Zewail City of Science and Technology

ZEWAIL CITY
ESTABLISHED 2000
INAUGURATED 2011

مدينة زويل للعلوم والتكنولوجيا

Space and Communications Engineering - Autonomous Vehicles Design and Control - Fall 2016

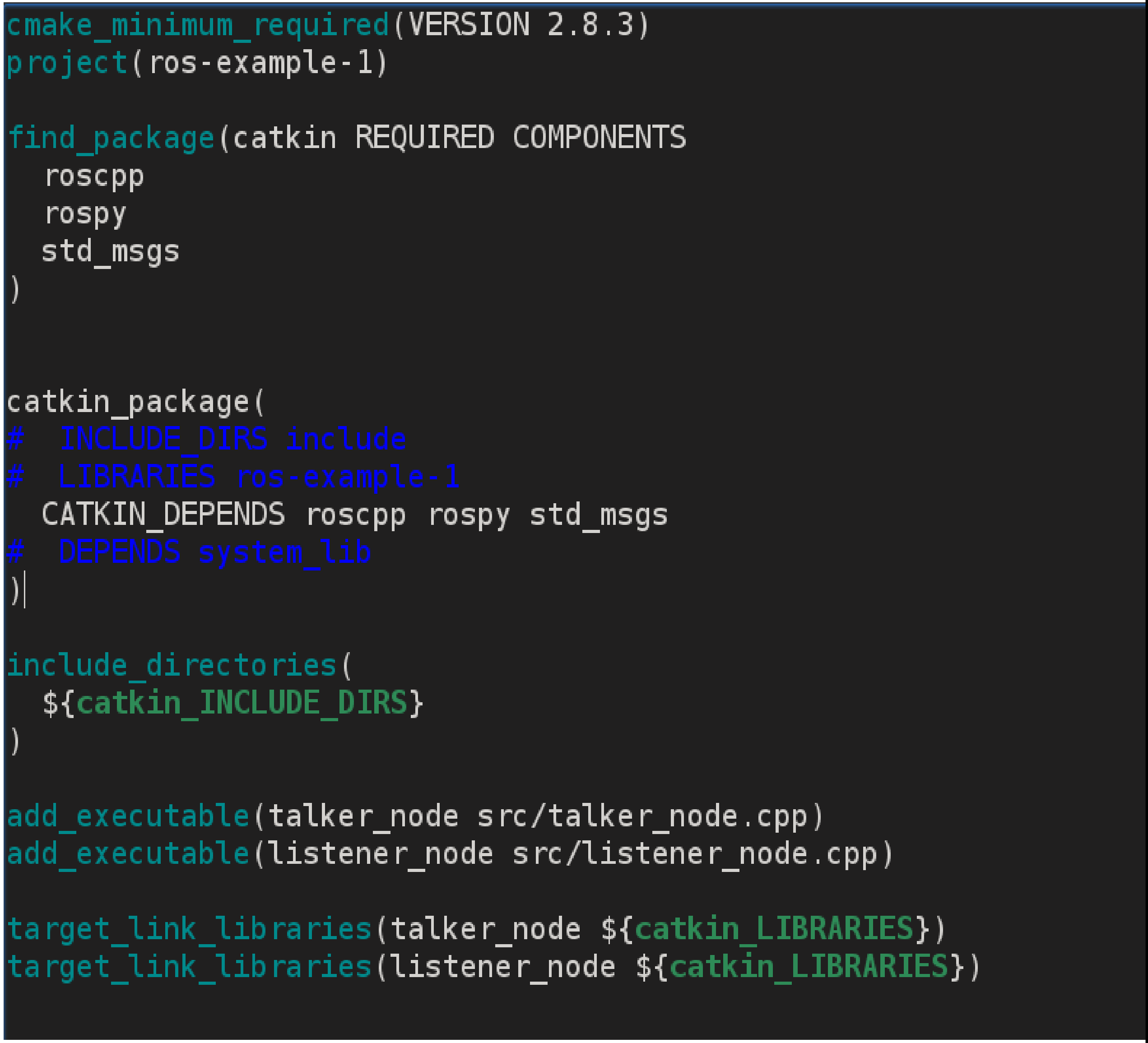
Setting UP CATBot Simulation Environment

Mahmoud Abdul Galil

Tutorial-3, Tuesday October 4th, 2016

ros-example-1 Cont'd: CMakeLists.txt

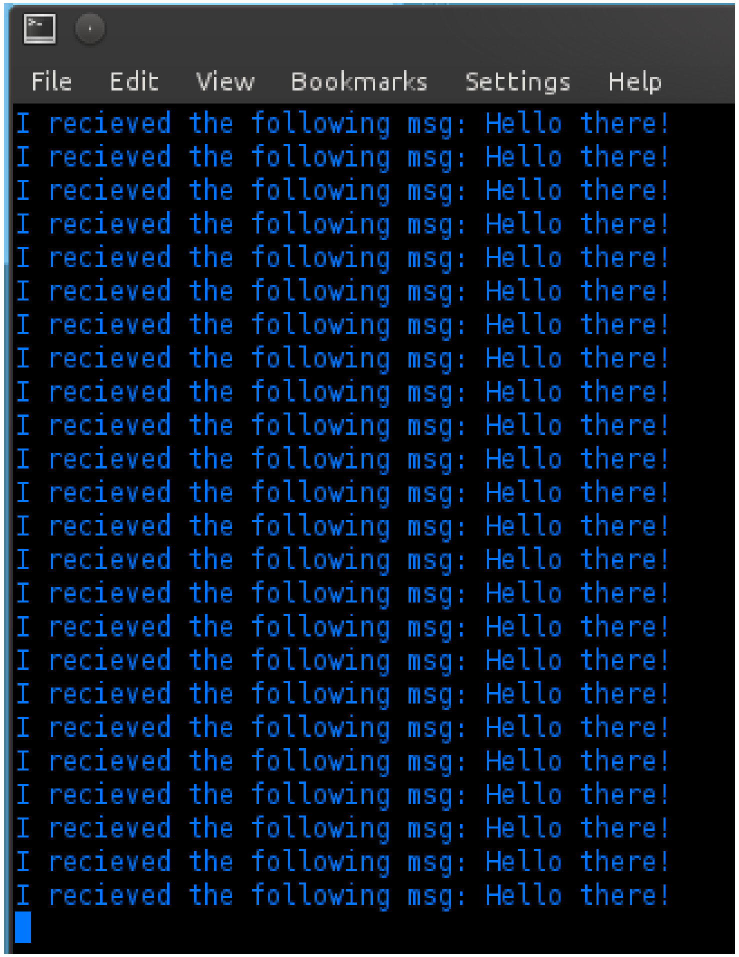
- We wrote our desired codes in src folder
- Now is time to inform the build system to include these codes while building our workspace.
- This is done by modifying the CMakeLists.txt file.
- Excluding documentation comments, the new CmakeLists.txt will look like the picture on the right
- Notice that we added two build targets as executables using **add_executable( )**, and linked catkin_LIBRARIES to then using **target_link_libraries( )**.



```
cmake_minimum_required(VERSION 2.8.3)
project(ros-example-1)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
)


catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES ros-example-1
  CATKIN_DEPENDS roscpp rospy std_msgs
#  DEPENDS system_lib
)

include_directories(
  ${catkin_INCLUDE_DIRS}
)

add_executable(talker_node src/talker_node.cpp)
add_executable(listener_node src/listener_node.cpp)

target_link_libraries(talker_node ${catkin_LIBRARIES})
target_link_libraries(listener_node ${catkin_LIBRARIES})
```

