Note: This tutorial assumes you know how to write well-formatted XML code.

Please ask about problems and questions regarding this tutorial on answers.ros.org (http://answers.ros.org). Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Building a Visual Robot Model with URDF from Scratch

Description: Learn how to build a visual model of a robot that you can view in Rviz

Keywords: URDF

Tutorial Level: BEGINNER

Next Tutorial: Making the Model Move (/urdf/Tutorials/Building%20a%20Movable%20Robot%20Model%20with%20URDF)

Contents

In this tutorial, we're going to build a visual model of a robot that vaguely looks like R2D2. In later tutorials, you'll learn how to articulate the model (/urdf/Tutorials/Building%20a%20Movable%20Robot%20Model%20with%20URDF), add in some physical properties (/urdf/Tutorials/Adding%20Physical%20and%20Collision%20Properties%20to%20a%20URDF%20Model), generate neater code with xacro (/urdf/Tutorials/Using%20Xacro%20to%20Clean%20Up%20a%20URDF%20File) and make it move in Gazebo (/urdf/Tutorials/Using%20a%20URDF%20in%20Gazebo). But for now, we're going to focus on getting the visual geometry correct.

Before continuing, make sure you have the joint_state_publisher (/joint_state_publisher) package installed. If you installed urdf_tutorial (/urdf_tutorial) using `apt-get`, this should already be the case. If not, please update your installation to include that package (use `rosdep` to check).

All of the robot models mentioned in this tutorial (and the source files) can be found in the urdf_tutorial (/urdf_tutorial) package.

1. One Shape

First, we're just going to explore one simple shape. Here's about as simple as a urdf as you can make. Source (https://github.com/ros/urdf_tutorial/tree/master/urdf/01-myfirst.urdf)

Toggle line numbers

```
 1 <?xml version="1.0"?>
 2 <robot name="myfirst">
 3   <link name="base_link">
 4     <visual>
 5       <geometry>
 6         <cylinder length="0.6" radius="0.2"/>
 7       </geometry>
 8     </visual>
 9   </link>
10 </robot>
```

To translate the XML into English, this is a robot with the name `myfirst`, that contains only one link (a.k.a. part), whose visual component is just a cylinder 0.6 meters long with a 0.2 meter radius. This may seem like a lot of enclosing tags for a simple "hello world" type example, but it will get more complicated, trust me.

To examine the model, launch the `display.launch` file:

```
$ roslaunch urdf_tutorial display.launch model:=urdf/01-myfirst.urdf
```

This does three things. It

- Loads the specified model into the parameter server
- Runs nodes to publish sensor_msgs/JointState (http://docs.ros.org/api/sensor_msgs/html/msg/JointState.html) and transforms (more on these later)
- Starts Rviz with a configuration file

Note that the `roslaunch` line above assumes that you are executing it from the urdf_tutorial (/urdf_tutorial) package directory (ie: the `urdf` directory is a direct child of the current working directory). If that is not the case, the relative path to `01-myfirst.urdf` will not be valid, and you'll receive an error as soon as `roslaunch` tries to load the urdf to the parameter server.

