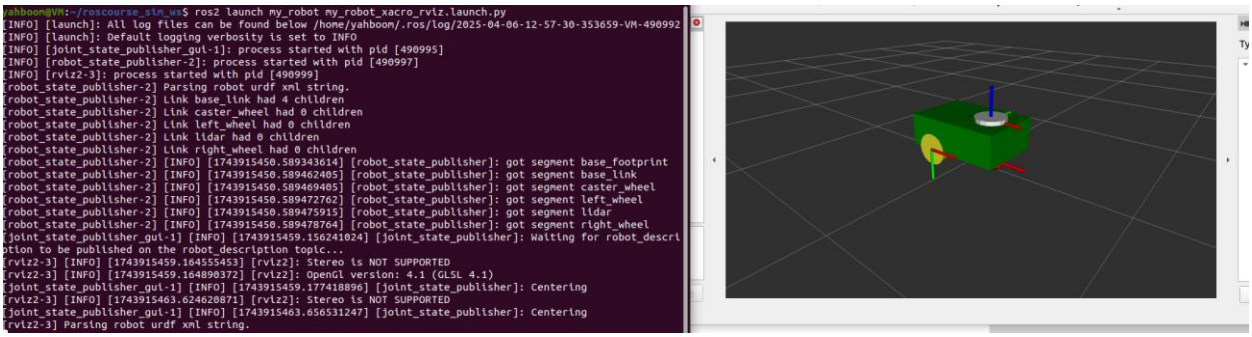
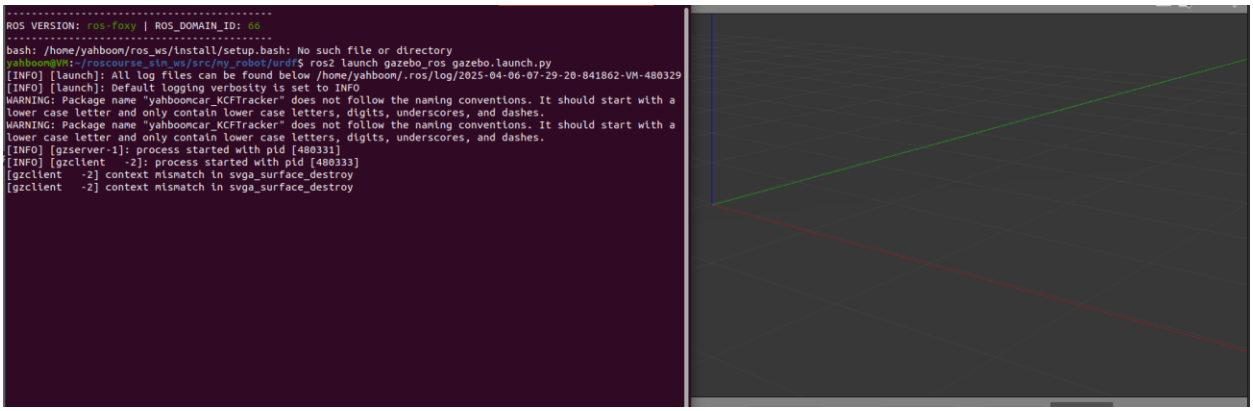
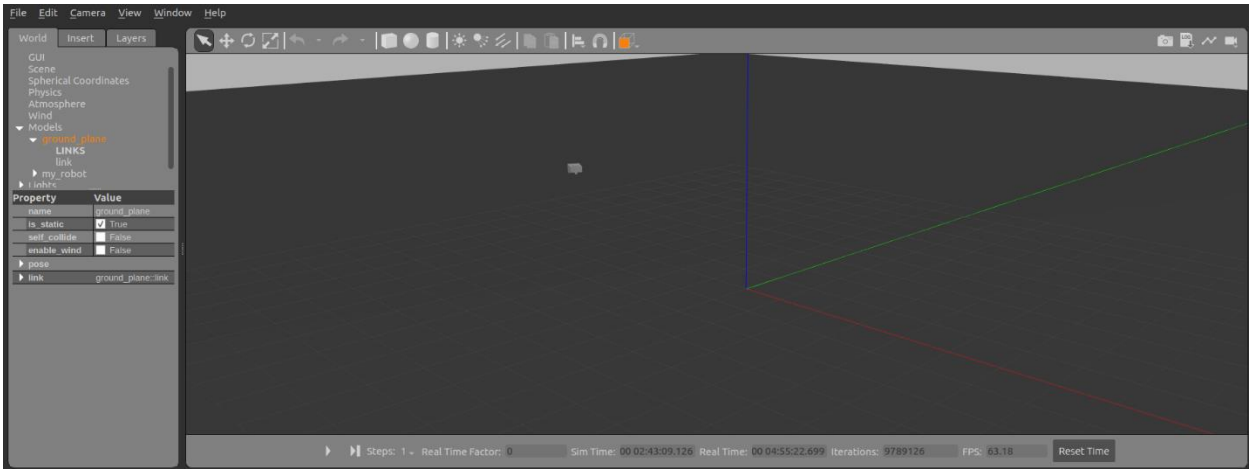
Task 1 Step 10