Let's Compile &Run the nodes :D

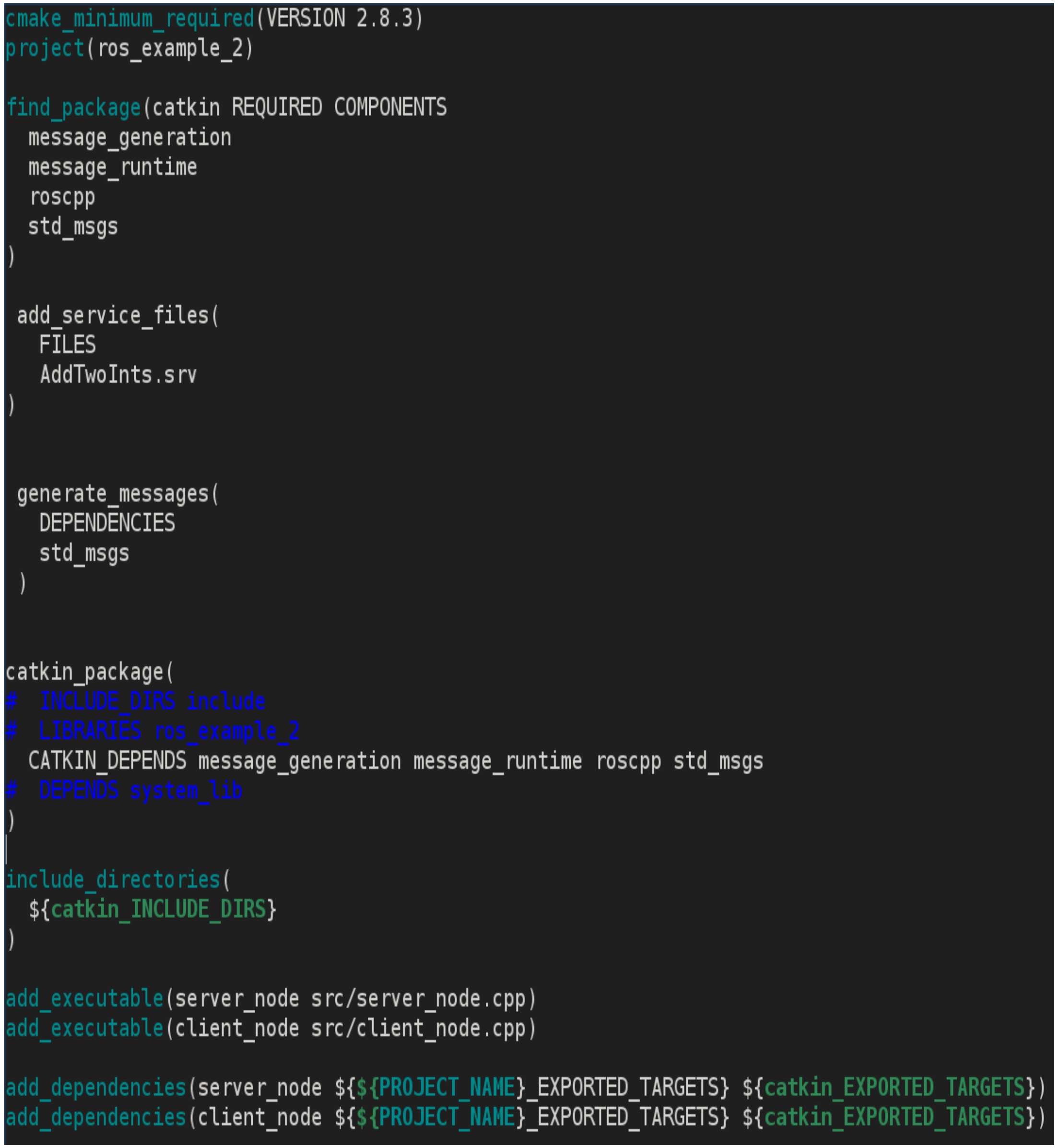
- catkin_make
- source devel/setup.bash
- rosrun ros-example-1 talker_node
- Open a new terminal
- source devel/setup.bash
- rosrun ros-example-2 listener_node



```
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
I recieved the following msg: Hello there!
```

ros_example_2: Overview

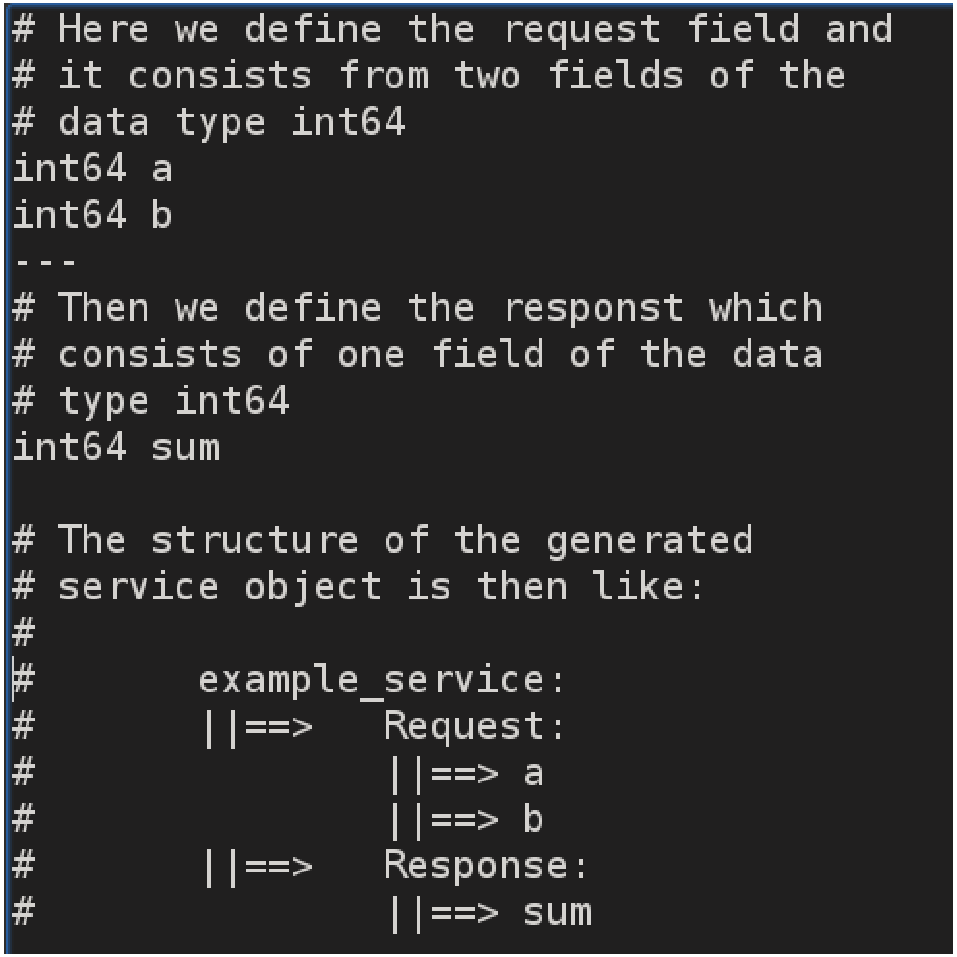
- The second example is about remote procedure call in ROS
- First we create the package using catkin_create_pkg ros_example_2 roscpp std_msgs
- To define a service, we need to add build and run dependencies for the two packages *message_generation* and *message_runtime*, in the manifest file (package.xml)
- We define our service in AddTwoInts.srv file in srv folder
- Next we edit the CMakeLists.txt to:
    - Generate the header files from the service definition
    - Add dependency on the exported targets (headers defining the service) for the project
- Next we add two c++ files, one for a server node and the other for a client node.
- Finally we edit the CMakeLists.txt file to compile the server and client nodes from their respective c++ files.



```
cmake_minimum_required(VERSION 2.8.3)
project(ros_example_2)

find_package(catkin REQUIRED COMPONENTS
  message_generation
  message_runtime
  roscpp
  std_msgs
)

 add_service_files(
   FILES
   AddTwoInts.srv
)


 generate_messages(
   DEPENDENCIES
   std_msgs
 )


catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES ros_example_2
  CATKIN_DEPENDS message_generation message_runtime roscpp std_msgs
#  DEPENDS system_lib
)

include_directories(
  ${catkin_INCLUDE_DIRS}
)

add_executable(server_node src/server_node.cpp)
add_executable(client_node src/client_node.cpp)

add_dependencies(server_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
add_dependencies(client_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
```

