

IRP LAB6 Deliverables

Jason Zhai

It contains everything for lab6

The deliverables are highlighted by the big red squares

Task 1 - Create a Wheeled Base URDF

Create Workspace and Initial Link

1. **Action:** Create a new workspace called 'roscourse_sim_ws'. Within this workspace, create the src folder.
2. **Action:** Create a package (Python type) called 'my_robot'. Within this package create a folder called 'urdf'.

```
-----  
ROS VERSION: ros-foxy | ROS_DOMAIN_ID: 66  
-----  
yahboom@VM:~$ mkdir -p roscourse_sim_ws/src  
yahboom@VM:~$ cd roscourse_sim_ws  
yahboom@VM:~/roscourse_sim_ws$ cd src  
yahboom@VM:~/roscourse_sim_ws/src$ ros2 pkg create --build-type ament_python my_robot  
going to create a new package  
package name: my_robot  
destination directory: /home/yahboom/roscourse_sim_ws/src  
package format: 3  
version: 0.0.0  
description: TODO: Package description  
maintainer: ['yahboom <"jzhai87@gmail.com">']  
licenses: ['TODO: License declaration']  
build type: ament_python  
dependencies: []  
creating folder ./my_robot  
creating ./my_robot/package.xml  
creating source folder  
creating folder ./my_robot/my_robot  
creating ./my_robot/setup.py  
creating ./my_robot/setup.cfg  
creating folder ./my_robot/resource  
creating ./my_robot/resource/my_robot  
creating ./my_robot/my_robot/__init__.py  
creating folder ./my_robot/test  
creating ./my_robot/test/test_copyright.py  
creating ./my_robot/test/test_flake8.py  
creating ./my_robot/test/test_pep257.py  
yahboom@VM:~/roscourse_sim_ws/src$ cd my_robot  
yahboom@VM:~/roscourse_sim_ws/src/my_robot$ mkdir urdf
```

3. **Action:** Create a file called 'my_robot.urdf' within the urdf folder. Add to it:

```
1  <?xml version="1.0"?>
2  <robot name="my_robot">
3    <link name="base_link">
4      <visual>
5        <geometry>
6          <box size="0.6 0.4 0.2" />
7        </geometry>
8        <origin xyz="0 0 0" rpy="0 0 0" />
9      </visual>
10   </link>
11 </robot>
12
```

4. **Question:** What are the units for the box size? What does each number represent?

```
yahboom@VM:~/roscourse_sim_ws/src/my_robot$ cd urdf
yahboom@VM:~/roscourse_sim_ws/src/my_robot/urdf$ nano my_robot.urdf
```

```
GNU nano 4.8
<?xml version="1.0"?>
<robot name="my_robot">
  <link name="base_link">
    <visual>
      <geometry>
        <box size="0.6 0.4 0.2" />
      </geometry>
      <origin xyz="0 0 0" rpy="0 0 0" />
    </visual>
  </link>
</robot>
```

4. Answer the Question

- **Units:** The box size values are in **meters**.
- **Values represent:**
 - 0.6 : length (x-axis)
 - 0.4 : width (y-axis)
 - 0.2 : height (z-axis)

5. **Note:** To visualize our URDF, we would need to write a launch file to provide the necessary publishers for Rviz. We will look at this later. For now, we can use a demo version launch file.

6. **Action:** Open a terminal and execute (all one command)

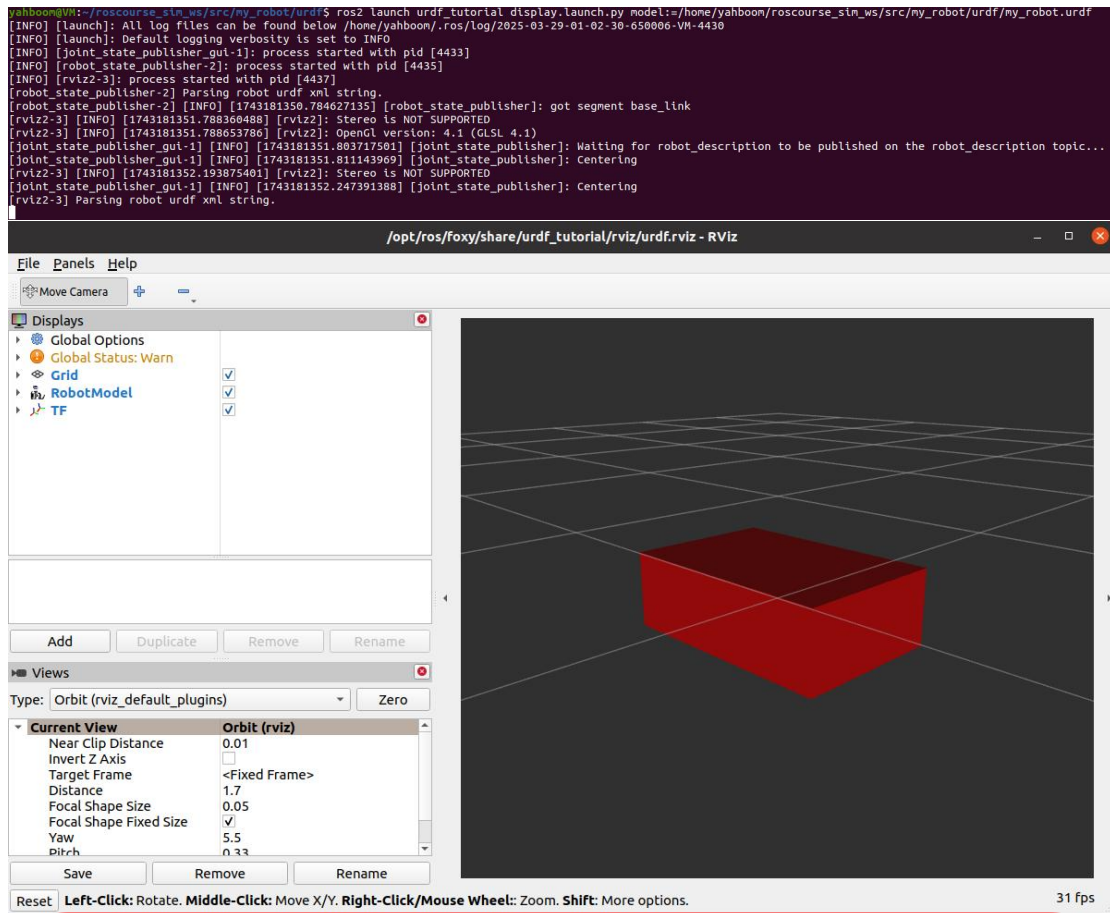
```
ros2 launch urdf_tutorial display.launch.py
```

```
model:=/home/yahboom/roscourse_sim_ws/src/my_robot/urdf/my_robot.urdf
```