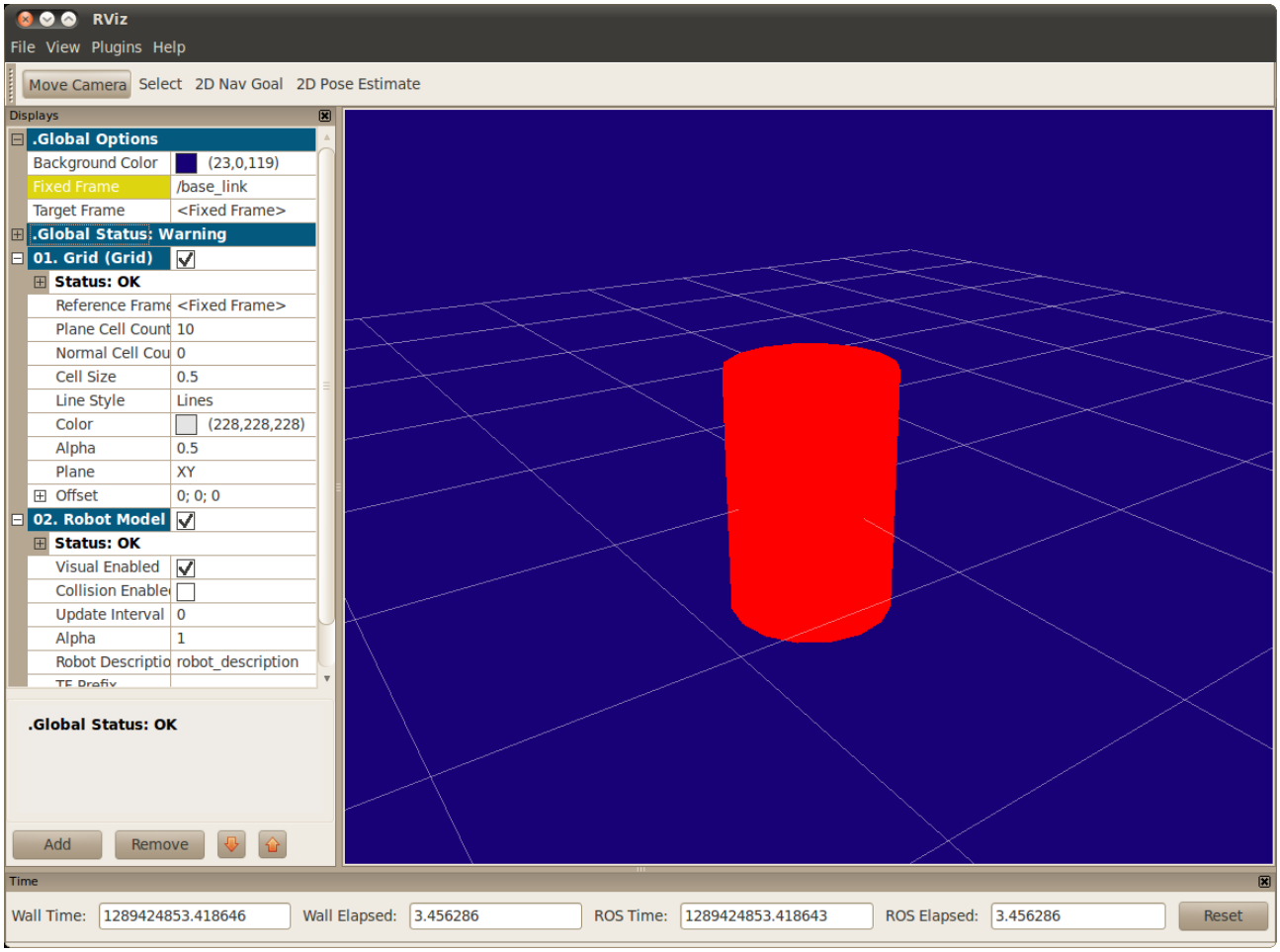
A slightly modified argument allows this to work regardless of the current working directory:

```
$ roslaunch urdf_tutorial display.launch model:='$(find urdf_tutorial)/urdf/01
-myfirst.urdf'
```

note the single quotes around the argument value.

You'll have to change all example `roslaunch` lines given in these tutorials if you are not running them from the `urdf_tutorial` package location.

After launching `display.launch`, you should end up with RViz showing you the following:



Things to note:

- The fixed frame is the transform frame where the center of the grid is located. Here, it's a frame defined by our one link, base_link.
- The visual element (the cylinder) has its origin at the center of its geometry as a default. Hence, half the cylinder is below the grid.