ros_example_2 Cont'd: AddTwoInts.srv

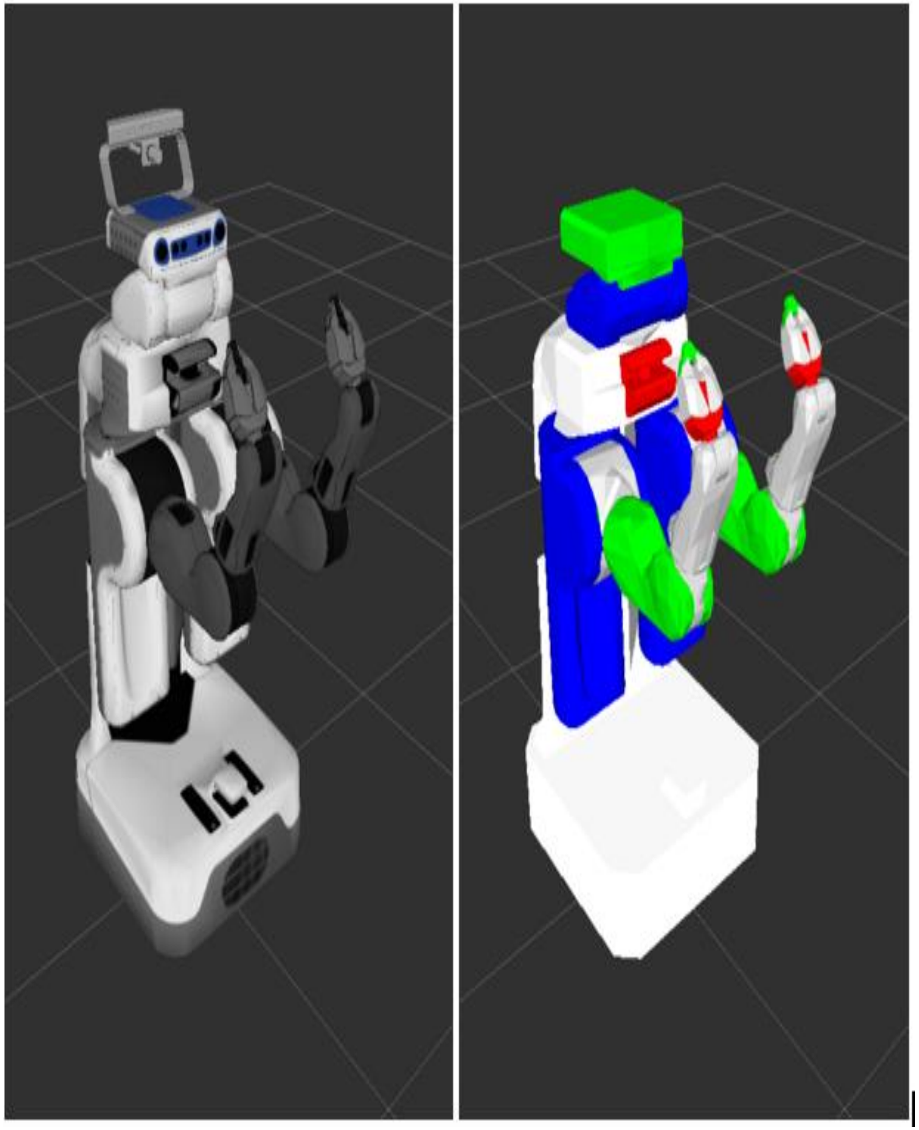
- The .srv file is used by message_generation package to generate headers to define services
- The headers must be included in the files we wish to use the defined service in
- Try to have a look on one of the generated header files to have more insight (you can find them inside the devel/include/ros_example_2 directory)



```
# Here we define the request field and
# it consists from two fields of the
# data type int64
int64 a
int64 b
---
# Then we define the respone which
# consists of one field of the data
# type int64
int64 sum

# The structure of the generated
# service object is then like:
#
#       example_service:
#       ||==>   Request:
#               ||==> a
#               ||==> b
#       ||==>   Response:
#               ||==> sum
```

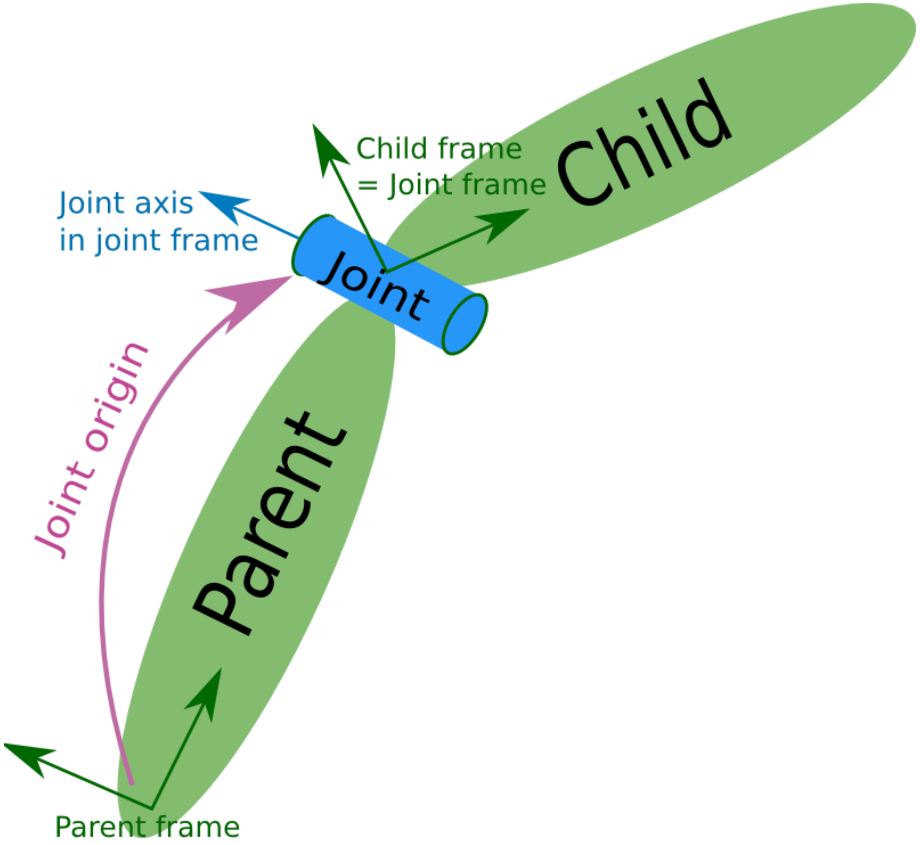
Universal Robot Description Format

- We need to have a model that describes the configuration of the robot
- In ROS, URDF is the official description format for robot models
- URDF is an XML-based language.
- URDF models are used with other ROS packages to obtain important run-time information about robots
- URDF is used in gazebo for simulation purposes
- URDF is combined with xacro to accelerate the process of writing big and complex XML files
- There is a SolidWorks plugin to export a SolidWorks model as a URDF description package