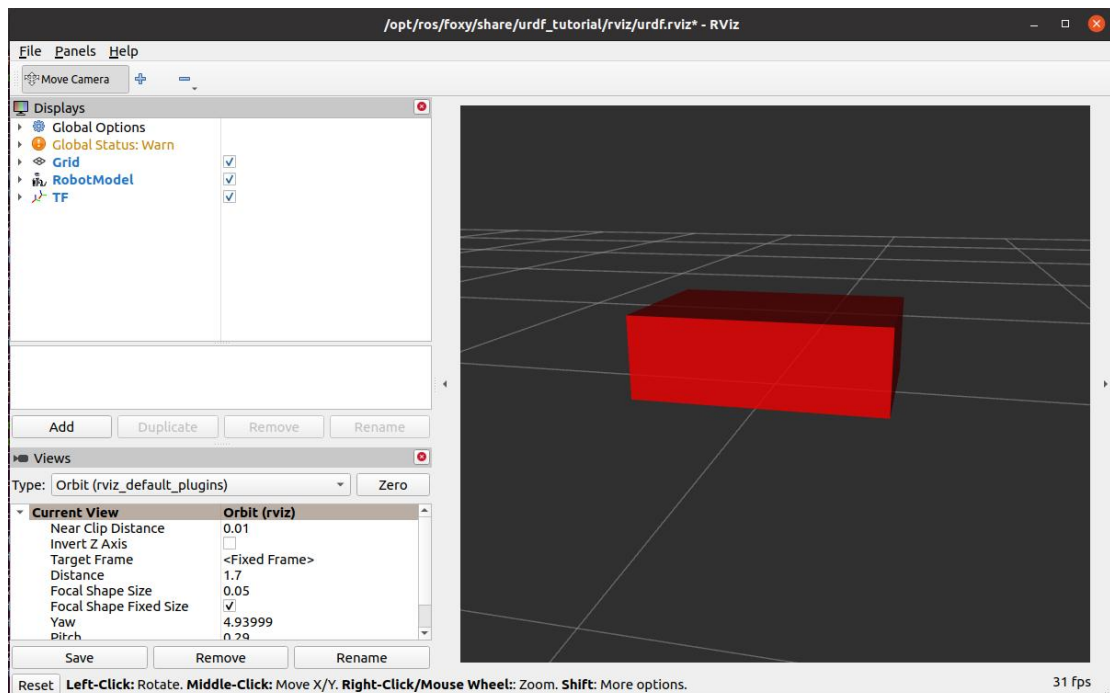
Note that your path may be different.

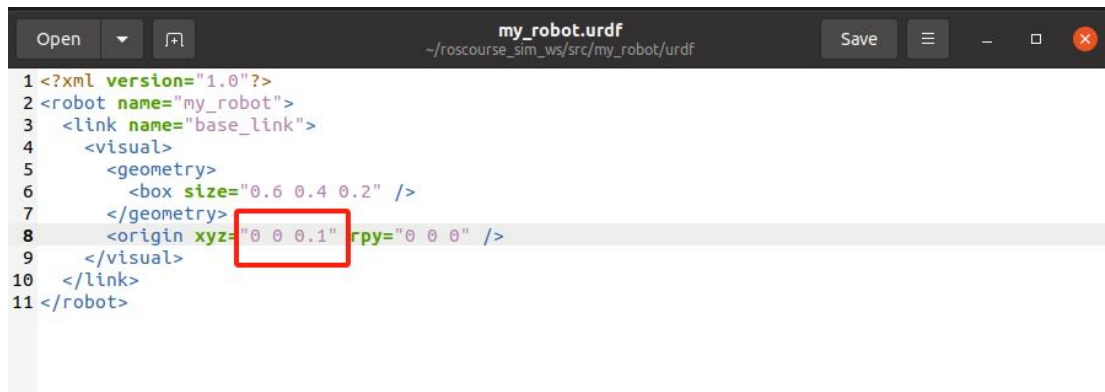
Bigger note: the path must be absolute (you must use /home/<etc>, not the tilde shortcut)

7. **Note:** You should have seen Rviz open with a single box visible. If you do not have urdf_tutorial installed, use the command `sudo apt install ros-foxy-urdf-tutorial`. Be sure to re-source the terminal after: `source /opt/ros/foxy/setup.bash` When prompted to provide a password: yahboom



8. **Action:** Modify the origin tag such that the bottom of the box is on the ground plane. (The box is on top of the grid rather than the grid passing through the box.) You should only need to change one value.





```
1 <?xml version="1.0"?>
2 <robot name="my_robot">
3   <link name="base_link">
4     <visual>
5       <geometry>
6         <box size="0.6 0.4 0.2" />
7       </geometry>
8       <origin xyz="0 0 0.1" rpy="0 0 0" />
9     </visual>
10  </link>
11 </robot>
```

9. **Note:** By default, everything is red. This will be hard for visualization so let's add colors.

10. **Action:** In the URDF file, before the link tag and after the robot start tag, let's add a material that is green color and set this first link to be that color.

```
1   <?xml version="1.0"?>
2   <robot name="my_robot">
3     <material name="green">
4       <color rgba="0 0.6 0 1"/>
5     </material>
6
7     <link name="base_link">
8       <visual>
9         <geometry>
10          <box size="0.6 0.4 0.2" />
11        </geometry>
12        <origin xyz="0 0 0" rpy="0 0 0" />
13        <material name="green"/>
14      </visual>
15    </link>
16  </robot>
17
```

11. **Action:** Re-run the demo file with this new version. You should now see a green box.

12. **Note:** The default alpha for Rviz is 0.8. You can change this transparency by opening 'RobotModel' in the 'Displays' section (upper left) and changing the 'Alpha' value.

/opt/ros/foxy/share/urdf_tutorial/rviz/urdf.rviz* - RViz

File Panels Help

Move Camera

Displays

- Global Options
- Global Status: Warn
- Grid ☒
- RobotModel ☒
- TF ☒

Add Duplicate Remove Rename

Views
Type: Orbit (rviz_default_plugins) Zero

Current View	Orbit (rviz)
Near Clip Distance	0.01
Invert Z Axis	<input type="checkbox"/>
Target Frame	<Fixed Frame>
Distance	1.7
Focal Shape Size	0.05
Focal Shape Fixed Size	<input checked="" type="checkbox"/>
Yaw	5.525
Pitch	0.17

Save Remove Rename

Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options. 31 fps

Open

my_robot.urdf
~/roscourse_sim_ws/src/my_robot/urdf

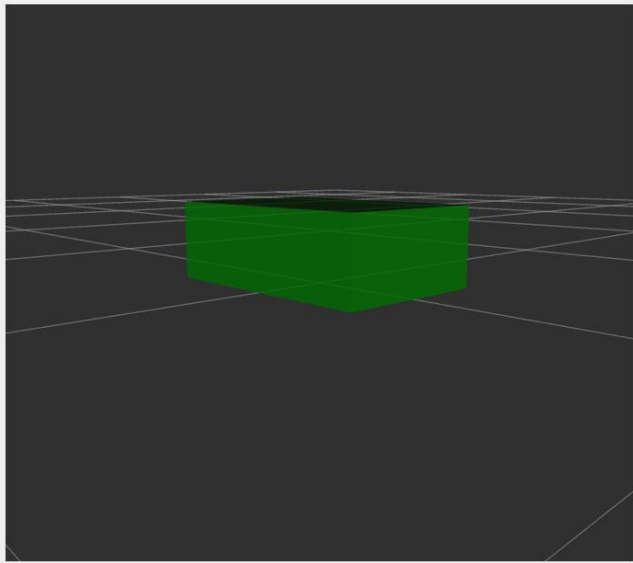
Save

```

1 <?xml version="1.0"?>
2 <robot name="my_robot">
3   <material name="green">
4     <color rgba="0 0.6 0 1" />
5   </material>
6
7   <link name="base_link">
8     <visual>
9       <geometry>
10        <box size="0.6 0.4 0.2" />
11      </geometry>
12      <origin xyz="0 0 0.1" rpy="0 0 0" />
13      <material name="green" />
14    </visual>
15  </link>
16 </robot>

```

Loading file "/home/yahboom/roscourse_sim_ws/src/my_robot/ur... XML Tab Width: 8 Ln 3, Col 24 INS



Add a Second Link

13. **Action:** Add a new material with name 'white'. Note that the RGB values should be in the range [0 1]. To be on the gray spectrum, the three values should be equal.