2. Multiple Shapes

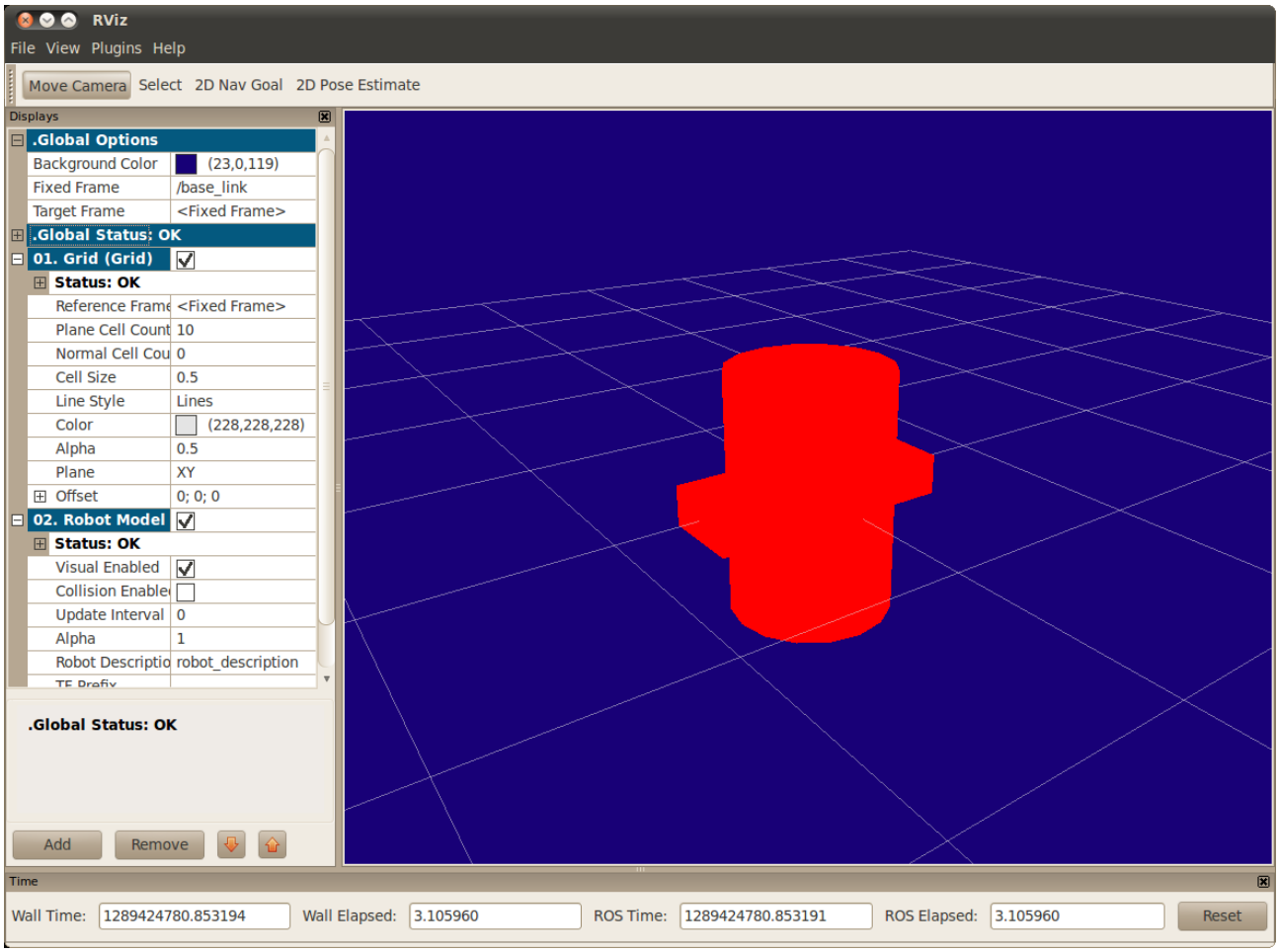
Now let's look at how to add multiple shapes/links. If we just add more link elements to the urdf, the parser won't know where to put them. So, we have to add joints. Joint elements can refer to both flexible and inflexible joints. We'll start with inflexible, or fixed joints. Source (https://github.com/ros/urdf_tutorial/tree/master/urdf/02-multipleshapes.urdf)

Toggle line numbers

```
<?xml version="1.0"?>
<robot name="multipleshapes">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
  </joint>

</robot>
```

- Note how we defined a 0.6m x 0.1m x 0.2m box
- The joint is defined in terms of a parent and a child. URDF is ultimately a tree structure with one root link. This means that the leg's position is dependent on the base_link's position.

```
roslaunch urdf_tutorial display.launch model:=urdf/02-multipleshapes.urdf
```



Both of the shapes overlap with each other, because they share the same origin. If we want them not to overlap we must define more origins.

