21.gazebo introduction

In the previous section, we talked about the URDF model to describe the robot. We also loaded the URDF model in rviz and used some plug-ins to make it move in rviz to achieve simulation. But often this is not enough to meet our simulation needs. We hope that the robot simulation environment can be as similar as possible to the actual environment. Then, the gazebo described in this course can meet the above needs.

21.1 Overview

Gazebo is a free robot simulation software that provides high-fidelity physical simulation, a complete set of sensor models, and a very user- and program-friendly interaction method. Functions that can accurately and efficiently simulate robot work in complex indoor and outdoor environments. By loading the model, an operating environment similar to the actual operating environment is constructed, the robot is loaded into it, and the program is run to realize simulation.

21.1.1 Install gazebo

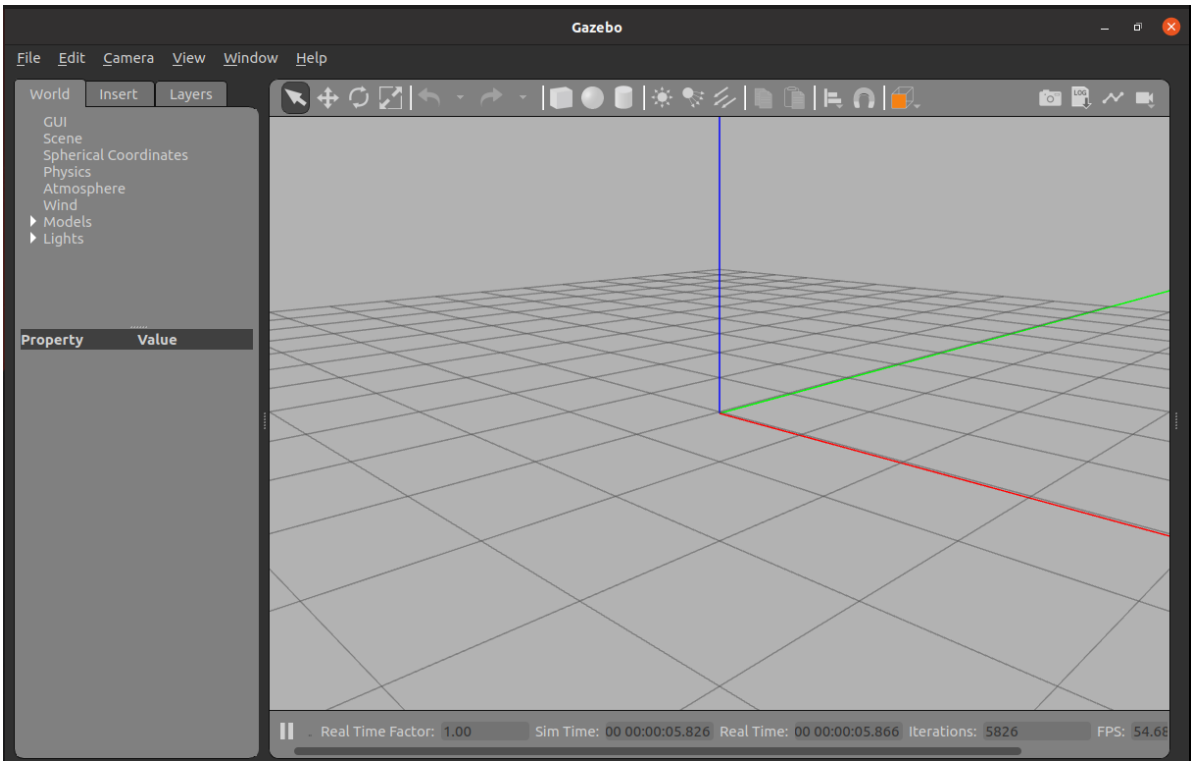
According to the installed ROS version, here is noetic as an example, input in the terminal,

```
sudo apt install ros-noetic-gazebo-* -y
sudo apt install gazebo11* -y
sudo apt install libgazebo11-dev -y
```