14. **Action:** Add a new link (after 'base_link'). Call this link 'lidar'

```
1 <link name="lidar">
2   <visual>
3     <geometry>
4       <cylinder radius="0.1" length="0.05" />
5     </geometry>
6     <origin xyz="0 0 0" rpy="0 0 0" />
7     <material name="white"/>
8   </visual>
9 </link>
```

3

15. **REQUIRED Question:** What happens if you try to run the demo now? Why?



```
Open my_robot.urdf Save
~/roscourse_sim_ws/src/my_robot/urdf

1 <?xml version="1.0"?>
2 <robot name="my_robot">
3   <material name="green">
4     <color rgba="0 0.6 0 1"/>
5   </material>
6
7   <material name="white">
8     <color rgba="0.8 0.8 0.8 1.0"/>
9   </material>
10
11   <link name="base_link">
12     <visual>
13       <geometry>
14         <box size="0.6 0.4 0.2" />
15       </geometry>
16       <origin xyz="0 0 0.1" rpy="0 0 0" />
17       <material name="green"/>
18     </visual>
19   </link>
20
21   <link name="lidar">
22     <visual>
23       <geometry>
24         <cylinder radius="0.1" length="0.05" />
25       </geometry>
26       <origin xyz="0 0 0" rpy="0 0 0" />
27       <material name="white"/>
28     </visual>
29   </link>
30
31 </robot>
```

Required question: Everything just disappeared

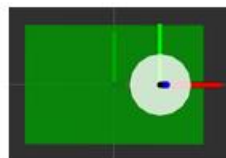
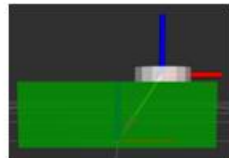
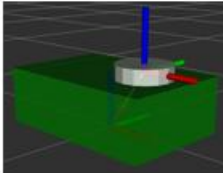
Because we added a second link (lidar) but didn't connect it to the base. In URDF, all links must be connected via joints to form a single tree. An unconnected link is considered "floating", which is not allowed.

16. **Action:** We need to add a joint to describe the relationship between these links.

```
1 <joint name="base_lidar_joint" type="fixed">
2   <parent link="base_link"/>
3   <child link="lidar"/>
4   <origin xyz="0 0 0" rpy="0 0 0"/>
5 </joint>
6
```

17. **Note:** As written, the lidar link will be centered at the base_link frame origin. Let's say we want the lidar to be on top of the base and halfway between the middle and front of the box as seen in the figures (start of next page).

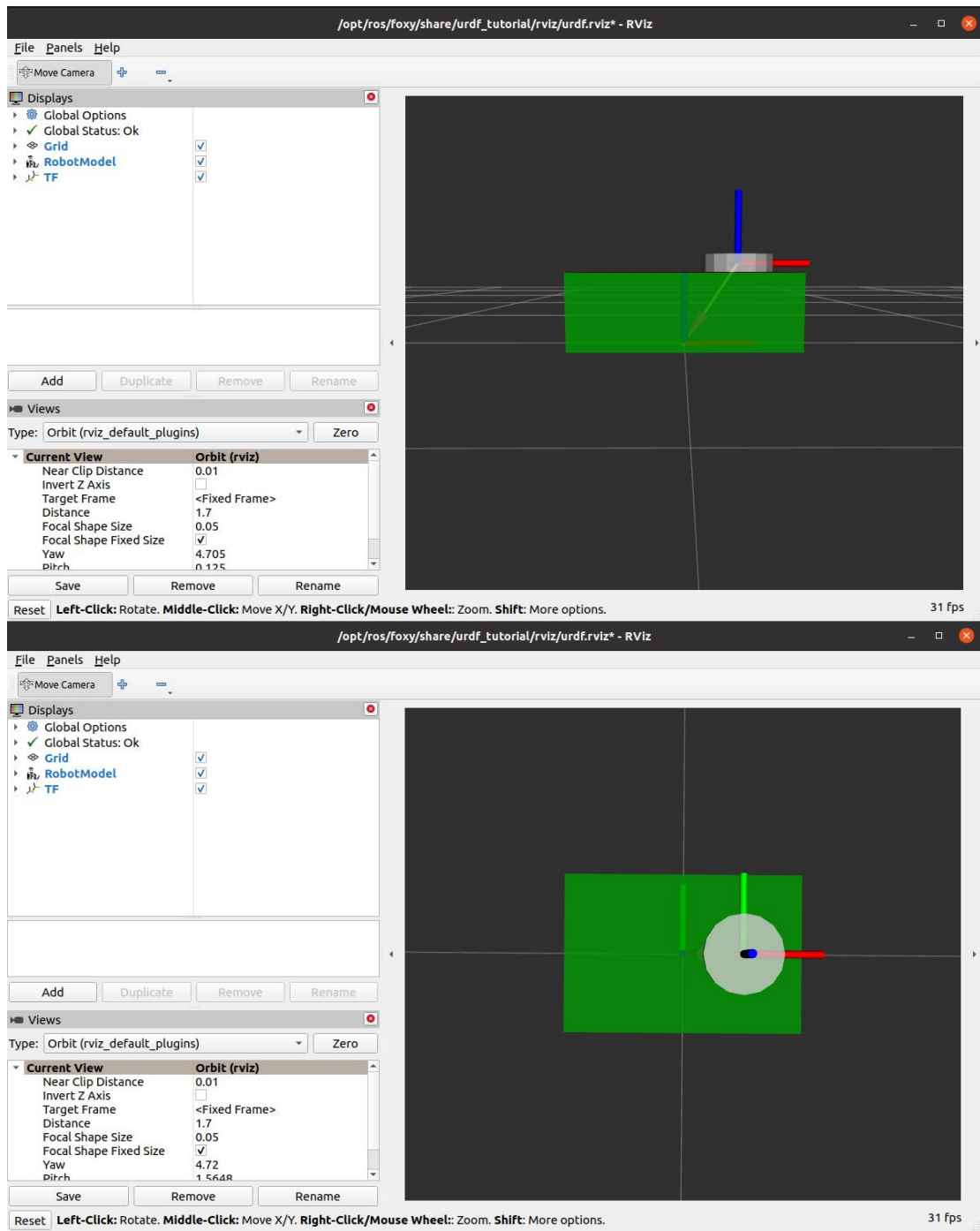
18. **Action:** Modify the origins so that the lidar is positioned as shown. Hint: Modify x,z for the joint.

