo
```




**More info**
http://gazebosim.org/
http://gazebosim.org/tutorials

# Simulation Descriptions
## Simulation Description Format (SDF)

- Defines an XML format to describe
  - Environments (lighting, gravity etc.)
  - Objects (static and dynamic)
  - Sensors
  - Robots
- SDF is the standard format for Gazebo
- Gazebo converts a URDF to SDF automatically

**More info**
http://sdformat.org

# Further References

- **ROS Wiki**
  - http://wiki.ros.org/
- **Installation**
  - http://wiki.ros.org/ROS/Installation
- **Tutorials**
  - http://wiki.ros.org/ROS/Tutorials
- **Available packages**
  - http://www.ros.org/browse/
- **ROS Cheat Sheet**
  - https://www.clearpathrobotics.com/ros-robot-operating-system-cheat-sheet/
  - https://kapeli.com/cheat_sheets/ROS.docset/Contents/Resources/Documents/index
- **ROS Best Practices**
  - https://github.com/leggedrobotics/ros_best_practices/wiki
- **ROS Package Template**
  - https://github.com/leggedrobotics/ros_best_practices/tree/master/ros_package_template