3. Origins

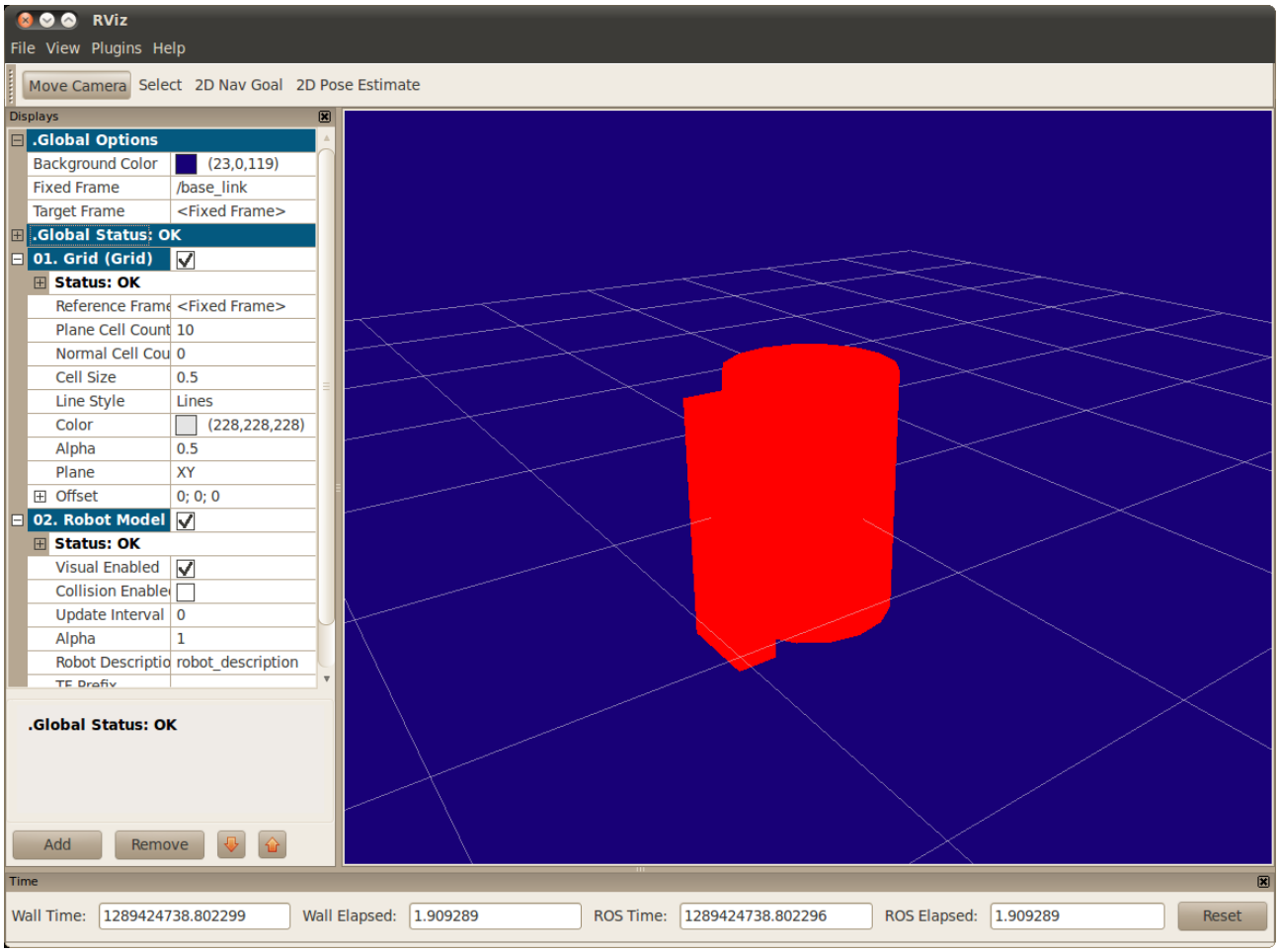
R2D2's leg attaches to the top half of his torso, on the side. So that's where we specify the origin of the JOINT to be. Also, it doesn't attach to the middle of the leg, it attaches to the upper part, so we must offset the origin for the leg as well. We also rotate the leg so it is upright. Source (https://github.com/ros/urdf_tutorial/tree/master/urdf/03-origins.urdf)

Toggle line numbers

```
<?xml version="1.0"?>
<robot name="origins">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
    <origin xyz="0 -0.22 0.25"/>
  </joint>

</robot>
```

- Let's start by examining the joint's origin. It is defined in terms of the parent's reference frame. So we are -0.22 meters in the y direction (to our left, but to the right relative to the axes) and 0.25 meters in the z direction (up). This means that the origin for the child link will be up and to the right, regardless of the child link's visual origin tag. Since we didn't specify a rpy (roll pitch yaw) attribute, the child frame will be default have the same orientation as the parent frame.
- Now, looking at the leg's visual origin, it has both a xyz and rpy offset. This defines where the center of the visual element should be, relative to its origin. Since we want the leg to attach at the top, we offset the origin down by setting the z offset to be -0.3 meters. And since we want the long part of the leg to be parallel to the z axis, we rotate the visual part PI/2 around the Y axis.

```
roslaunch urdf_tutorial display.launch model:=urdf/03-origins.urdf
```



- The launch file runs packages that will create TF frames for each link in your model based on your URDF. Rviz uses this information to figure out where to display each shape.
- If a TF frame does not exist for a given URDF link, then it will be placed at the origin in white (ref. related question (http://answers.ros.org/question/207947/how-do-you-use-externally-defined-materials-in-a-urdfxacro-file/)).