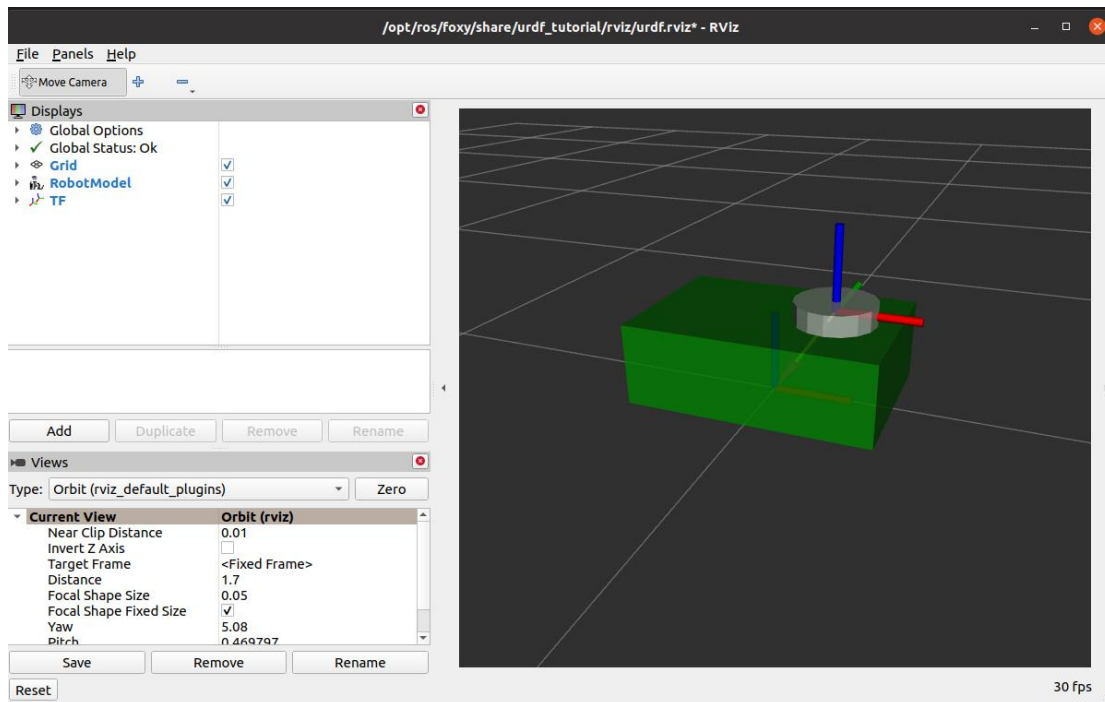


```
Open  my_robot.urdf  Save  -  x
~/roscourse_sim_ws/src/my_robot/urdf

1 <?xml version="1.0"?>
2 <robot name="my_robot">
3   <material name="green">
4     <color rgba="0 0.6 0 1"/>
5   </material>
6
7   <material name="white">
8     <color rgba="0.8 0.8 0.8 1.0"/>
9   </material>
10
11 <link name="base_link">
12   <visual>
13     <geometry>
14       <box size="0.6 0.4 0.2" />
15     </geometry>
16     <origin xyz="0 0 0.1" rpy="0 0 0" />
17     <material name="green"/>
18   </visual>
19 </link>
20
21 <link name="lidar">
22   <visual>
23     <geometry>
24       <cylinder radius="0.1" length="0.05" />
25     </geometry>
26     <origin xyz="0 0 0" rpy="0 0 0" />
27     <material name="white"/>
28   </visual>
29 </link>
30
31 <joint name="base_lidar_joint" type="fixed">
32   <parent link="base_link"/>
33   <child link="lidar"/>
34   <origin xyz="0.15 0 0.225" rpy="0 0 0"/>
35 </joint>
36
37 </robot>

XML  Tab Width: 8  Ln 34, Col 30  INS
```





Add Wheels

19. **Note:** Next we need to add some wheels. We will have two new links, and therefore by necessity two new joints.
A good procedure is to 1) create the new link with visual origin values all zero 2) create the joint 3) modify the joint origin 4) modify the visual origin if needed. **Why?** (Hint: How are frames inherited?)
20. **Action:** Create a new material called 'gray' with RGB values of 0.7.
21. **Action:** Create a new link named 'left_wheel'. Base this off of the 'lidar' link. Use the same shape and dimensions, but use the new gray material.

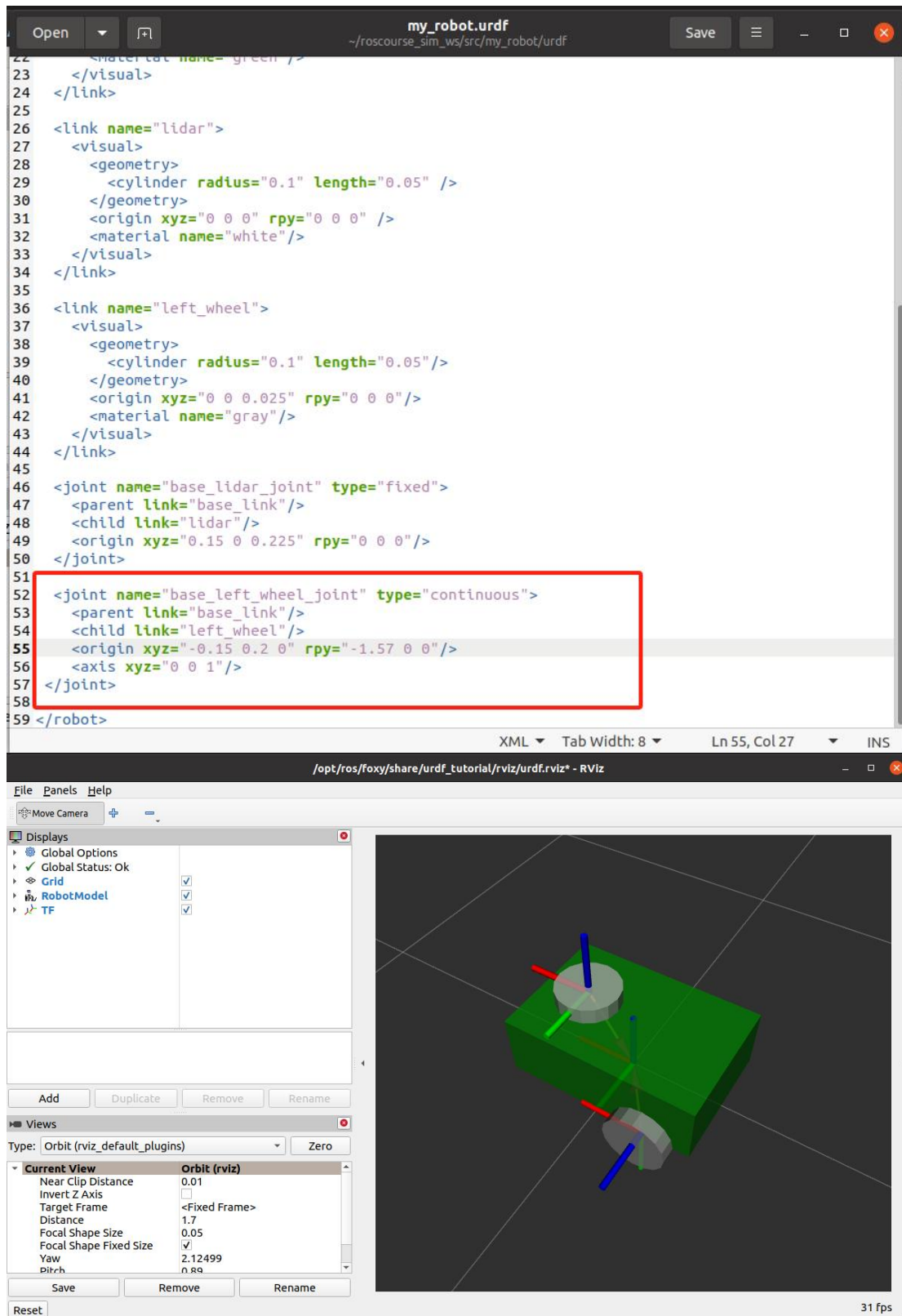
```
Open  [icon] *my_robot.urdf ~/roscourse_sim_ws/src/my_robot/urdf Save [icon] [icon] [icon] [icon] [icon]
1 <?xml version="1.0"?>
2 <robot name="my_robot">
3   <material name="green">
4     <color rgba="0 0.6 0 1"/>
5   </material>
6
7   <material name="white">
8     <color rgba="0.8 0.8 0.8 1.0"/>
9   </material>
10
11   <material name="gray">
12     <color rgba="0.7 0.7 0.7 1.0"/>
13   </material>
14
15
16   <link name="base_link">
17     <visual>
18       <geometry>
19         <box size="0.6 0.4 0.2" />
20       </geometry>
21       <origin xyz="0 0 0.1" rpy="0 0 0" />
22       <material name="green"/>
23     </visual>
24   </link>
25
26   <link name="lidar">
27     <visual>
28       <geometry>
29         <cylinder radius="0.1" length="0.05" />
30       </geometry>
31       <origin xyz="0 0 0" rpy="0 0 0" />
32       <material name="white"/>
33     </visual>
34   </link>
35
36   <link name="left_wheel">
37     <visual>
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52 </robot>
```

```
Open  [icon] *my_robot.urdf ~/roscourse_sim_ws/src/my_robot/urdf Save [icon] [icon] [icon] [icon] [icon]
15
16 <link name="base_link">
17   <visual>
18     <geometry>
19       <box size="0.6 0.4 0.2" />
20     </geometry>
21     <origin xyz="0 0 0.1" rpy="0 0 0" />
22     <material name="green"/>
23   </visual>
24 </link>
25
26 <link name="lidar">
27   <visual>
28     <geometry>
29       <cylinder radius="0.1" length="0.05" />
30     </geometry>
31     <origin xyz="0 0 0" rpy="0 0 0" />
32     <material name="white"/>
33   </visual>
34 </link>
35
36 <link name="left_wheel">
37   <visual>
38     <geometry>
39       <cylinder radius="0.1" length="0.05"/>
40     </geometry>
41     <origin xyz="0 0 0.025" rpy="0 0 0"/>
42     <material name="gray"/>
43   </visual>
44 </link>
45
46 <joint name="base_lidar_joint" type="fixed">
47   <parent link="base_link"/>
48   <child link="lidar"/>
49   <origin xyz="0.15 0 0.225" rpy="0 0 0"/>
50 </joint>
51
52 </robot>
```