21.1.2 Run gazebo

After the installation is complete, enter in the terminal,

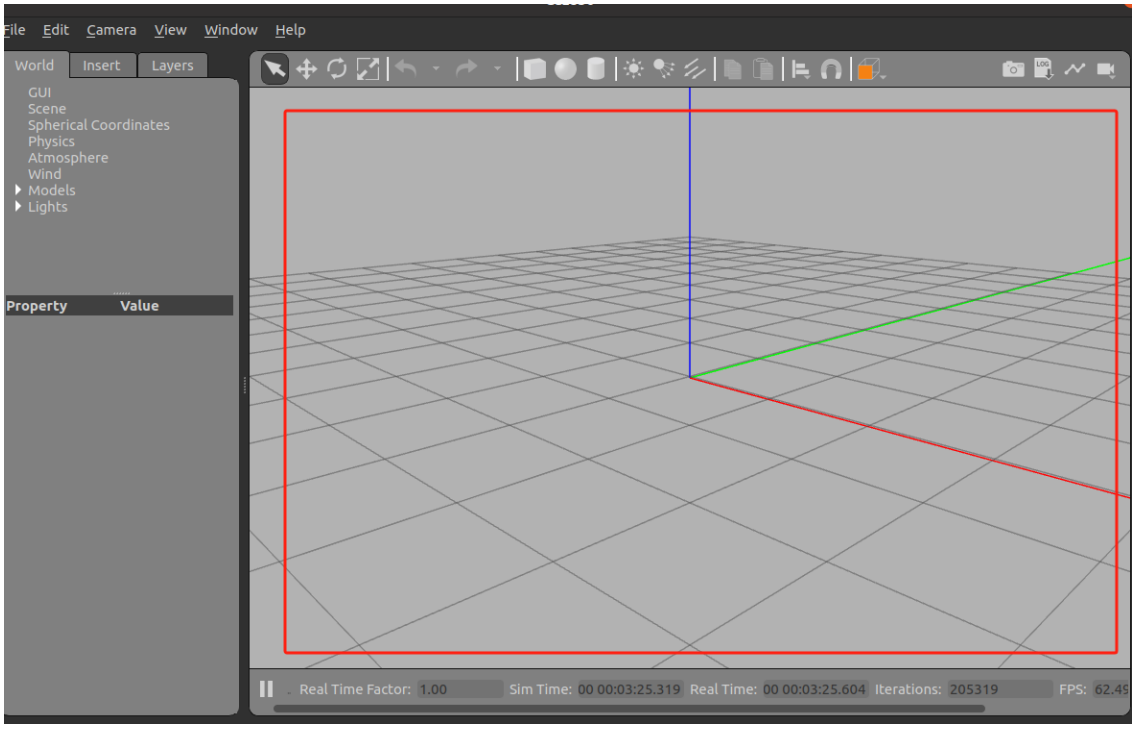
```
gazebo
```



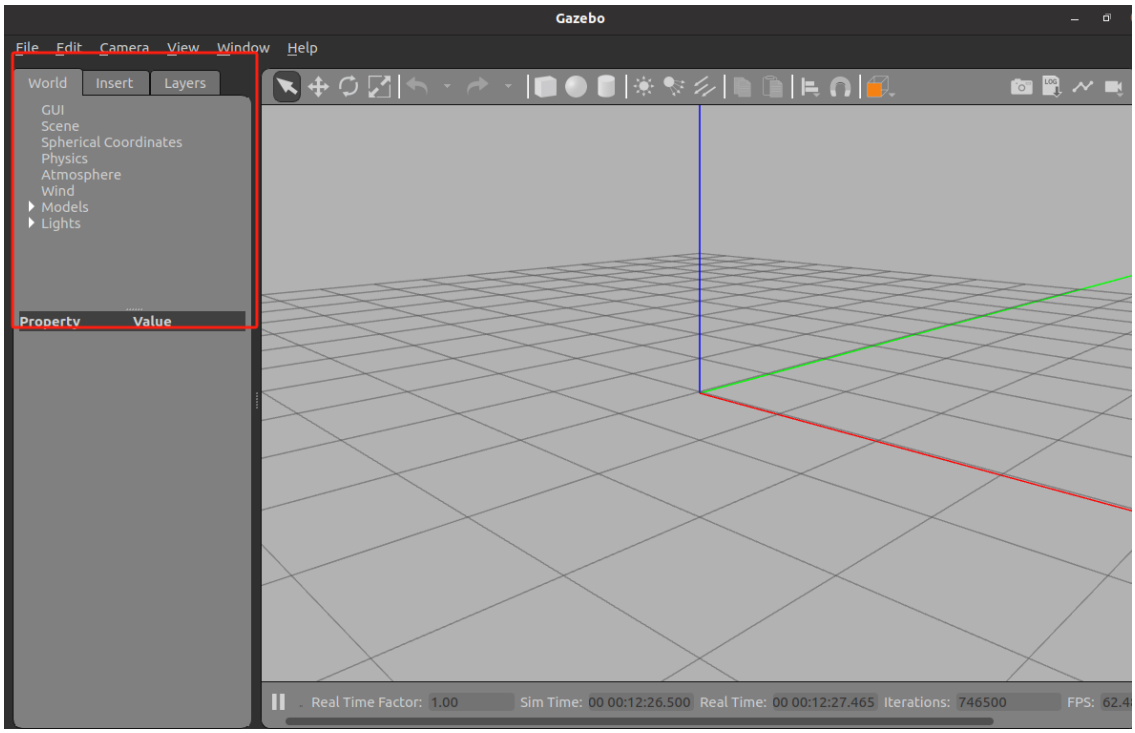
21.1.3 GUI interface introduction

- Scenes

  The scene is the main part of the simulator, where the simulation model is displayed.