Task 3 Step 2

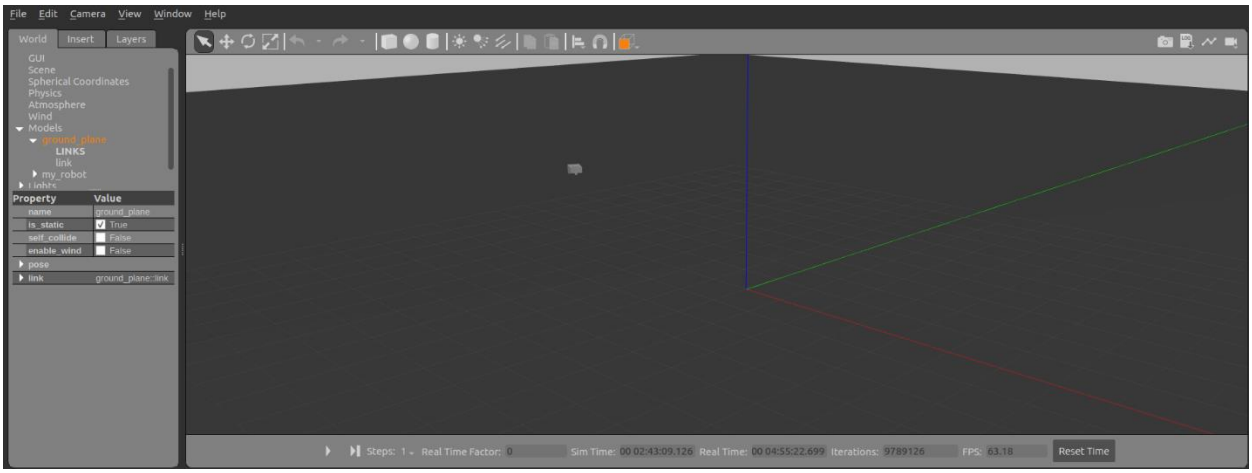


Task 3 Step 7



```
import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument, IncludeLaunchDescription
from launch.substitutions import LaunchConfiguration
from launch_ros.actions import Node
from launch.launch_description_sources import PythonLaunchDescriptionSource

def generate_launch_description():

        my_robot_path = get_package_share_directory('my_robot')
        urdf = os.path.join(my_robot_path , 'my_robot.urdf')
        rviz_config = os.path.join(my_robot_path, 'my_robot.rviz')

        use_sim_time = LaunchConfiguration('use_sim_time', default='true')

        sim_time_arg = DeclareLaunchArgument(
        name='use_sim_time',
        default_value='true', choices=['true', 'false'],
        description='Use simulation (Gazebo) clock if true')

        rviz_arg = DeclareLaunchArgument(
        name='rvizconfig',
        default_value=str(rviz_config),
        description='Absolute path to rviz config file')

        with open(urdf, 'r') as infp:
        robot_desc = infp.read()

        robot_state_publisher_node = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        name='robot_state_publisher',
        output='screen',
        parameters=[{'use_sim_time': use_sim_time, 'robot_description': robot_desc}],
        arguments=[urdf])

        rviz_node = Node(
        package='rviz2',
        executable='rviz2',
        name='rviz2',
        output='screen',
        arguments=['-d', LaunchConfiguration('rvizconfig')])

        # Gazebo launch
        gazebo_launch = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
        os.path.join(get_package_share_directory('gazebo_ros'), 'launch', 'gazebo.launch.py')
        )
        )

        gazebo_robot = Node(
        package='gazebo_ros',
        executable='spawn_entity.py',
        arguments=['-topic', 'robot_description', '-entity', 'my_robot'],
        output='screen'
        )

        return LaunchDescription([
        sim_time_arg,
        rviz_arg,
        robot_state_publisher_node,
        rviz_node,
        gazebo_launch,
        gazebo_robot
        ])