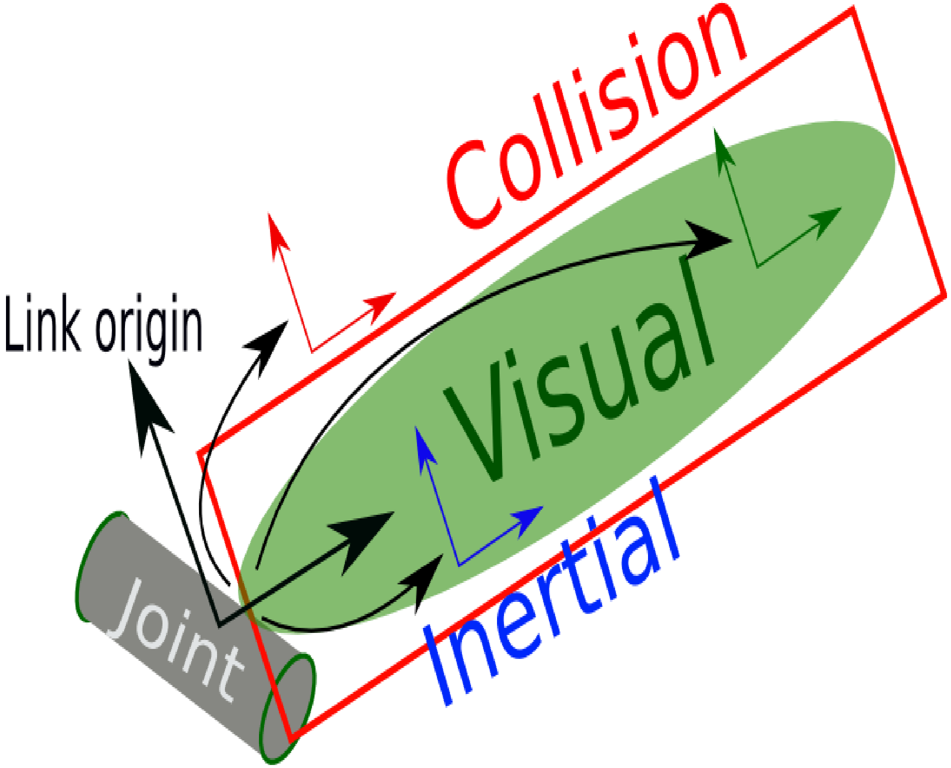
Universal Robot Description Format cont'd

- A robot is modeled in ROS as a **tree-like** structure of non-deforming components.
- The non-deforming parts (also called **Links**) are connected to each other through **Joints.**
- Exactly like there are many ways to connect two rigid bodies in the real world, there are many types of **Joints** available with varying degrees of freedom for use in URDF.
- By defining **inertial, visual** and **collision** properties of individual links and the way the links are connected to each other, a robot model is made.
- Once we have a model description, we can parse it to a data structure and use it later for **simulation**, **visualization** and **frame transformation** between different parts of a robot



Universal Robot Description Format cont'd

- Main tags in a URDF file are:
  - <robot>
    - <link>: http://wiki.ros.org/urdf/XML/link
      - <visual>
        - <origin>
        - <geometry>
        - <mesh>
      - <collision>
        - <origin>
        - <geometry>
        - <mesh>
      - <inertial>
        - <mass>
        - <inertia>
    - <joint>: http://wiki.ros.org/urdf/XML/joint
      - <parent>
      - <child>
      - <origin>
      - <dynamics>

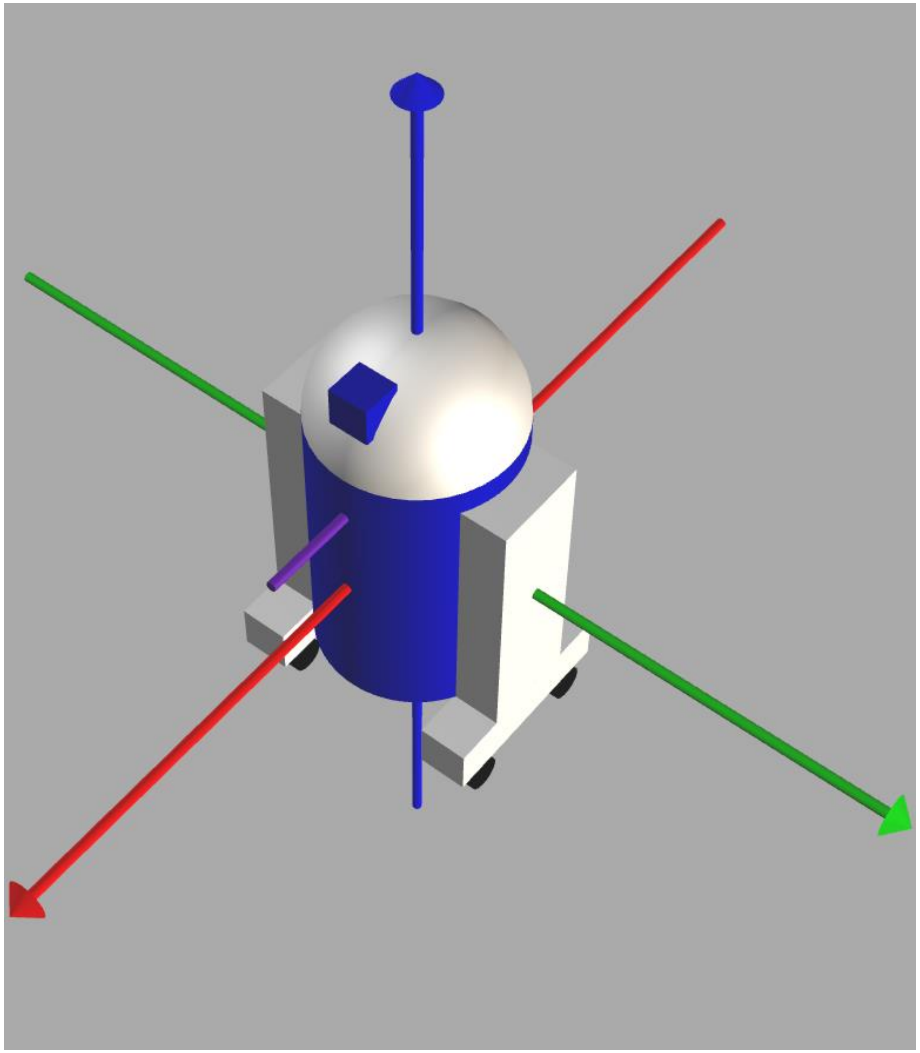


Universal Robot Description Format cont'd

- A useful tool for building and visualizing URDF models can be found here: http://mymodelrobot.appspot.com/
- There's also a tool for exporting Solidworks Assemblies as description packages containing URDF, meshes and launch files for uses with ROS, it can be found here: http://wiki.ros.org/sw_urdf_exporter
    - However, this tool is a bit buggy and needs attention while designing and exporting models so as not to face problems with the generated URDF models later on
- Important Notes:
    - Try to make collision geometries as simple as possible
    - Pay attention to physical properties (such as mass and inertia) because they can cause problems if they are ill-defined.