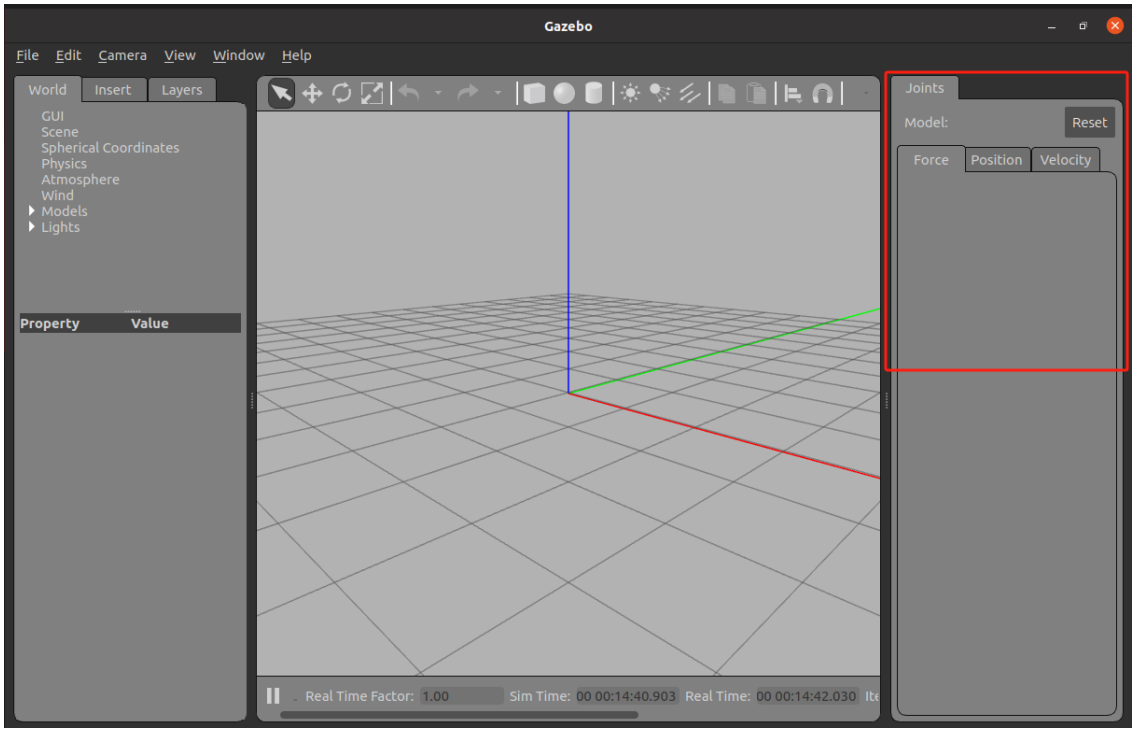


- Left and right panels
  - Left panel
    - world：Display the model currently in the scene, view and modify model parameters;
    - Insert: Add a new object (model) to the simulation. To view a list of models click the arrow to expand the folder. Click (and release) on the model you want to insert, then click again in the scene to add;
    - Layer：Organizes and displays the different visualization groups available in the simulation. Layer generally contains one or more models. Multiple models can display the model of this layer by turning it on or off.



- Right panel

  By default, the right panel is hidden. Click and drag the bar to open it. The right panel can be used to interact with the moving parts (joints) of the selected model, including pose, velocity, etc. If no model is selected in the scene, the panel will display no information.

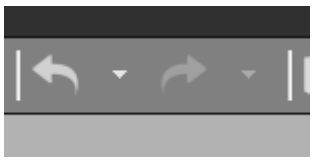


- Top toolbar

  Starting from the arrow on the left to the cube on the right, they are:

  - Select model: mark in the scene
  - Translate mode: select the model to be moved;
  - Rotate mode: select the model to be rotated;
  - Scale mode: select the model to be scaled;

  - Undo/Redo: Undo/redo operations in the scene;



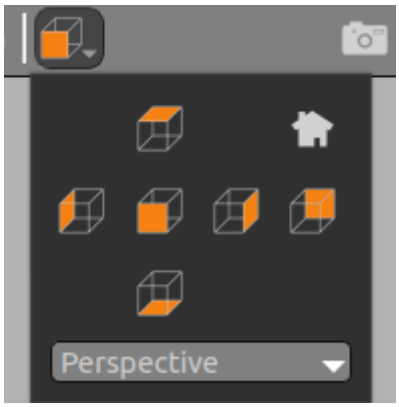
  - Create simple shapes (cube, sphere, cylinder)

  - Lights: Add lights to the scene, including the positions illuminated by light sources in all directions

- Copy/Paste: Copy/paste the model in the scene

- Align, snap models: Align models to each other, snap one model to another

- Change view: View the scene from different angles



21.2 Load urdf model in gazebo

To use urdf files with gazebo, you must add some tags specific for simulation to work properly with gazebo. For each robot model, each needs to add a tag. Let's take Yahboom's rosmaster-X3 as an example to explain how gazebo loads the urdf model.

Here are two function packages:

- yahboomcar_description：stores urdf file of rosmaster-X3
- yahboomcar_gazebo：stores files related to the started gazebo

In addition, two function packages need to be installed. One is the function package for keyboard control nodes and terminal input.

```
sudo apt install ros-noetic-teleop-twist-keyboard -y
```