22. **Action:** Add a new joint for this wheel.

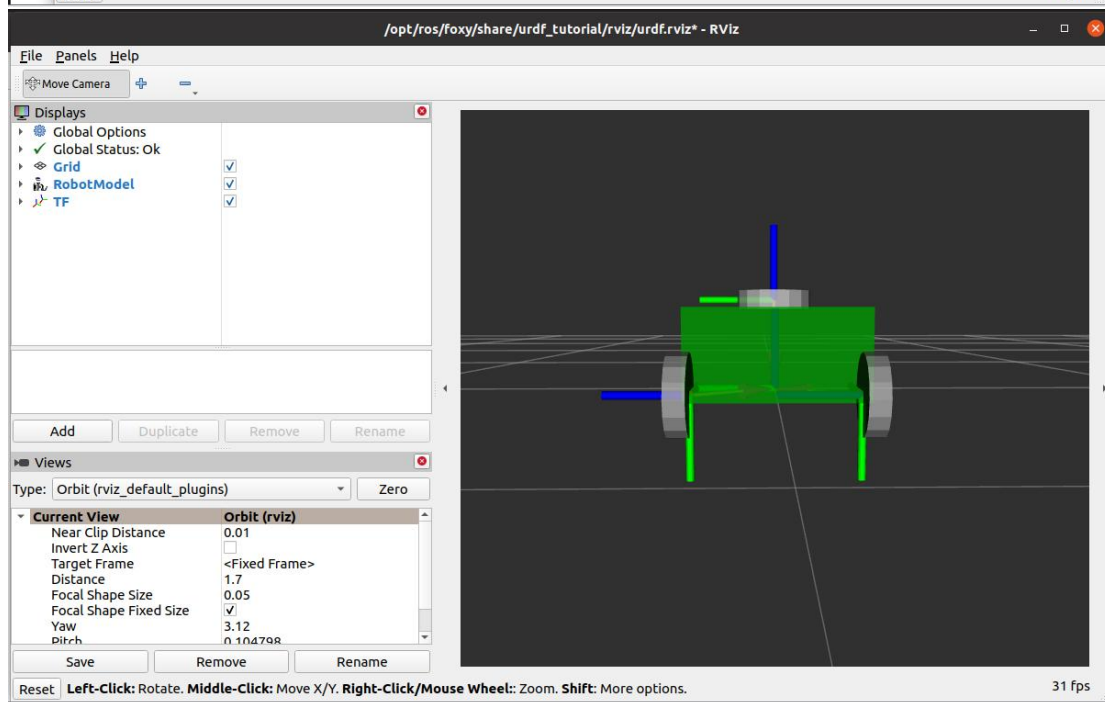
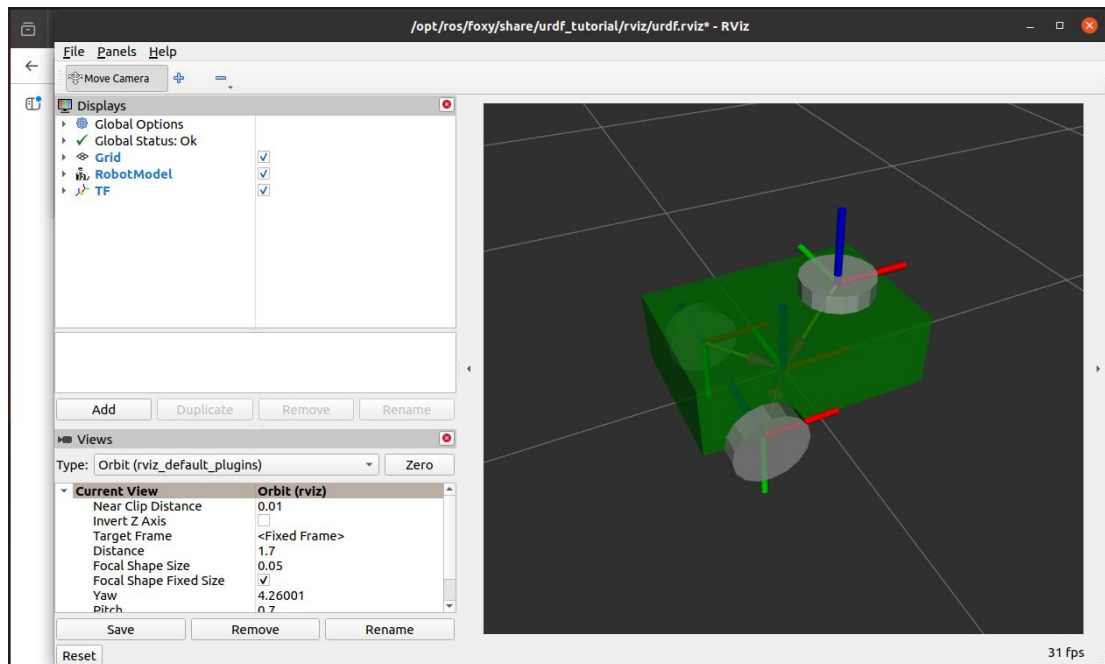
```
1 <joint name="base_left_wheel_joint" type="continuous">
2   <parent link="base_link"/>
3   <child link="left_wheel"/>
4   <origin xyz="0 0 0" rpy="0 0 0"/>
5   <axis xyz="1 0 0"/>
6 </joint>
7
```