4. Material Girl

“Alright,” I hear you say. “That’s very cute, but not everyone owns a B21. My robot and R2D2 are not red!” That’s a good point. Let’s take a look at the material tag. Source (https://github.com/ros/urdf_tutorial/tree/master/urdf/04-materials.urdf)

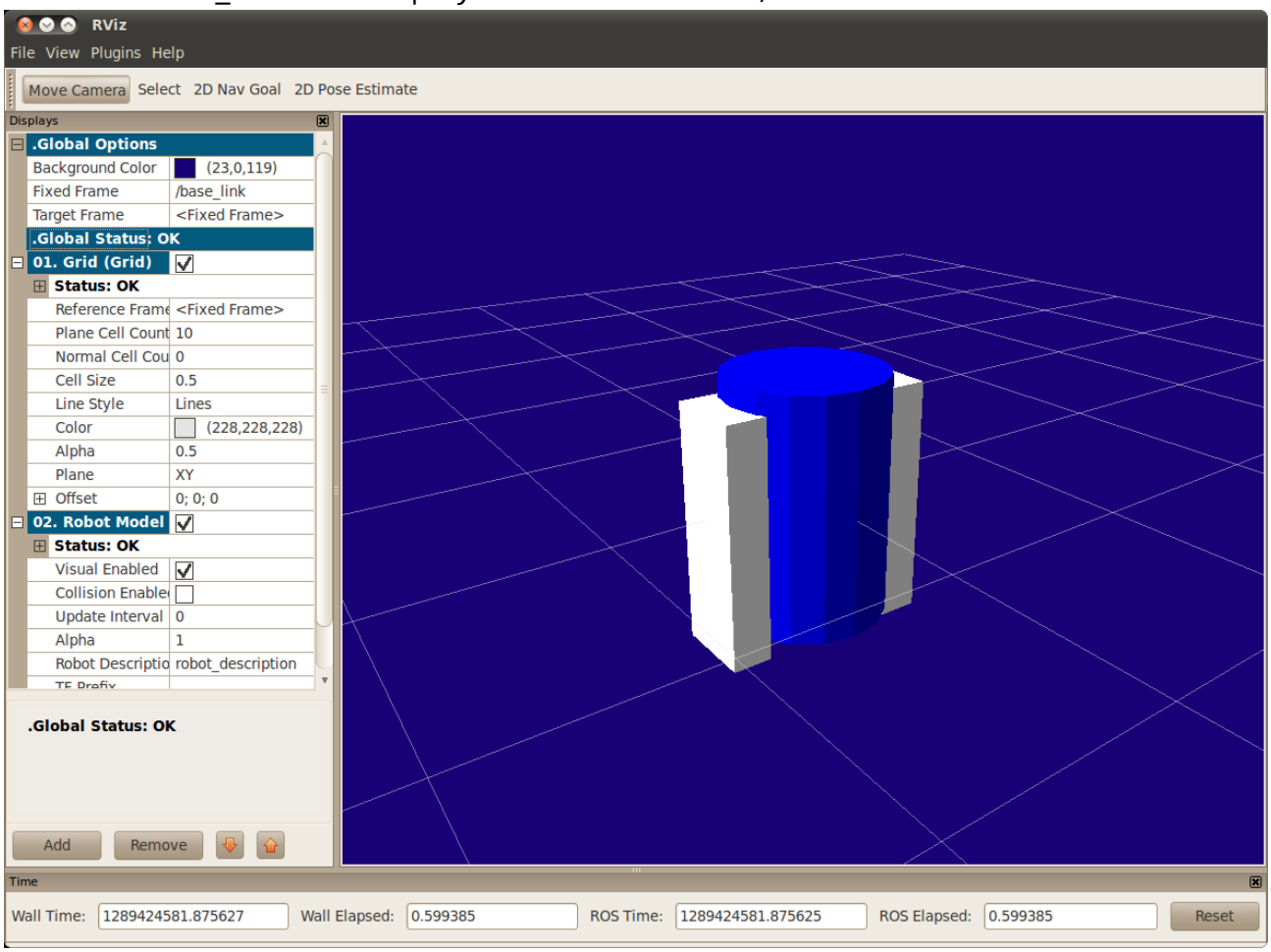
Toggle line numbers

```
<?xml version="1.0"?>
<robot name="materials">

  <material name="blue">
    <color rgba="0 0 0.8 1"/>
  </material>

  <material name="white">
    <color rgba="1 1 1 1"/>
  </material>


  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
      <material name="blue"/>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
      <material name="white"/>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
    <origin xyz="0 -0.22 0.25"/>
  </joint>

  <link name="left_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
      <material name="white"/>
    </visual>
  </link>

  <joint name="base_to_left_leg" type="fixed">
    <parent link="base_link"/>
    <child link="left_leg"/>
```

```
51      <origin xyz="0 0.22 0.25"/>
52    </joint>
53
54 </robot>
```