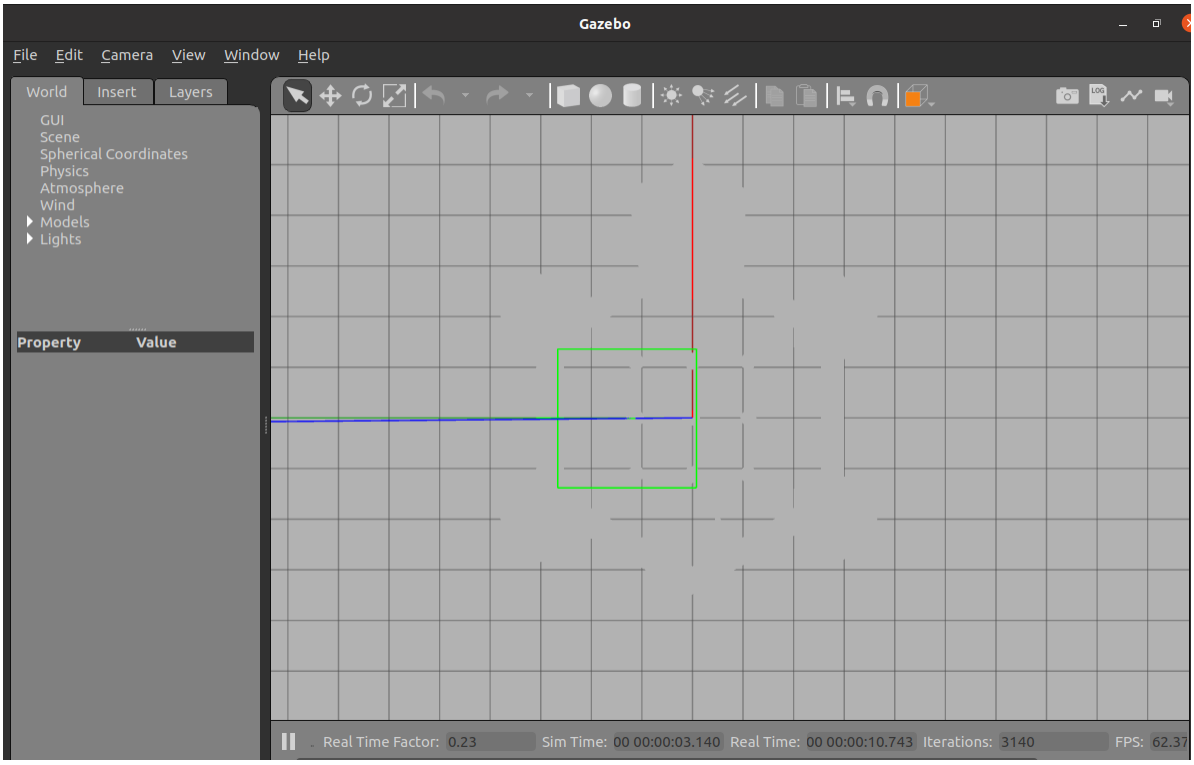
The other is the turtlebot3-gazebo function package, which contains a set gazebo environment and terminal input.

```
sudo apt install ros-noetic-turtlebot3-gazebo -y
```

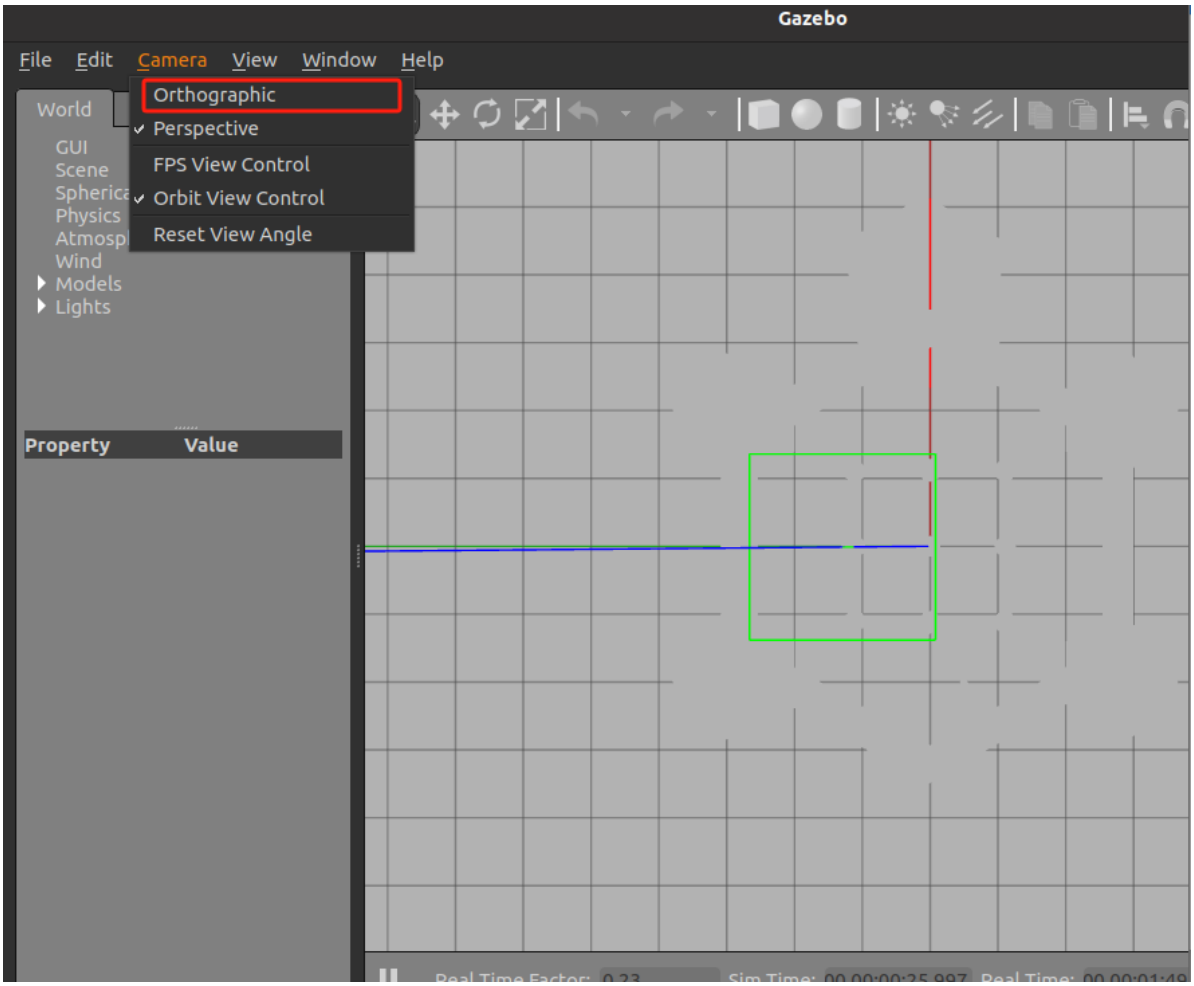
Let’s start the program first to see the effect, enter in the terminal,

```
roslaunch yahboomcar_gazebo yahboom.launch
```