23. **Note:** If we use the demo launch file to visualize this, we can see that the wheel is not where we want it to be (obviously since the new origins are all zero). We also note that the axis of rotation is incorrect. How do we know that? Use the GUI sliders to rotate the new joint.
24. **Note:** To put the wheel where we want, let's first slide the joint origin, then rotate the joint frame, then adjust the link visual frame if needed.
25. **Action:** Change the joint origin to have `xyz="-0.15 0.2 0"`
26. **Question:** Where did these numbers come from?
27. **Note:** Now the position is correct, but the wheel is not at the correct orientation. I want to set up the wheel so that a positive rotation about its z-axis results in forward motion. If the base x-axis represents my forward motion, that means that the z-axis of the wheel should be aligned with the base y-axis. To get this orientation, we need to rotate this frame about the x-axis in the negative direction by 90 degrees.
28. **Action:** Change the joint origin to have `rpy="-1.57 0 0"` and the joint axis to `xyz="0 0 1"`
29. **Question:** Why 1.57 and not 90?
30. **Note:** Now the wheel axis is at the right location and we can see that increasing the joint angle results in wheel motion that would correspond to forward motion. The only problem is we can see the wheel visual intersects the box. Let's adjust that.
31. **Action:** Change the 'left_wheel' visual origin to `xyz="0 0 0.025"`
32. **Question:** Why did I change the z value?
33. **Note:** We could have modified the joint origin so that the joint rotates about the center of the wheel rather than at the side of the box. The choice depends on how you want to represent your frames, but there may be preference to have as many zeros as possible.



34. **Challenge:** Add the right wheel using the same method that was used for the left wheel. The size and material should be the same as the left and a positive rotation about the wheel's z-axis should result in forward motion. The link should be named 'right_wheel' and the joint 'base_right_wheel_joint'.
35. **Note:** Copy/paste in vi or vim: Move to the first or last line you want to copy. Press v to enter visual mode. Use the arrow keys to select the lines that you want to copy. Press y (stands for 'yank') to copy. Move to the line where you want to insert and press p.

```
35
36 <link name="left_wheel">
37   <visual>
38     <geometry>
39       <cylinder radius="0.1" length="0.05"/>
40     </geometry>
41     <origin xyz="0 0 0.025" rpy="0 0 0"/>
42     <material name="gray"/>
43   </visual>
44 </link>
45
46 <link name="right_wheel">
47   <visual>
48     <geometry>
49       <cylinder radius="0.1" length="0.05"/>
50     </geometry>
51     <origin xyz="0 0 -0.025" rpy="0 0 0"/>
52     <material name="gray"/>
53   </visual>
54 </link>
55
56
57 <joint name="base_lidar_joint" type="fixed">
58   <parent link="base_link"/>
59   <child link="lidar"/>
60   <origin xyz="0.15 0 0.225" rpy="0 0 0"/>
61 </joint>
62
63 <joint name="base_left_wheel_joint" type="continuous">
64   <parent link="base_link"/>
65   <child link="left_wheel"/>
66   <origin xyz="-0.15 0.2 0" rpy="-1.57 0 0"/>
67   <axis xyz="0 0 1"/>
68 </joint>
69
70 <joint name="base_right_wheel_joint" type="continuous">
71   <parent link="base_link"/>
72   <child link="right_wheel"/>
73   <origin xyz="-0.15 -0.2 0" rpy="-1.57 0 0"/>
74   <axis xyz="0 0 1"/>
75 </joint>
76
77
78 </robot>
```

36. **Action:** Add a caster wheel on the back for balance.

```
1 <link name="caster_wheel">
2   <visual>
3     <geometry>
4       <sphere radius="0.05" />
5     </geometry>
6     <origin xyz="0 0 0" rpy="0 0 0" />
7     <material name="gray" />
8   </visual>
9 </link>
```

```
1 <joint name="base_caster_wheel_joint" type="fixed">
2   <parent link="base_link"/>
3   <child link="caster_wheel"/>
4   <origin xyz="0.2 0 -0.05" rpy="0 0 0" />
5 </joint>
```

37. **Note:** Our robot model is complete! Since we have a mobile robot, we will want to move it around. For this purpose, it is useful to have a fixed reference frame (a 'world' frame). We can incorporate this by adding a virtual link to define the world which we can link to our robot base. This is also useful for simulation in an application like Gazebo.

```
<geometry>
  <cylinder radius="0.1" length="0.05"/>
</geometry>
<origin xyz="0 0 0.025" rpy="0 0 0"/>
<material name="gray"/>
</visual>
</link>
```

```
<link name="right_wheel">
  <visual>
    <geometry>
      <cylinder radius="0.1" length="0.05"/>
    </geometry>
    <origin xyz="0 0 -0.025" rpy="0 0 0"/>
    <material name="gray"/>
  </visual>
</link>
```

```
<link name="caster_wheel">
  <visual>
    <geometry>
      <sphere radius="0.05"/>
    </geometry>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <material name="gray"/>
  </visual>
</link>
```

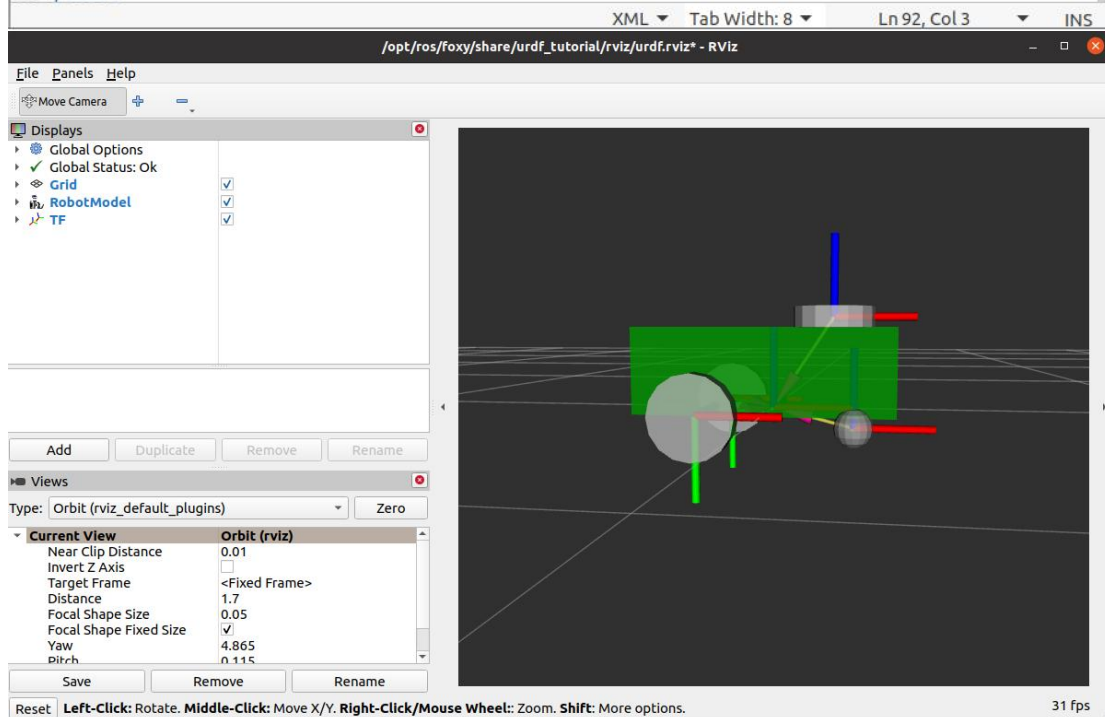
```
<joint name="base_lidar_joint" type="fixed">
  <parent link="base_link"/>
  <child link="lidar"/>
  <origin xyz="0.15 0 0.225" rpy="0 0 0"/>
</joint>
```

```
<joint name="base_left_wheel_joint" type="continuous">
```

```

66
67
68 <joint name="base_lidar_joint" type="fixed">
69   <parent link="base_link"/>
70   <child link="lidar"/>
71   <origin xyz="0.15 0 0.225" rpy="0 0 0"/>
72 </joint>
73
74 <joint name="base_left_wheel_joint" type="continuous">
75   <parent link="base_link"/>
76   <child link="left_wheel"/>
77   <origin xyz="-0.15 0.2 0" rpy="-1.57 0 0"/>
78   <axis xyz="0 0 1"/>
79 </joint>
80
81 <joint name="base_right_wheel_joint" type="continuous">
82   <parent link="base_link"/>
83   <child link="right_wheel"/>
84   <origin xyz="-0.15 -0.2 0" rpy="-1.57 0 0"/>
85   <axis xyz="0 0 1"/>
86 </joint>
87
88 <joint name="base_caster_wheel_joint" type="fixed">
89   <parent link="base_link"/>
90   <child link="caster_wheel"/>
91   <origin xyz="0.2 0 -0.05" rpy="0 0 0"/>
92 </joint>
93
94
95
96 </robot>