- The body is now blue. We've defined a new material called "blue", with the red, green, blue and alpha channels defined as 0,0,0.8 and 1 respectively. All of the values can be in the range [0,1]. This material is then referenced by the base_link's visual element. The white material is defined similarly
- You could also define the material tag from within the visual element, and even reference it in other links. No one will even complain if you redefine it though.
- You can also use a texture to specify an image file to be used for coloring the object

`roslaunch urdf_tutorial display.launch model:=urdf/04-materials.urdf`



5. Finishing the Model

Now we finish the model off with a few more shapes: feet, wheels, and head. Most notably, we add a sphere and a some meshes. We'll also add few other pieces that we'll use later. Source (https://github.com/ros/urdf_tutorial/tree/master/urdf/05-visual.urdf)

Toggle line numbers

```xml
<?xml version="1.0"?>
<robot name="visual">

  <material name="blue">
    <color rgba="0 0 0.8 1"/>
  </material>
  <material name="black">
    <color rgba="0 0 0 1"/>
  </material>
  <material name="white">
    <color rgba="1 1 1 1"/>
  </material>

  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
      <material name="blue"/>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
      <material name="white"/>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
    <origin xyz="0 -0.22 0.25"/>
  </joint>

  <link name="right_base">
    <visual>
      <geometry>
        <box size="0.4 0.1 0.1"/>
      </geometry>
      <material name="white"/>
    </visual>
  </link>

  <joint name="right_base_joint" type="fixed">
    <parent link="right_leg"/>
    <child link="right_base"/>
```