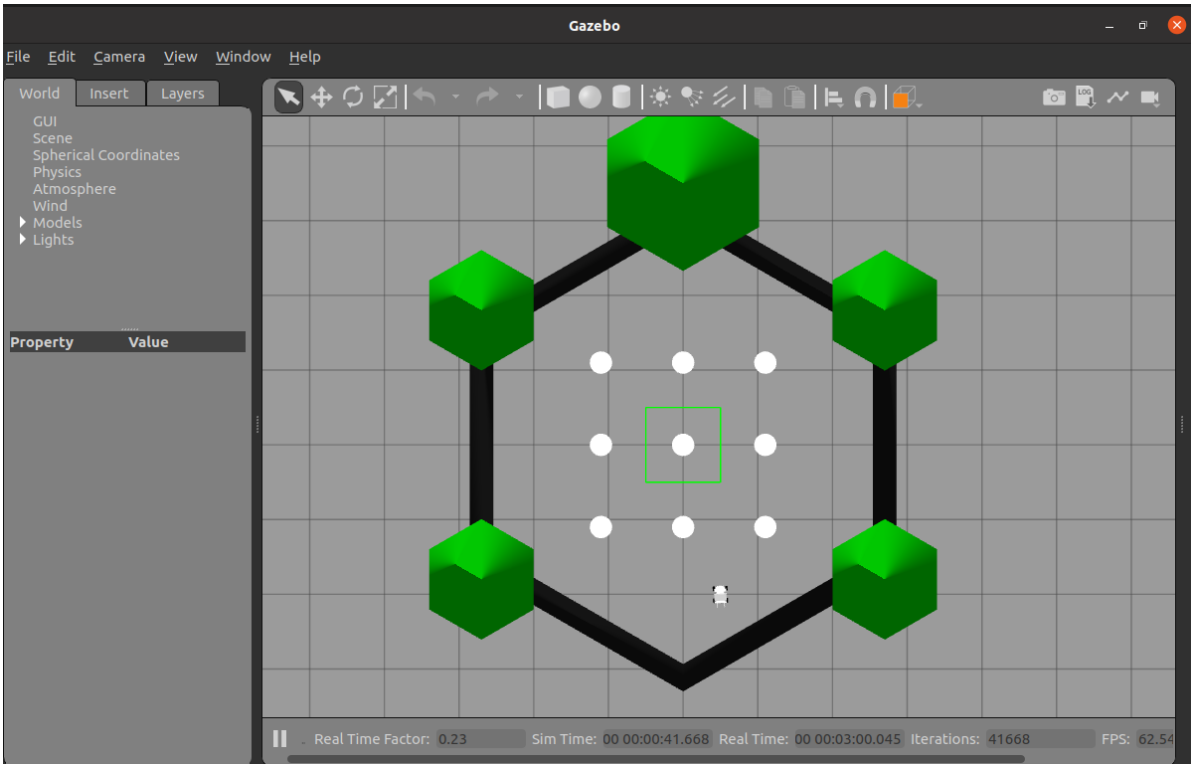
After starting, the following screen appears,



Click [Camera] above and select [Orthographic].



Then you can see the built environment, as shown below,



Then start the keyboard control node,

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

```
yahboom@yahboom-virtual-machine:~$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py
the rosdep view is empty: call 'sudo rosdep init' and 'rosdep update'

Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
```

Press [i], [u], [o], [j], [l], [m], [,], [.] to control the movement of the car, and press the space bar to stop. Since the car is equipped with a camera and radar, we can also see the image data and radar data in rviz when it is started.

21.2.1 Launch file analysis

First look at the launched launch file,

```
<?xml version="1.0"?>
<launch>
  <arg name="ns" default="robot1"/>
  <arg name="format" default="xarco" doc="xacro ; urdf"/>
```

```
  <arg name="robot_type" value="X3" />

  <arg name="gui" default="true" />
  <arg name="run_camera" default="false"/>

  <arg name="x_pos" default="-2.0"/>
  <arg name="y_pos" default="-0.5"/>
  <arg name="z_pos" default="0.0"/>

  <node name="tf_footprint_base" pkg="tf"   type="static_transform_publisher"
args="0 0 0 0 0 0 base_link base_footprint 40" />

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find
turtlebot3_gazebo)/worlds/turtlebot3_world.world"/>
    <arg name="paused" value="false"/>
    <arg name="use_sim_time" value="true"/>
    <arg name="gui" value="true"/>
    <arg name="headless" value="false"/>
    <arg name="debug" value="false"/>
  </include>


<!-- urdf xml robot description loaded on the Parameter Server, converting the
xacro into a proper urdf file-->
 <param name="robot_description" command="$(find xacro)/xacro --inorder $(find
yahboomcar_description)/urdf/yahboomcar_X3.gazebo.urdf.xacro ns:=$(arg ns)" />


    <node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf"  args="-urdf -
model yahboomcar_$(arg robot_type)  -x $(arg x_pos) -y $(arg y_pos) -z $(arg
z_pos) -param robot_description" />
      <!-- load the controllers -->
    <!--include file="$(find yahboomcar_gazebo)/launch/control.launch"></include-
->
    <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" />
 <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
yahboomcar_gazebo)/config/rviz/yahboomcar_X3.rviz"/>

</launch>
```