```



Add a Virtual Reference Link

38. **Action:** Add a new link: `<link name="base_footprint" />`
This link is virtual so it has no visual information.
39. **Action:** Add the joint such that the world reference frame (base_footprint) is at the same x,y position and orientation as the mobile base link but the origin is at the bottom of the wheels.

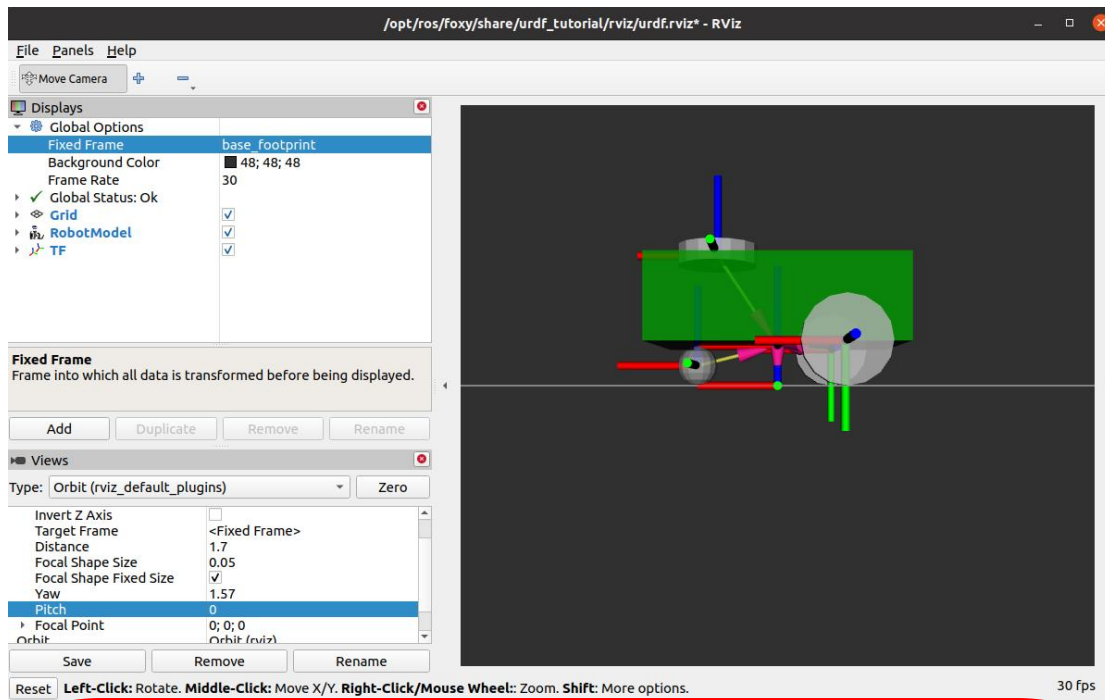
```
1 <joint name="base_joint" type="fixed">
2   <parent link="base_footprint"/>
3   <child link="base_link"/>
```

6

```
4   <origin xyz="0 0 0.1" rpy="0 0 0" />
5   </joint>
6
```

40. **Action:** Re-run the demo file with our completed URDF. In the 'Displays' panel, under 'Global Options', set 'Fixed Frame' to 'base_footprint'. Now in the 'Views' panel, set the 'Yaw' to 1.57 and the 'Pitch' to 0. You should now see that your robot is "on the ground".
41. **Note:** Keep in mind this is a visualizer, not a simulator. So while it looks like it is "on the ground", that is simply because that is the view we chose. Try changing the 'Fixed Frame' and moving the joint sliders to see the effect on the view.

```
73
74 <joint name="base_left_wheel_joint" type="continuous">
75   <parent link="base_link"/>
76   <child link="left_wheel"/>
77   <origin xyz="-0.15 0.2 0" rpy="-1.57 0 0"/>
78   <axis xyz="0 0 1"/>
79 </joint>
80
81 <joint name="base_right_wheel_joint" type="continuous">
82   <parent link="base_link"/>
83   <child link="right_wheel"/>
84   <origin xyz="-0.15 -0.2 0" rpy="-1.57 0 0"/>
85   <axis xyz="0 0 1"/>
86 </joint>
87
88 <joint name="base_caster_wheel_joint" type="fixed">
89   <parent link="base_link"/>
90   <child link="caster_wheel"/>
91   <origin xyz="0.2 0 -0.05" rpy="0 0 0"/>
92 </joint>
93
94 <link name="base_footprint" />
95
96 <joint name="base_joint" type="fixed">
97   <parent link="base_footprint"/>
98   <child link="base_link"/>
99   <origin xyz="0 0 0.1" rpy="0 0 0"/>
100 </joint>
101
102
103
104 </robot>
```

Task 2 - Make Launch File For RVIZ

1. **Action:** On the same level as the 'urdf' folder, make a new folder called 'launch' and add the file 'my_robot_rviz.launch.py'.
2. **REQUIRED Question:** What are some of the nodes in the launch file and what do they do?
3. **Action:** On the same level as the 'urdf' folder, make a new folder called 'rviz' and add the file 'my_robot.rviz'.
4. **Action:** Modify the setup.py:
 - Add `from glob import glob`
 - Add `import os`
 - Add `from setuptools import find_packages`
 - Add inside `data_files` `(os.path.join('share', package_name), glob('urdf/*'))`
 - Add inside `data_files` `(os.path.join('share', package_name, 'launch'), glob(os.path.join('launch', '*launch.[pxy][yma]*')))`
 - Add inside `data_files` `(os.path.join('share', package_name), glob('rviz/*'))`
5. **Note:** The data files arguments are going to make copies of all files in the urdf and rviz folders so that they are accessible via the install folder after building.
6. **Action:** Build and source the workspace. View our model using the new launch file
`ros2 launch my_robot my_robot_rviz.launch.py`