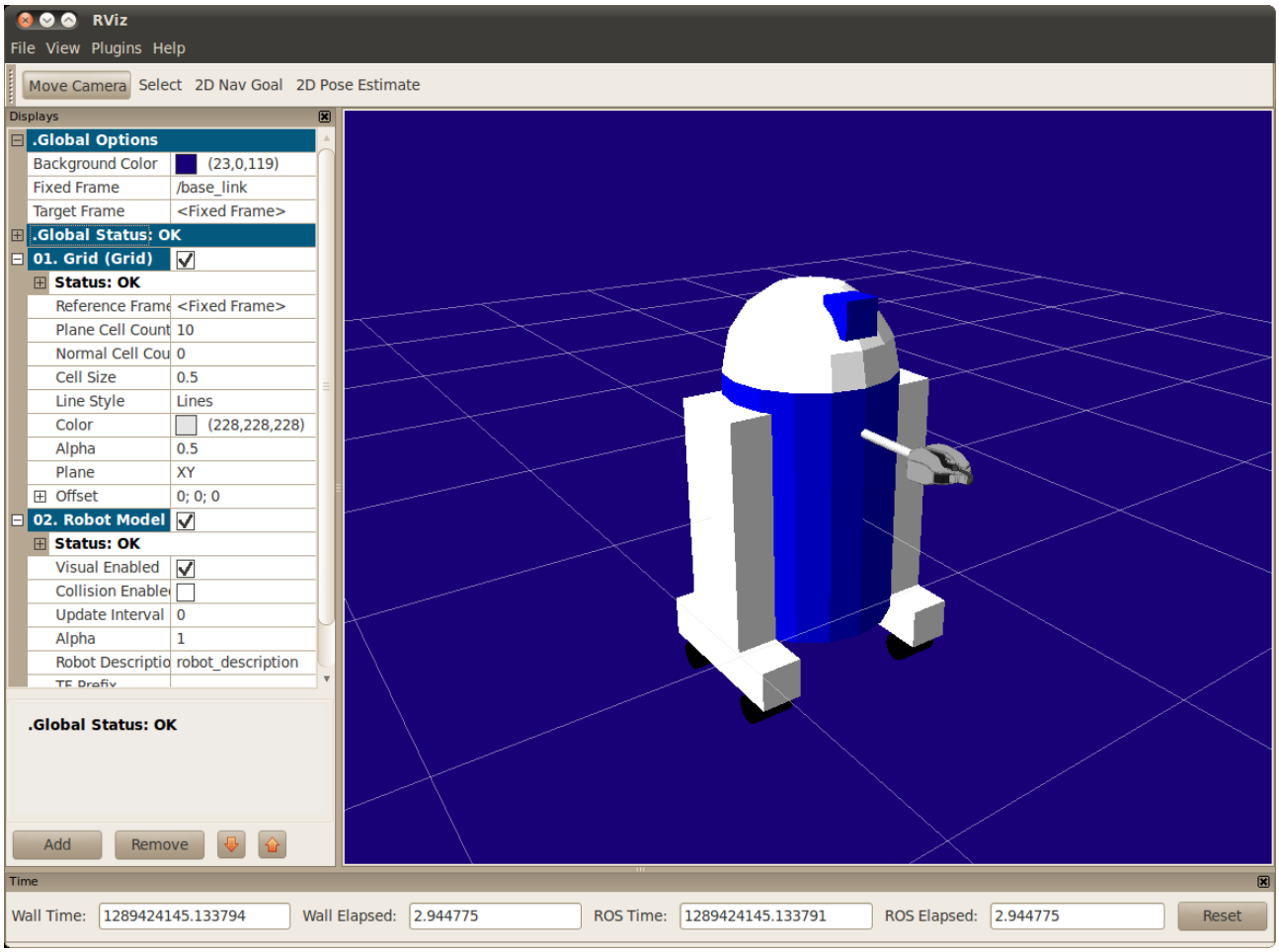
```
    <origin xyz="0 0 -0.6"/>
  </joint>

  <link name="right_front_wheel">
    <visual>
      <origin rpy="1.57075 0 0" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.1" radius="0.035"/>
      </geometry>
      <material name="black"/>
    </visual>
  </link>
  <joint name="right_front_wheel_joint" type="fixed">
    <parent link="right_base"/>
    <child link="right_front_wheel"/>
    <origin rpy="0 0 0" xyz="0.133333333333 0 -0.085"/>
  </joint>

  <link name="right_back_wheel">
    <visual>
      <origin rpy="1.57075 0 0" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.1" radius="0.035"/>
      </geometry>
      <material name="black"/>
    </visual>
  </link>
  <joint name="right_back_wheel_joint" type="fixed">
    <parent link="right_base"/>
    <child link="right_back_wheel"/>
    <origin rpy="0 0 0" xyz="-0.133333333333 0 -0.085"/>
  </joint>

  <link name="left_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
      <material name="white"/>
    </visual>
  </link>

  <joint name="base_to_left_leg" type="fixed">
    <parent link="base_link"/>
    <child link="left_leg"/>
    <origin xyz="0 0.22 0.25"/>
  </joint>

  <link name="left_base">
    <visual>
```

```
      <geometry>
        <box size="0.4 0.1 0.1"/>
      </geometry>
      <material name="white"/>
    </visual>
  </link>

  <joint name="left_base_joint" type="fixed">
    <parent link="left_leg"/>
    <child link="left_base"/>
    <origin xyz="0 0 -0.6"/>
  </joint>

  <link name="left_front_wheel">
    <visual>
      <origin rpy="1.57075 0 0" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.1" radius="0.035"/>
      </geometry>
      <material name="black"/>
    </visual>
  </link>
  <joint name="left_front_wheel_joint" type="fixed">
    <parent link="left_base"/>
    <child link="left_front_wheel"/>
    <origin rpy="0 0 0" xyz="0.133333333333 0 -0.085"/>
  </joint>

  <link name="left_back_wheel">
    <visual>
      <origin rpy="1.57075 0 0" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.1" radius="0.035"/>
      </geometry>
      <material name="black"/>
    </visual>
  </link>
  <joint name="left_back_wheel_joint" type="fixed">
    <parent link="left_base"/>
    <child link="left_back_wheel"/>
    <origin rpy="0 0 0" xyz="-0.133333333333 0 -0.085"/>
  </joint>

  <joint name="gripper_extension" type="fixed">
    <parent link="base_link"/>
    <child link="gripper_pole"/>
    <origin rpy="0 0 0" xyz="0.19 0 0.2"/>
  </joint>

  <link name="gripper_pole">
    <visual>
```

```
 153       <geometry>
 154         <cylinder length="0.2" radius="0.01"/>
 155       </geometry>
 156       <origin rpy="0 1.57075 0 " xyz="0.1 0 0"/>
 157     </visual>
 158   </link>
 159
 160   <joint name="left_gripper_joint" type="fixed">
 161     <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
 162     <parent link="gripper_pole"/>
 163     <child link="left_gripper"/>
 164   </joint>
 165
 166   <link name="left_gripper">
 167     <visual>
 168       <origin rpy="0.0 0 0" xyz="0 0 0"/>
 169       <geometry>
 170         <mesh filename="package://urdf_tutorial/meshes/l_finger.dae"/>
 171       </geometry>
 172     </visual>
 173   </link>
 174
 175   <joint name="left_tip_joint" type="fixed">
 176     <parent link="left_gripper"/>
 177     <child link="left_tip"/>
 178   </joint>
 179
 180   <link name="left_tip">
 181     <visual>
 182       <origin rpy="0.0 0 0" xyz="0.09137 0.00495 0"/>
 183       <geometry>
 184         <mesh filename="package://urdf_tutorial/meshes/l_finger_tip.da
e"/>
 185       </geometry>
 186     </visual>
 187   </link>
 188   <joint name="right_gripper_joint" type="fixed">
 189     <origin rpy="0 0 0" xyz="0.2 -0.01 0"/>
 190     <parent link="gripper_pole"/>
 191     <child link="right_gripper"/>
 192   </joint>
 193
 194   <link name="right_gripper">
 195     <visual>
 196       <origin rpy="-3.1415 0 0" xyz="0 0 0"/>
 197       <geometry>
 198         <mesh filename="package://urdf_tutorial/meshes/l_finger.dae"/>
 199       </geometry>
 200     </visual>
 201   </link>
 202
```

```
 203   <joint name="right_tip_joint" type="fixed">
 204     <parent link="right_gripper"/>
 205     <child link="right_tip"/>
 206   </joint>
 207
 208   <link name="right_tip">
 209     <visual>
 210       <origin rpy="-3.1415 0 0" xyz="0.09137 0.00495 0"/>
 211       <geometry>
 212         <mesh filename="package://urdf_tutorial/meshes/l_finger_tip.da
e"/>
 213       </geometry>
 214     </visual>
 215   </link>
 216
 217   <link name="head">
 218     <visual>
 219       <geometry>
 220         <sphere radius="0.2"/>
 221       </geometry>
 222       <material name="white"/>
 223     </visual>
 224   </link>
 225   <joint name="head_swivel" type="fixed">
 226     <parent link="base_link"/>
 227     <child link="head"/>
 228     <origin xyz="0 0 0.3"/>
 229   </joint>
 230
 231   <link name="box">
 232     <visual>
 233       <geometry>
 234         <box size="0.08 0.08 0.08"/>
 235       </geometry>
 236       <material name="blue"/>
 237     </visual>
 238   </link>
 239
 240   <joint name="tobox" type="fixed">
 241     <parent link="head"/>
 242     <child link="box"/>
 243     <origin xyz="0.1814 0 0.1414"/>
 244   </joint>
 245 </robot>
```

```
roslaunch urdf_tutorial display.launch model:=urdf/05-visual.urdf
```