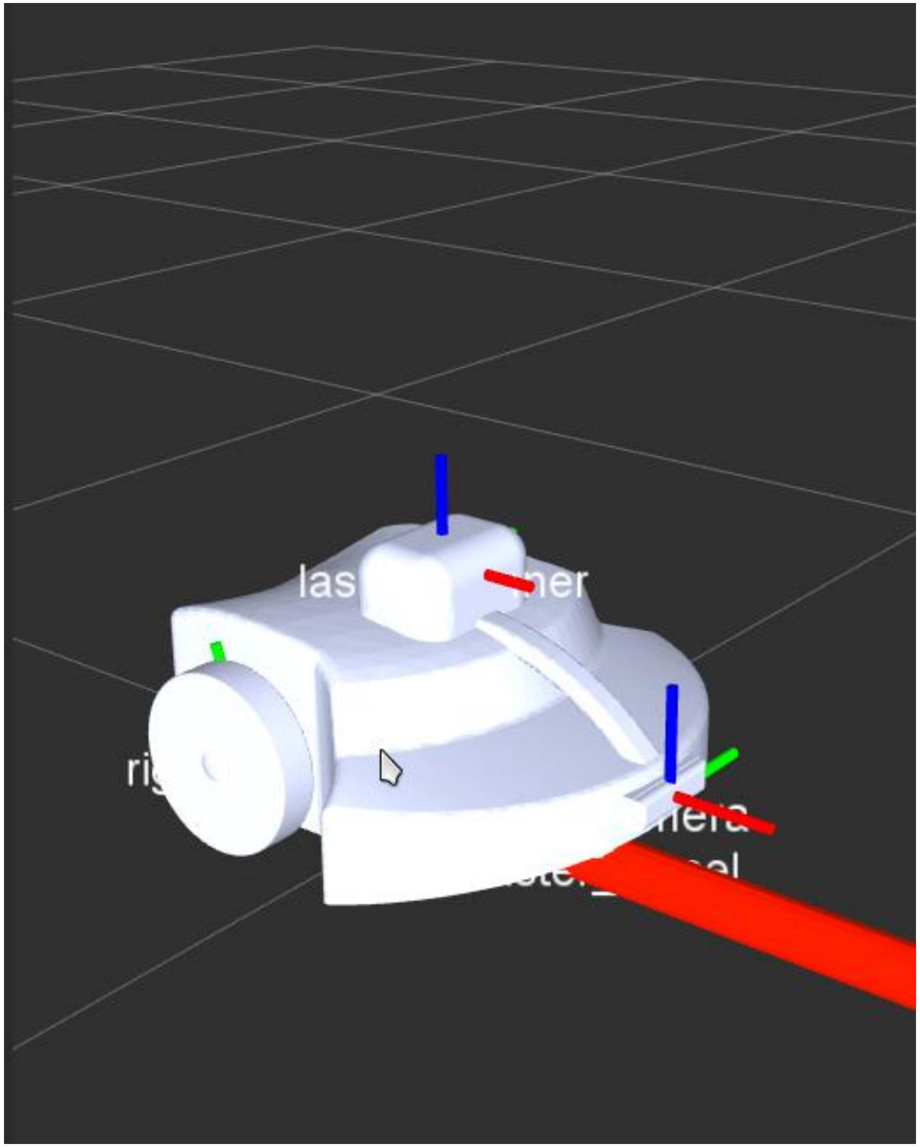
Example URDF

- Here is a series of links to introductory URDF tutorials on the R2D2 robot model
  - Building the robot from scratch
  - Adding Movable Joints
  - Adding collision and physical properties
- Using xacro, modeling complex robot models can be made easier
  - Using xacro to clean up a URDF
- A complex urdf.xacro robot model is provided on the following links:
  - Understanding PR2 model
- Some handly command-line tools:
  - urdf_to_graphiz
  - check_urdf



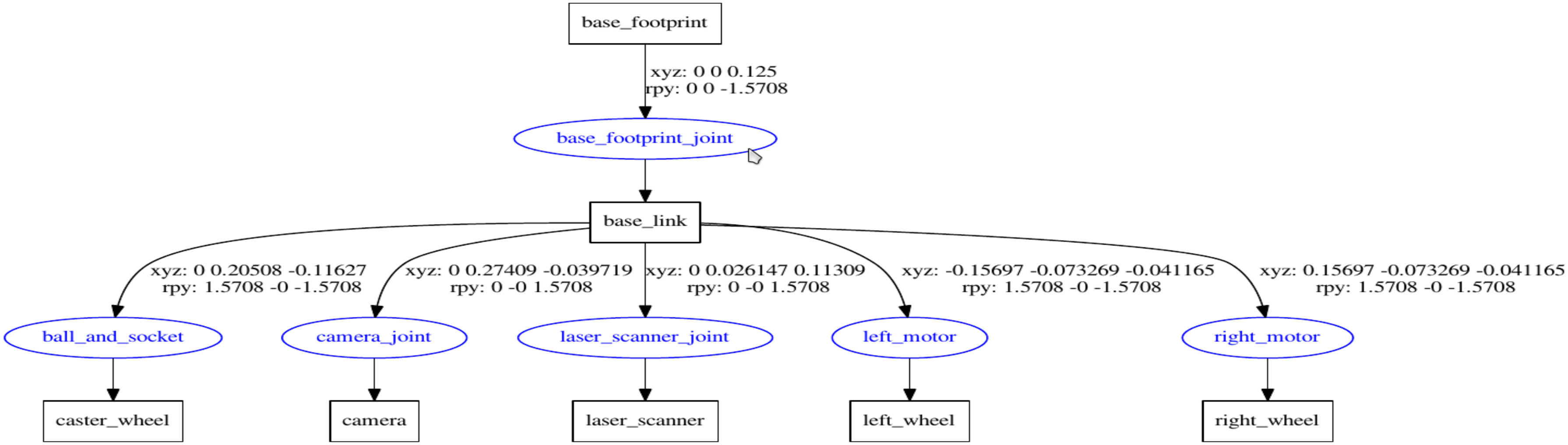
Introducing CATBot

- CATBot is a differential drive robot
- We will use CATBot as a mobile robot platform to apply the algorithms we will study in the course
- CATBot model is exported from a SolidWorks assembly model
- As we go on in the course, we will add functionalities to CATBot.



Introducing CATBot cont'd: catbot_description package

- Download the package containing urdf model of catbot inside your local catkin workspace using the command:
    - $ svn checkout https://github.com/mahmoudabdulazim/src/trunk/catbot_description/
- Let's see the hierarchy of the robot
    - Get into the urdf directory inside the package
    - Use the command:
        - $ urdf_to_graphiz diff_catbot.urdf
    - You'll see two generated files: diff_catbot.pdf and diff_catbot.gv
    - Open the diff_catbot.pdf file, you ought to see something similar to this:

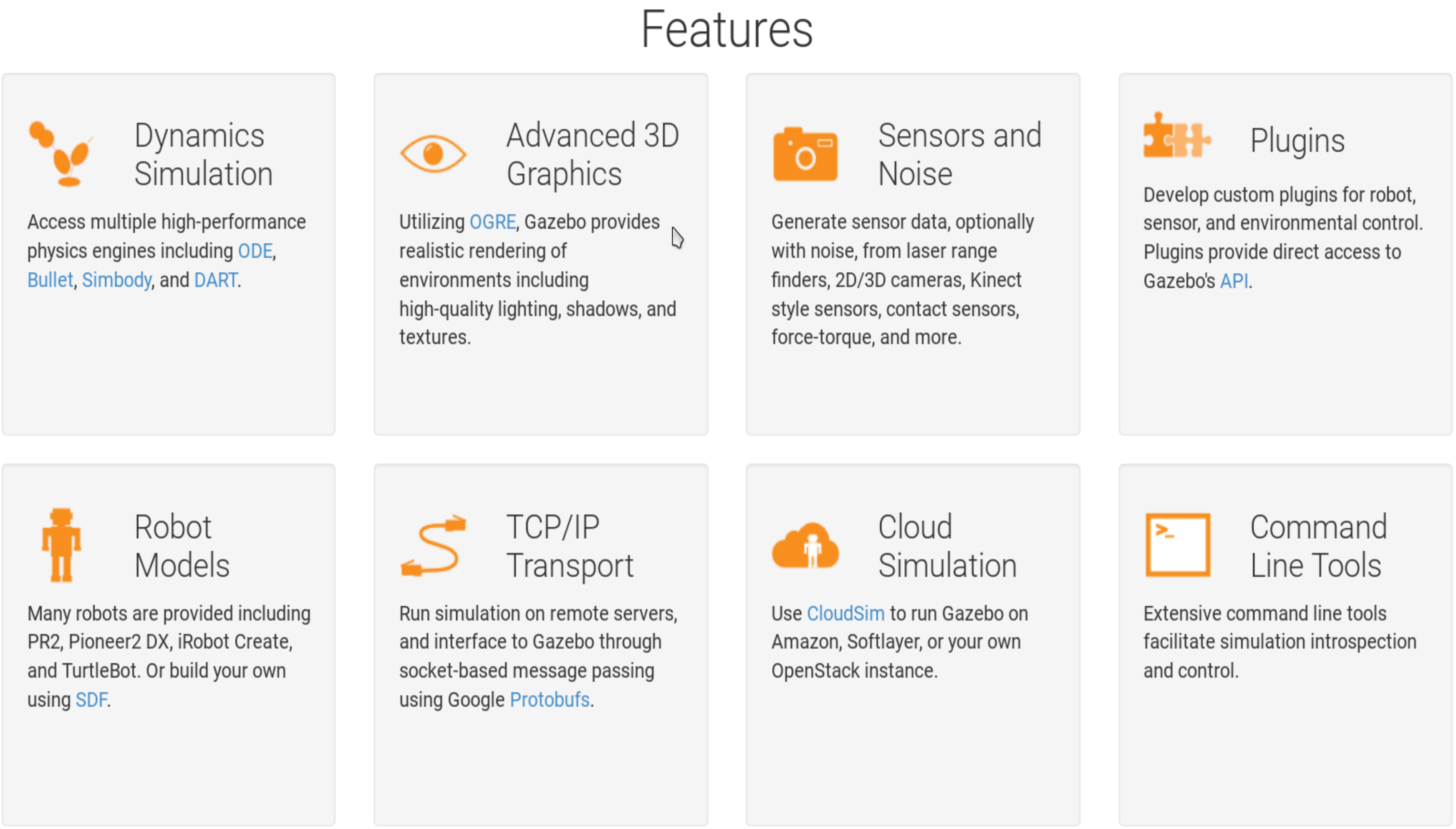


Introducing RViz

- Rviz, is a tool for visualizing most data types in ROS such as:
    - Images
    - Laser Scans
    - Odometry
    - Trajectories
    - Maps
    - Point Clouds
- In Addition, Rviz has plugins that help to integrate it with important packages such as navigation package and moveit package, making the task of goal setting easier and more convenient.
- Note: Rviz is a “**VISUALIZATION”** tool, not a simulation tool.