Required question:

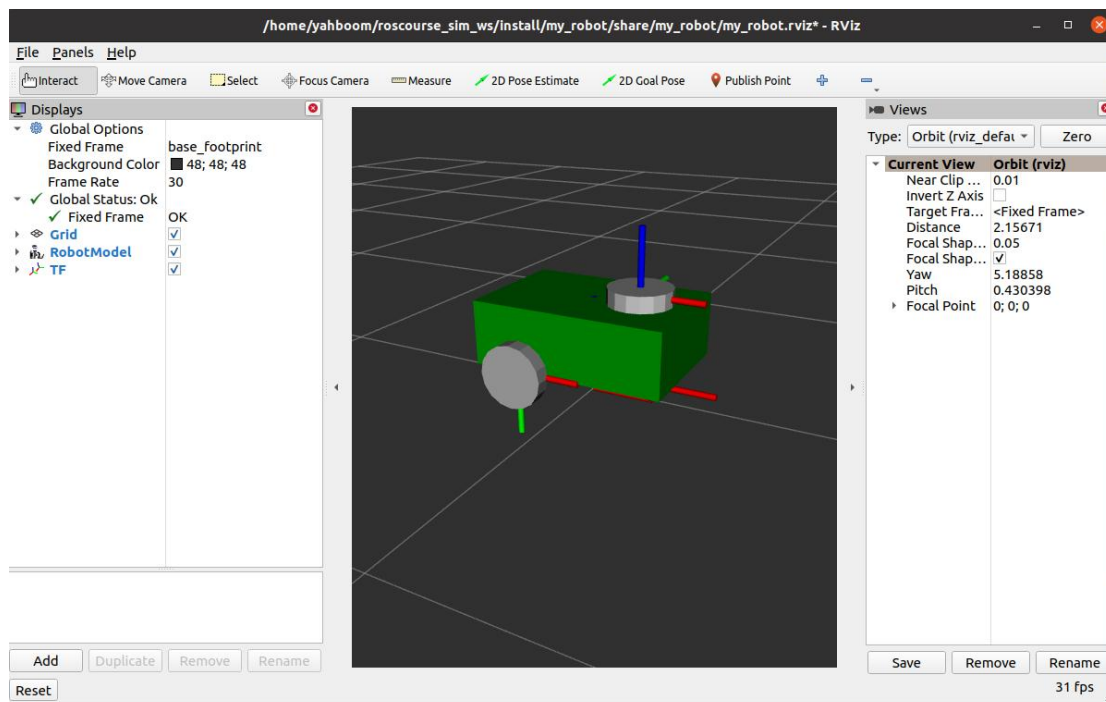
robot_state_publisher: Publishes the transforms (/tf) between all robot links based on the URDF.
rviz2: Opens the RViz GUI with a preset config (my_robot.rviz) to visualize the robot model.


```
Open  setup.py  Save  -  x
~/roscourse_sim_ws/src/my_robot

my_robot.urdf  my_robot_rviz.launch.py  setup.py  my_robot.rviz

1 from setuptools import setup, find_packages
2 from glob import glob
3 import os
4
5 package_name = 'my_robot'
6
7 setup(
8     name=package_name,
9     version='0.0.0',
10    packages=[package_name],
11    data_files=[
12        (os.path.join('share', package_name), glob('urdf/*')),
13        (os.path.join('share', package_name, 'launch'), glob('launch/*.launch.py[y]')),
14        (os.path.join('share', package_name), glob('rviz/*')),
15
16        ('share/ament_index/resource_index/packages',
17         ['resource/' + package_name]),
18        ('share/' + package_name, ['package.xml']),
19    ],
20    install_requires=['setuptools'],
21    zip_safe=True,
22    maintainer='yahboom',
23    maintainer_email='jzhai87@gmail.com',
24    description='TODO: Package description',
25    license='TODO: License declaration',
26    tests_require=['pytest'],
27    entry_points={
28        'console_scripts': [
29        ],
30    },
31 )
```

```
yahboom@VM:~/roscourse_sim_ws$ source install/setup.bash
yahboom@VM:~/roscourse_sim_ws$ ros2 launch my_robot my_robot_rviz.launch.py
[INFO] [launch]: All log files can be found below /home/yahboom/.ros/log/2025-04-02 12-17-45-200477-VM-10639
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [joint_state_publisher_gui-1]: process started with pid [10641]
[INFO] [robot_state_publisher-2]: process started with pid [10643]
[INFO] [rviz2-3]: process started with pid [10645]
[robot_state_publisher-2] Parsing robot urdf xml string.
[robot_state_publisher-2] Link base_link had 4 children
[robot_state_publisher-2] Link caster_wheel had 0 children
[robot_state_publisher-2] Link left_wheel had 0 children
[robot_state_publisher-2] Link lidar had 0 children
[robot_state_publisher-2] Link right_wheel had 0 children
[robot_state_publisher-2] [INFO] [1743567465.303005821] [robot_state_publisher]: got segment base_footprint
[robot_state_publisher-2] [INFO] [1743567465.303100578] [robot_state_publisher]: got segment base_link
[robot_state_publisher-2] [INFO] [1743567465.303107721] [robot_state_publisher]: got segment caster_wheel
[robot_state_publisher-2] [INFO] [1743567465.303112250] [robot_state_publisher]: got segment left_wheel
[robot_state_publisher-2] [INFO] [1743567465.303116580] [robot_state_publisher]: got segment lidar
[robot_state_publisher-2] [INFO] [1743567465.303120575] [robot_state_publisher]: got segment right_wheel
[joint_state_publisher_gui-1] [INFO] [1743567483.191908400] [joint_state_publisher]: Waiting for robot_description to be published on the robot_description topic...
[rviz2-3] [INFO] [1743567483.194696556] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-3] [INFO] [1743567483.194983461] [rviz2]: OpenGL version: 4.1 (GLSL 4.1)
[joint_state_publisher_gui-1] [INFO] [1743567483.199826258] [joint_state_publisher]: Centering
[rviz2-3] [INFO] [1743567490.344469513] [rviz2]: Stereo is NOT SUPPORTED
[joint_state_publisher_gui-1] [INFO] [1743567490.380739223] [joint_state_publisher]: Centering
[rviz2-3] Parsing robot urdf xml string.
```



Task 3 - Visualize Webcam in RVIZ

1. **Action:** Launch the webcam publisher from the previous lab.
2. **Action:** In another terminal, launch Rviz by typing `rviz2`
3. **Action:** Add a new display that shows the camera image. Do this using the 'Add' button and searching by topic.

Alternative if your webcam is not functional:

1. **Action:** In one terminal, run


```
ros2 run robot_state_publisher robot_state_publisher --ros-args -p
robot_description:="$(xacro /opt/ros/foxy/share/urdf_tutorial/urdf/05-visual.
urdf)"
```
2. **Action:** In another terminal, launch Rviz by typing `rviz2`
3. **Action:** Add the robot visual.
 - (a) Select the 'Add' button. Then select 'RobotModel'.
 - (b) Under 'RobotModel' in 'Displays', set the 'Description Topic' value to the correct topic. (It is a drop down menu.)
 - (c) Change 'Fixed Frame' to be 'base_link'.