Mainly focus on the following points:

```
<include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find
turtlebot3_gazebo)/worlds/turtlebot3_world.world"/>
    <arg name="paused" value="false"/>
    <arg name="use_sim_time" value="true"/>
    <arg name="gui" value="true"/>
    <arg name="headless" value="false"/>
    <arg name="debug" value="false"/>
  </include>
```

Here is to start gazebo and then load the turtlebot3_world.world file, followed by some parameter settings, such as whether to use simulation time, etc. Generally, simulation time is used in gazebo, that is, use_sim_time is true.

```
 <param name="robot_description" command="$(find xacro)/xacro --inorder $(find
yahboomcar_description)/urdf/yahboomcar_X3.gazebo.urdf.xacro ns:=$(arg ns)" />
```

Describe the urdf model that needs to be loaded. We will analyze the model here later.

```
<node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf"  args="-urdf -model
yahboomcar_$(arg robot_type)  -x $(arg x_pos) -y $(arg y_pos) -z $(arg z_pos) -
param robot_description" />
```

Load the robot model in gazebo, the parameter is robot_description.

21.2.2 Model file analysis

File directory:: ~/ros_ws/src/yahboomcar_description/urdf/yahboomcar_X3.gazebo.urdf.xacro

Most of the tags of the file were mentioned in the previous lesson, so I won't go into details here. The main ones are the following parts.

```
 <xacro:include filename="$(find
yahboomcar_description)/urdf/yahboomcar_X3.gazebo.xacro"/>
```

The yahboomcar_X3.gazebo.xacro file is loaded here. The directory of this file is,

```
~/ros_ws/src/yahboomcar_description/urdf
```