Introducing Gazebo

- “Gazebo is a well-designed simulator that makes it possible to rapidly test algorithms, design robots, and perform regression testing using realistic scenarios.” - gazebosim.org



Features

Dynamics Simulation

Access multiple high-performance physics engines including ODE, Bullet, Simbody, and DART.

Advanced 3D Graphics

Utilizing OGRE, Gazebo provides realistic rendering of environments including high-quality lighting, shadows, and textures.

Sensors and Noise

Generate sensor data, optionally with noise, from laser range finders, 2D/3D cameras, Kinect style sensors, contact sensors, force-torque, and more.

Plugins

Develop custom plugins for robot, sensor, and environmental control. Plugins provide direct access to Gazebo's API.

Robot Models

Many robots are provided including PR2, Pioneer2 DX, iRobot Create, and TurtleBot. Or build your own using SDF.

TCP/IP Transport

Run simulation on remote servers, and interface to Gazebo through socket-based message passing using Google Protobufs.

Cloud Simulation

Use CloudSim to run Gazebo on Amazon, Softlayer, or your own OpenStack instance.

Command Line Tools

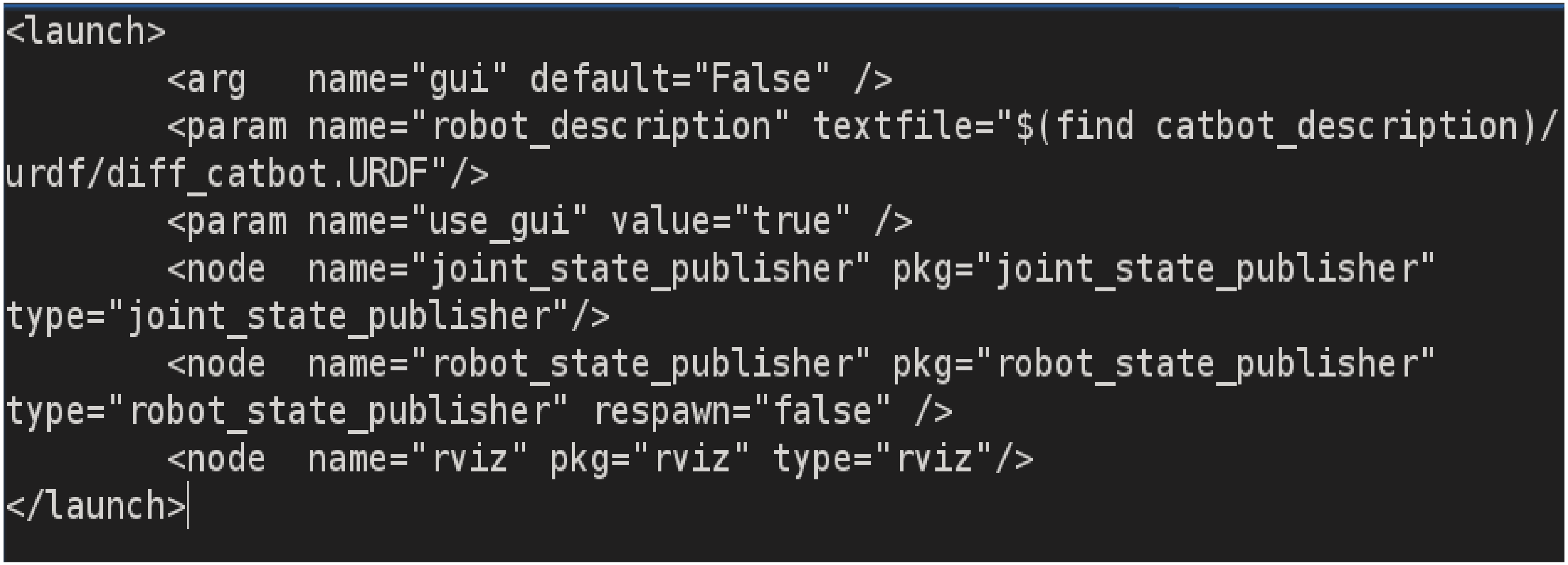
Extensive command line tools facilitate simulation introspection and control.

Introducing Gazebo cont'd

- ROS and Gazebo combined form a very strong tool for doing realistic simulations
- If configured properly, once the algorithms work on the simulation, the task of migration to actual hardware will be a piece of cake thanks to the abstract nature of the interface between ROS and Gazebo
- Gazebo is buggy, it's an open-source project and is still under-development
- Gazebo's official description language is SDF (Standard Description Format), but it also supports URDF
- Additional physical properties used with simulation in Gazebo can be added to a URDF model using the <gazebo> tag
- Let's have a look on CATBot URDF:
  - Go go the directory of catbot_description package
  - Open the file diff_catbot.URDF (you'll find it inside the folder urdf)

Launch Files

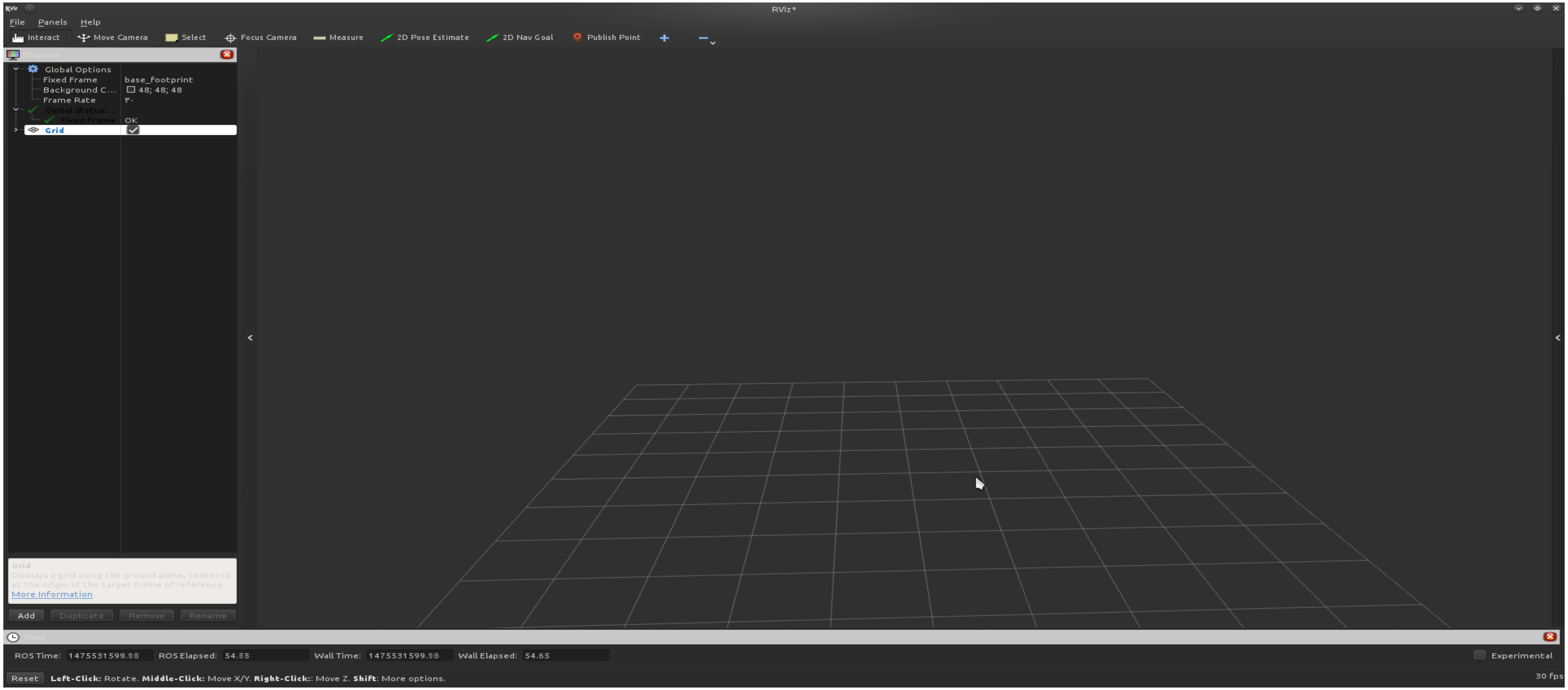
- Launch files are a convenient way to run big projects using a single command
- The idea is to write the commands in a file and use the roslaunch tool to run the nodes
- Launch files have other functionalities such as topic remapping (Advanced)
- Launch files are usually kept inside a directory called launch inside a package (by convention)
- Launch files have a separate syntax that is based on XML, you can find the XML-description of launch files here on this link: http://wiki.ros.org/roslaunch/XML



```
<launch>
        <arg   name="gui" default="False" />
        <param name="robot_description" textfile="$(find catbot_description)/
urdf/diff_catbot.URDF"/>
        <param name="use_gui" value="true" />
        <node  name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher"/>
        <node  name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher" respawn="false" />
        <node  name="rviz" pkg="rviz" type="rviz"/>
</launch>
```