```

Task 4 Step 8



Urdf.xacro file:

```
<?xml version="1.0"?>
<robot name="my_robot" xmlns:xacro="http://www.ros.org/wiki/xacro">

    <!-- Define properties for base dimensions -->
    <xacro:property name="base_length" value="0.6" />
    <xacro:property name="base_width" value="0.4" />
    <xacro:property name="base_height" value="0.2" />

    <!-- Material definitions -->
    <material name="green">
    <color rgba="0 0.6 0 1"/>
    </material>

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

```
<material name="gray">
<color rgba="0.7 0.7 0.7 1"/>
</material>

<material name="red">
<color rgba="0.8 0.1 0.1 1"/>
</material>

<material name="blue">
<color rgba="0.1 0.1 0.8 1"/>
</material>

<material name="yellow">
<color rgba="1 1 0.2 1"/>
</material>

<!-- Macro for inertia definition for a box -->
<xacro:macro name="box_inertia" params="mass lx ly lz xyz rpy">
<inertial>
<origin xyz="${xyz}" rpy="${rpy}" />
<mass value="${mass}" />
<inertia
ixx="${(mass / 12.0) * (ly * ly + lz * lz)}"
ixy="0" ixz="0"
iyy="${(mass / 12.0) * (lx * lx + lz * lz)}"
```

```
iyz="0"
izz="${(mass / 12.0) * (lx * lx + ly * ly)}" />
</inertial>
</xacro:macro>

<!-- Macro for inertia definition for a cylinder -->
<xacro:macro name="cylinder_inertia" params="mass h r xyz rpy">
<inertial>
<origin xyz="${xyz}" rpy="${rpy}" />
<mass value="${mass}" />
<inertia
ixx="${(mass / 12.0) * (3 * r * r + h * h)}"
ixy="0" ixz="0"
iyy="${(mass / 12.0) * (3 * r * r + h * h)}"
iyz="0"
izz="${(mass / 2.0) * r * r}" />
</inertial>
</xacro:macro>

<!-- Macro for inertia definition for a sphere -->
<xacro:macro name="sphere_inertia" params="mass r xyz rpy">
<inertial>
<origin xyz="${xyz}" rpy="${rpy}" />
<mass value="${mass}" />
<inertia
ixx="${(2.0 / 5.0) * mass * r * r}"
```

```
ixy="0" ixz="0"
iyy="${(2.0 / 5.0) * mass * r * r}"
iyz="0"
izz="${(2.0 / 5.0) * mass * r * r}" />
</inertial>
</xacro:macro>

<!-- Base link -->
<link name="base_link">
<visual>
<geometry>
<box size="${base_length} ${base_width} ${base_height}" />
</geometry>
<origin xyz="0 0 ${base_height / 2.0}" rpy="0 0 0" />
<material name="green"/>
</visual>