How to add the sphere should be fairly self explanatory:

Toggle line numbers

```
1   <link name="head">
2     <visual>
3       <geometry>
4         <sphere radius="0.2"/>
5       </geometry>
6       <material name="white"/>
7     </visual>
8   </link>
```

The meshes here were borrowed from the PR2. They are separate files which you have to specify the path for. You should use the `package://NAME_OF_PACKAGE/path` notation. The meshes for this tutorial are located within the `urdf_tutorial` package, in a folder called meshes.

Toggle line numbers

```
<link name="left_gripper">
  <visual>
    <origin rpy="0.0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://urdf_tutorial/meshes/l_finger.dae"/>
    </geometry>
  </visual>
</link>
```

- The meshes can be imported in a number of different formats. STL is fairly common, but the engine also supports DAE, which can have its own color data, meaning you don't have to specify the color/material. Often these are in separate files. These meshes reference the `.tif` files also in the meshes folder.
- Meshes can also be sized using relative scaling parameters or a bounding box size.
- We could have also referred to meshes in a completely different package, i.e. `package://pr2_description/meshes/gripper_v0/l_finger.dae` which will work if the `pr2_description` package is installed.

There you have it. A R2D2-like URDF model. Now you can continue on to the next step, making it move (/urdf/Tutorials/Building%20a%20Movable%20Robot%20Model%20with%20URDF).

Wiki: urdf/Tutorials/Building a Visual Robot Model with URDF from Scratch (last edited 2021-10-07 18:42:21 by DavidLu (mailto:davidlu@wustl.edu))

Except where otherwise noted, the ROS wiki is licensed under the
Creative Commons Attribution 3.0 (http://creativecommons.org/licenses/by/3.0/)

Brought to you by: Open Robotics

(https://www.openrobotics.org/)