Take a look at the contents of this file. This involves adding the tag and some plug-ins to each .

```
<?xml version="1.0"?>
<robot name="yahboomcar_X3" xmlns:xacro="http://ros.org/wiki/xacro">
    <gazebo reference="back_left_wheel">
        <mu1 value="2.0"/>
        <mu2 value="2.0"/>
        <kp  value="10000000.0" />
        <kd  value="1.0" />
        <fdir1 value="1 0 0"/>
        <material>Gazebo/Black</material>
    </gazebo>

    <gazebo reference="back_right_wheel">
        <mu1 value="2.0"/>
        <mu2 value="2.0"/>
        <kp  value="10000000.0" />
        <kd  value="1.0" />
        <fdir1 value="1 0 0"/>
        <material>Gazebo/Black</material>
    </gazebo>
    <gazebo reference="front_left_wheel">
        <mu1 value="2.0"/>
        <mu2 value="2.0"/>
```

```
        <kp  value="10000000.0" />
        <kd  value="1.0" />
        <fdir1 value="0 0 1"/>
        <material>Gazebo/Black</material>
    </gazebo>
        <gazebo reference="front_right_wheel">
        <mu1 value="2.0"/>
        <mu2 value="2.0"/>
        <kp  value="10000000.0" />
        <kd  value="1.0" />
        <fdir1 value="0 0 1"/>
        <material>Gazebo/Black</material>
    </gazebo>

    <gazebo>
        <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
            <robotNamespace>/</robotNamespace>
            <!--robotParam>robot_description</robotParam-->
            <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
            <legacyModeNS>true</legacyModeNS>
        </plugin>
    </gazebo>

    <gazebo>
        <plugin name="mecanum_controller"
filename="libgazebo_ros_planar_move.so">
            <robotNamespace>/</robotNamespace>
            <commandTopic>cmd_vel</commandTopic>
            <odometryTopic>odom</odometryTopic>
            <odometryFrame>odom</odometryFrame>
            <leftFrontJoint>front_left_joint</leftFrontJoint>
            <rightFrontJoint>front_right_joint</rightFrontJoint>
            <leftRearJoint>back_left_joint</leftRearJoint>
            <rightRearJoint>back_right_joint</rightRearJoint>
            <odometryRate>60.0</odometryRate>
            <robotBaseFrame>base_link</robotBaseFrame>
        </plugin>
    </gazebo>

<xacro:property name="camera_name"  value  = "camera"/>
<xacro:property name="frame_name"  value  = "camera_link"/>
<gazebo reference="camera_link">
    <!--arg name="camera_name"  default  = "mycamera"/-->
    <sensor name="abzg_camera" type="depth">
        <update_rate>30</update_rate>
        <camera>
            <horizontal_fov>1.047198</horizontal_fov>
            <image>
                <width>640</width>
                <height>480</height>
                <format>R8G8B8</format>
            </image>
            <clip>
                <near>0.05</near>
                <far>3</far>
```

```
            </clip>
        </camera>
        <plugin name="camera_link_controller"
filename="libgazebo_ros_openni_kinect.so">
            <baseline>0.2</baseline>
            <alwaysOn>true</alwaysOn>
            <updateRate>1.0</updateRate>
            <cameraName>${camera_name}_ir</cameraName>
            <imageTopicName>/${camera_name}/color/image_raw</imageTopicName>

 <cameraInfoTopicName>/${camera_name}/color/camera_info</cameraInfoTopicName>

 <depthImageTopicName>/${camera_name}/depth/image_raw</depthImageTopicName>

 <depthImageInfoTopicName>/${camera_name}/depth/camera_info</depthImageInfoTopic
Name>

 <pointCloudTopicName>/${camera_name}/depth/points</pointCloudTopicName>
            <frameName>${frame_name}</frameName>
            <pointCloudCutoff>0.5</pointCloudCutoff>
            <pointCloudCutoffMax>3.0</pointCloudCutoffMax>
            <distortionK1>0.00000001</distortionK1>
            <distortionK2>0.00000001</distortionK2>
            <distortionK3>0.00000001</distortionK3>
            <distortionT1>0.00000001</distortionT1>
            <distortionT2>0.00000001</distortionT2>
            <CxPrime>0</CxPrime>
            <Cx>0</Cx>
            <Cy>0</Cy>
            <focalLength>0</focalLength>
            <hackBaseline>0</hackBaseline>
        </plugin>
    </sensor>
</gazebo>


    <gazebo reference="laser_link">
        <sensor type="ray" name="head_hokuyo_sensor">
            <pose>0 0 0 0 0 0</pose>
            <visualize>false</visualize>
            <update_rate>40</update_rate>
            <ray>
                <scan>
                    <horizontal>
                        <samples>720</samples>
                        <resolution>1</resolution>
                        <min_angle>-1.570796</min_angle>
                        <max_angle>1.570796</max_angle>
                    </horizontal>
                </scan>
                <range>
                    <min>0.10</min>
                    <max>30.0</max>
                    <resolution>0.01</resolution>
                </range>
```

```
                <noise>
                    <type>gaussian</type>
                    <!-- Noise parameters based on published spec for Hokuyo
laser
                    achieving "+-30mm" accuracy at range < 10m.  A mean of 0.0m
and
                    stddev of 0.01m will put 99.7% of samples within 0.03m of
the true
                    reading. -->
                    <mean>0.0</mean>
                    <stddev>0.01</stddev>
                </noise>
            </ray>
            <plugin name="gazebo_ros_head_hokuyo_controller"
filename="libgazebo_ros_laser.so">
                <topicName>/scan</topicName>
                <frameName>laser_link</frameName>
            </plugin>
        </sensor>
    </gazebo>

    <gazebo reference="imu_link">
            <material>Gazebo/Black</material>
        </gazebo>

        <gazebo reference="imu_link">
            <gravity>true</gravity>
            <sensor name="imu_sensor" type="imu">
                <always_on>true</always_on>
                <update_rate>100</update_rate>
                <visualize>true</visualize>
                <topic>/imu/imu_data</topic>
                <plugin filename="libgazebo_ros_imu_sensor.so"
name="imu_plugin">
                    <topicName>/imu/imu_data</topicName>
                    <bodyName>imu_link</bodyName>
                    <updateRateHZ>100.0</updateRateHZ>
                    <gaussianNoise>0.0</gaussianNoise>
                    <xyzOffset>0 0 0</xyzOffset>
                    <rpyOffset>0 0 0</rpyOffset>
                    <frameName>imu_link</frameName>
                </plugin>
                <pose>0 0 0 0 0 0</pose>
            </sensor>
        </gazebo>
</robot>
```

First look at the link of one of the wheels.

```
<gazebo reference="back_left_wheel">
        <mu1 value="2.0"/>
        <mu2 value="2.0"/>
        <kp  value="10000000.0" />
        <kd  value="1.0" />
        <fdir1 value="1 0 0"/>
        <material>Gazebo/Black</material>
    </gazebo>
```