Visualizing CATBot using RViz

- We'll start by visualizing CATBot first using Rviz tool.
- A launch file already exists that will take care of launching necessary nodes
- Let's run the file using the command:
  - roslaunch catbot_description diff_catbot_display.launch



Global Options
Fixed Frame base_footprint
Background C... □ 48; 48; 48
Frame Rate ٣٠
Global Status:...
Fixed Frame OK
Grid ☑

Grid
Displays a grid along the ground plane, centered at the origin of the target frame of reference.
More Information

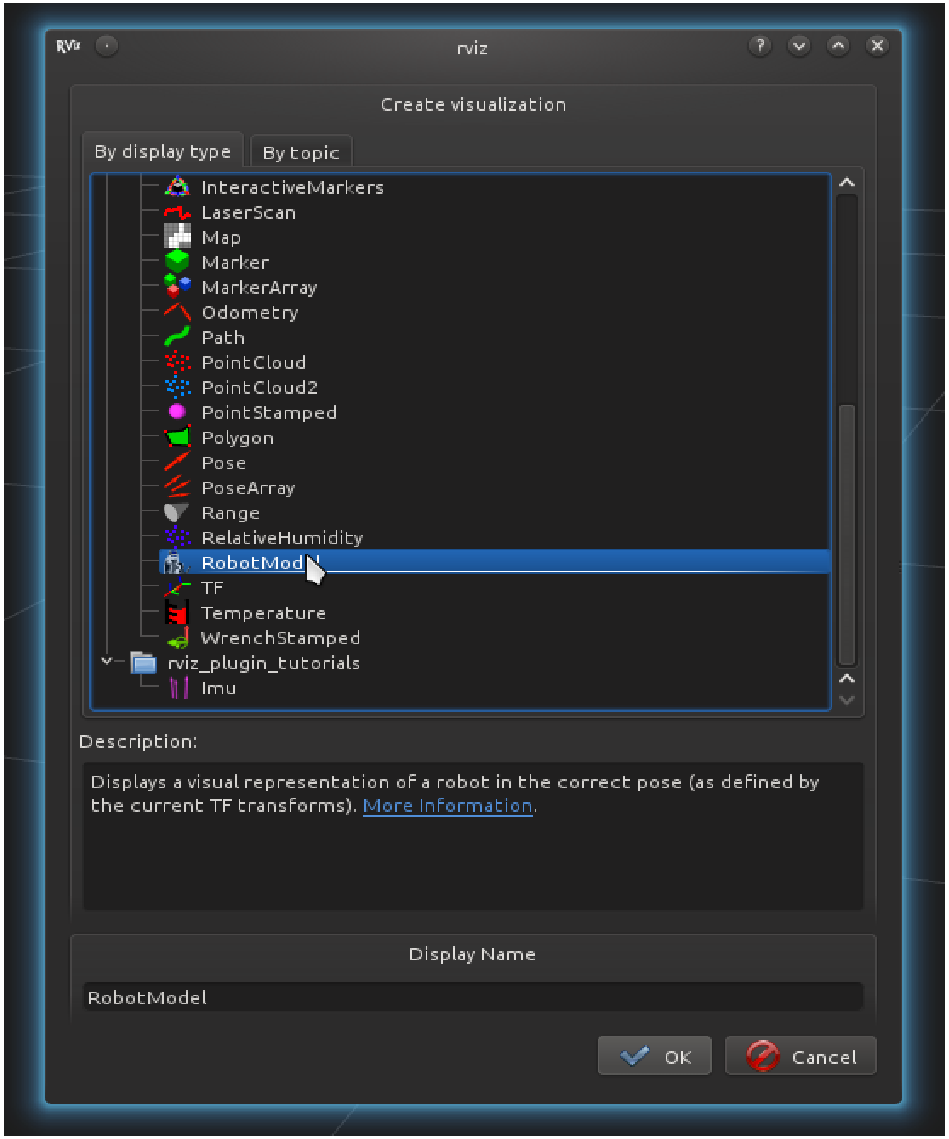
ROS Time: 1475531599.08 ROS Elapsed: 54.65 Wall Time: 1475531599.08 Wall Elapsed: 54.65

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Move Z. **Shift**: More options.

30 fps

Adding A RobotModel for visualization

- On the left bottom side, there's a button labeled "Add", click on it
- Pick RobotModel



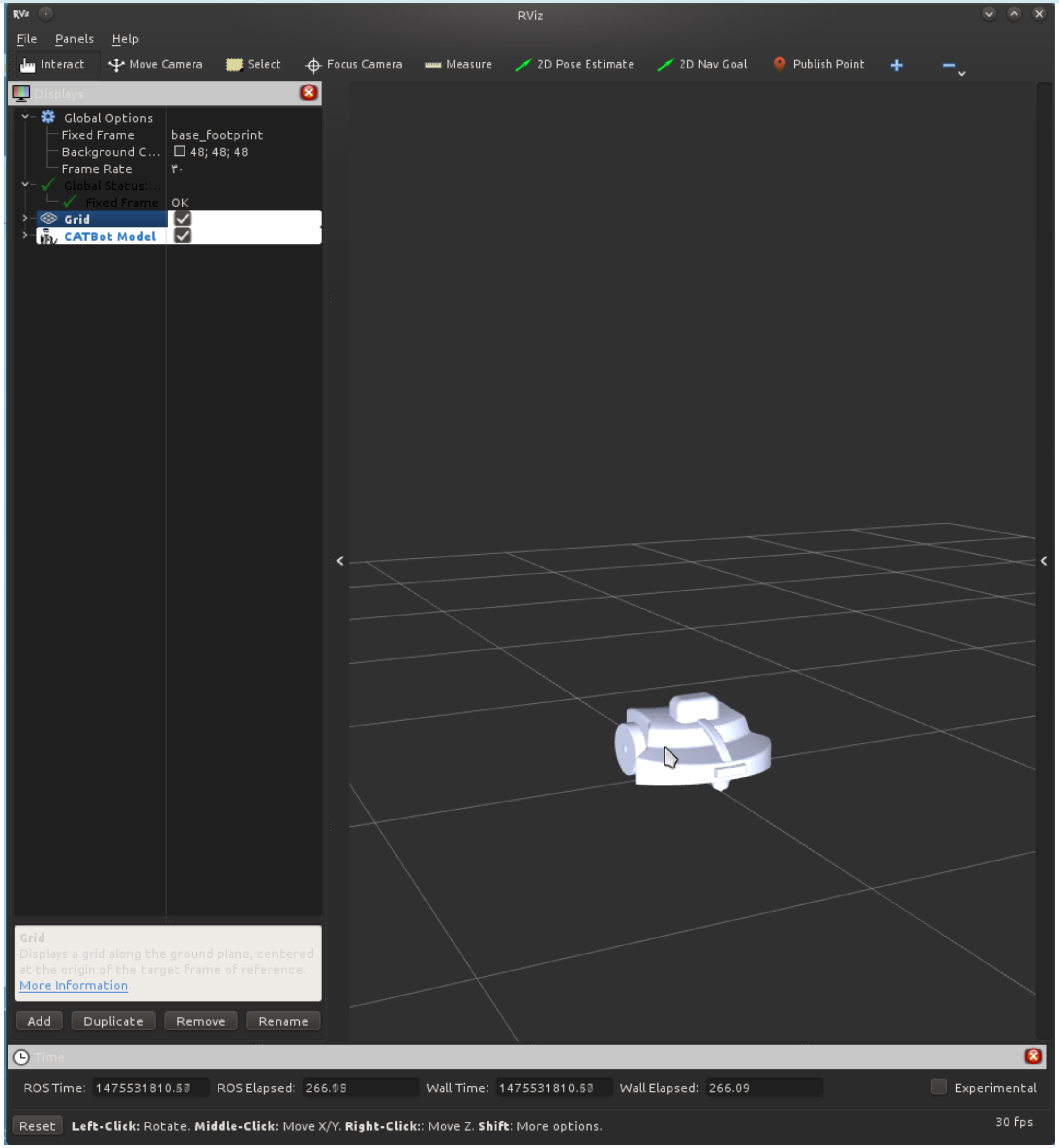
Description:

Displays a visual representation of a robot in the correct pose (as defined by the current TF transforms). More Information.

Display Name: RobotModel

Adding A RobotModel for visualization

- If you have any problems, check that the fixed frame field under the Global Options tab is set to base_footprint
- This is a visualization of the current configuration of the robot.
- Since there is no simulation here, this is just for visualization purposes
- RobotModel option uses the joint states published from joint_state_publisher and robot_state_publisher to visualize the 6DoF state of all links in a URDF model



Global Options
Fixed Frame base_footprint
Background C... □ 48; 48; 48
Frame Rate ٣٠
Global Status:...
✓ Fixed Frame OK
Grid ☑
CATBot Model ☑

Grid
Displays a grid along the ground plane, centered at the origin of the target frame of reference.
More Information

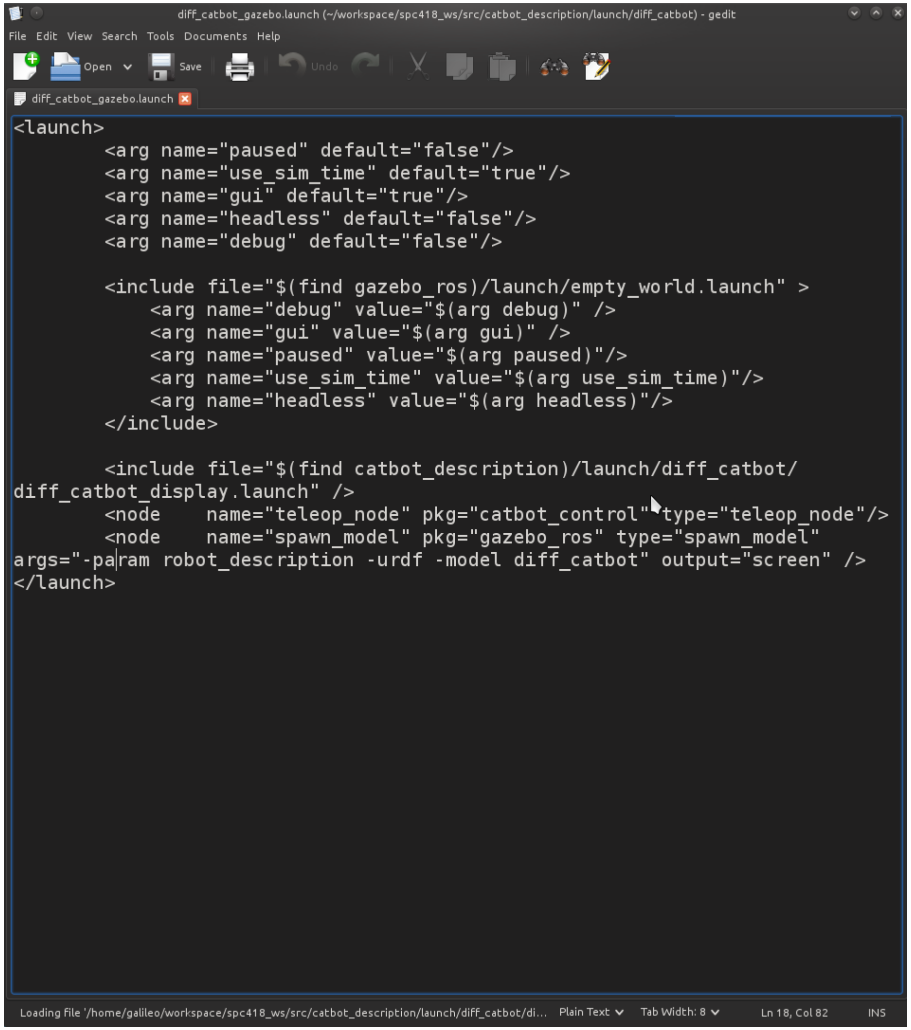
ROS Time: 1475531810.58 ROS Elapsed: 266.08 Wall Time: 1475531810.60 Wall Elapsed: 266.09 □ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Move Z. **Shift**: More options.

30 fps

Simulation

- So far, we've only done visualization using Rviz, let's use Gazebo Simulator
- Inside the catbot_description package directory, a launch file exists that will take care of launching gazebo and spawning a model of our robot
- Use the command:
  - roslaunch catbot_description diff_catbot_gazebo.launch


diff_catbot_gazebo.launch (~/workspace/spc418_ws/src/catbot_description/launch/diff_catbot) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
diff_catbot_gazebo.launch

```
<launch>
        <arg name="paused" default="false"/>
        <arg name="use_sim_time" default="true"/>
        <arg name="gui" default="true"/>
        <arg name="headless" default="false"/>
        <arg name="debug" default="false"/>

        <include file="$(find gazebo_ros)/launch/empty_world.launch" >
            <arg name="debug" value="$(arg debug)" />
            <arg name="gui" value="$(arg gui)" />
            <arg name="paused" value="$(arg paused)"/>
            <arg name="use_sim_time" value="$(arg use_sim_time)"/>
            <arg name="headless" value="$(arg headless)"/>
        </include>

        <include file="$(find catbot_description)/launch/diff_catbot/
diff_catbot_display.launch" />
        <node    name="teleop_node" pkg="catbot_control" type="teleop_node"/>
        <node    name="spawn_model" pkg="gazebo_ros" type="spawn_model"
args="-param robot_description -urdf -model diff_catbot" output="screen" />
</launch>
```

Loading file '/home/galileo/workspace/spc418_ws/src/catbot_description/launch/diff_catbot/di...
Plain Text
Tab Width: 8
Ln 18, Col 82
INS

Exercises

- Go through the R2D2 tutorials on ROS Wiki
- Build a four wheeled rover URDF model using any method of the following
    - Xacro (highly recommended)
    - Normal URDF
    - Solidworks and export it as URDF
- Write a launch file from scratch to visualize your robot, and to spawn it in Gazebo
    - Hint: have a look on the diff_catbot_gazevo.launch file and the diff_catbot_display.launch file
- There's a deliberate mistake in the file diff_catbot_gazebo_willow_garage_world.launch file, find and fix the issue

References

- https://wiki.ros.org/urdf/Tutorials/
- http://gazebosim.org/tutorials/?tut=ros_urdf
- https://github.com/qboticslabs/mastering_ros