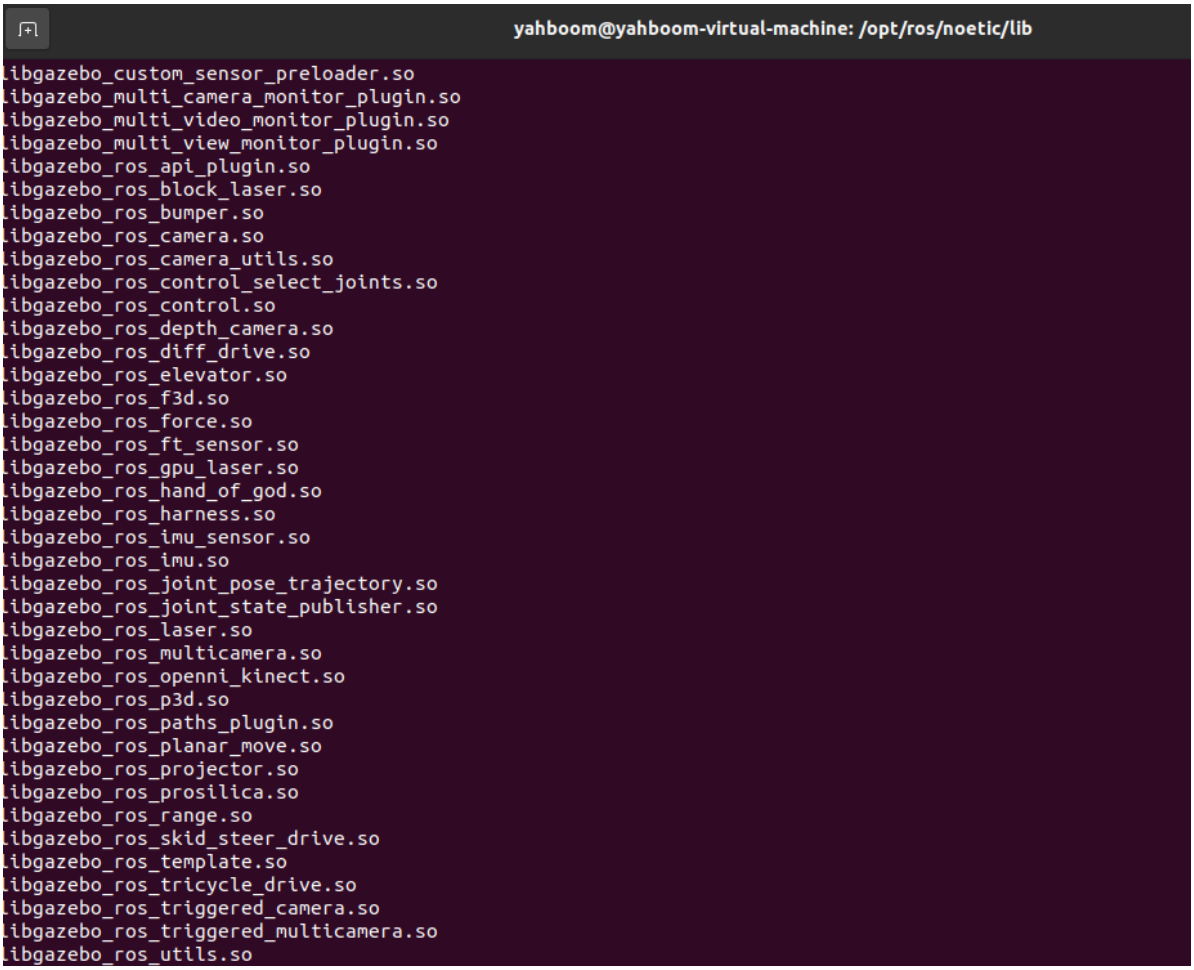
A necessary tag here is , which is used to set appearance parameters. The color set here is black, which is the color of the left rear wheel. The other parameters represent:

- mu1、mu1：friction coefficient
- kp、kd：stiffness coefficient, damping coefficient
- fdir1：direction corresponding to friction coefficient mu1

Let's take a look at one of the plug-ins:

```
<gazebo>
        <plugin name="mecanum_controller"
filename="libgazebo_ros_planar_move.so">
            <robotNamespace>/</robotNamespace>
            <commandTopic>cmd_vel</commandTopic>
            <odometryTopic>odom</odometryTopic>
            <odometryFrame>odom</odometryFrame>
            <leftFrontJoint>front_left_joint</leftFrontJoint>
            <rightFrontJoint>front_right_joint</rightFrontJoint>
            <leftRearJoint>back_left_joint</leftRearJoint>
            <rightRearJoint>back_right_joint</rightRearJoint>
            <odometryRate>60.0</odometryRate>
            <robotBaseFrame>base_link</robotBaseFrame>
        </plugin>
</gazebo>
```

Add a plug-in here. The name of the plug-in file is libgazebo_ros_planar_move.so, which is included with a pair of . There are some parameter settings inside, such as this `commandTopic`, which is set to `cmd_vel` here. Plug-in files are saved in, `/opt/ros/noetic/lib`



For additional information about the plug-in, please refer to the following URL:

[gazebo_plugins - ROS Wiki]

Let's take a look at some of the added sensors,

```
<gazebo reference="laser_link">
        <sensor type="ray" name="head_hokuyo_sensor">
            <pose>0 0 0 0 0 0</pose>
            <visualize>false</visualize>
            <update_rate>40</update_rate>
            <ray>
                <scan>
                    <horizontal>
                        <samples>720</samples>
                        <resolution>1</resolution>
                        <min_angle>-1.570796</min_angle>
                        <max_angle>1.570796</max_angle>
                    </horizontal>
                </scan>
                <range>
                    <min>0.10</min>
                    <max>30.0</max>
                    <resolution>0.01</resolution>
                </range>
                <noise>
                    <type>gaussian</type>
                    <!-- Noise parameters based on published spec for Hokuyo
laser
```

```
                    achieving "+-30mm" accuracy at range < 10m.  A mean of 0.0m
and
                    stddev of 0.01m will put 99.7% of samples within 0.03m of
the true
                    reading. -->
                    <mean>0.0</mean>
                    <stddev>0.01</stddev>
                </noise>
            </ray>
            <plugin name="gazebo_ros_head_hokuyo_controller"
filename="libgazebo_ros_laser.so">
                <topicName>/scan</topicName>
                <frameName>laser_link</frameName>
            </plugin>
        </sensor>
    </gazebo>
```

This is a sensor that adds a radar, including the radar-related parameters set in the tag, including the scanning angle, scanning depth, etc.;A plug-in libgazebo_ros_laser.so was also added later to set the radar topic and frameid.