<collision>
<geometry>
<box size="${base_length} ${base_width} ${base_height}" />
</geometry>
<origin xyz="0 0 ${base_height / 2.0}" rpy="0 0 0" />
</collision>

<xacro:box_inertia mass="5.0" lx="${base_length}" ly="${base_width}"
lz="${base_height}" xyz="0 0 ${base_height / 2.0}" rpy="0 0 0" />
```

```
</link>

<!-- Lidar -->
<link name="lidar">
<visual>
<geometry>
<cylinder radius="0.1" length="0.05" />
</geometry>
<origin xyz="0 0 0" rpy="0 0 0" />
<material name="white"/>
</visual>

<collision>
<geometry>
<cylinder radius="0.1" length="0.05" />
</geometry>
<origin xyz="0 0 0" rpy="0 0 0" />
</collision>

<xacro:cylinder_inertia mass="1.0" h="0.05" r="0.1" xyz="0 0 0" rpy="0 0 0"/>
</link>

<!-- Left wheel -->
<link name="left_wheel">
<visual>
<geometry>
```

<cylinder radius="0.1" length="0.05" />

</geometry>

<origin xyz="0 0 0.025" rpy="0 0 0" />

<material name="blue"/>

</visual>

<collision>

<geometry>

<cylinder radius="0.1" length="0.05" />

</geometry>

<origin xyz="0 0 0.025" rpy="0 0 0" />

</collision>

<xacro:cylinder_inertia mass="1.0" h="0.05" r="0.1" xyz="0 0 0.025" rpy="0 0 0" />

</link>

<!-- Right wheel -->

<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="yellow"/>

</visual>

```
<collision>
<geometry>
<cylinder radius="0.1" length="0.05" />
</geometry>
<origin xyz="0 0 0.025" rpy="0 0 0" />
</collision>

<xacro:cylinder_inertia mass="1.0" h="0.05" r="0.1" xyz="0 0 0.025" rpy="0 0 0" />
</link>

<!-- Caster wheel -->
<link name="caster_wheel">
<visual>
<geometry>
<sphere radius="0.05" />
</geometry>
<origin xyz="0 0 0" rpy="0 0 0" />
<material name="red"/>
</visual>

<collision>
<geometry>
<sphere radius="0.05" />
</geometry>
<origin xyz="0 0 0" rpy="0 0 0" />
</collision>
```

```
<xacro:sphere_inertia mass="2.0" r="0.05" xyz="0 0 0" rpy="0 0 0" />
</link>

<!-- Base footprint -->
<link name="base_footprint" />

<!-- Joints -->
<joint name="base_caster_wheel_joint" type="fixed">
<parent link="base_link"/>
<child link="caster_wheel"/>
<origin xyz="${base_length / 3.0} 0 -0.05" rpy="0 0 0"/>
</joint>

<joint name="base_lidar_joint" type="fixed">
<parent link="base_link"/>
<child link="lidar"/>
<origin xyz="${base_length / 4.0} 0 0.2" rpy="0 0 0"/>
</joint>

<joint name="base_joint" type="fixed">
<parent link="base_footprint"/>
<child link="base_link"/>
<origin xyz="0 0 0.1" rpy="0 0 0"/>
</joint>
```

```
<joint name="base_left_wheel_joint" type="continuous">
<parent link="base_link"/>
<child link="left_wheel"/>
<origin xyz="-${base_length / 4.0} 0.2 0" rpy="-1.5708 0 0"/>
<axis xyz="0 0 1"/>
</joint>

<joint name="base_right_wheel_joint" type="continuous">
<parent link="base_link"/>
<child link="right_wheel"/>
<origin xyz="-${base_length / 4.0} -0.2 0" rpy="-1.5708 0 0"/>
<axis xyz="0 0 1"/>
</joint>

<!-- Gazebo simulation tags -->
<gazebo reference="base_link">
<material>Gazebo/Green</material>
</gazebo>

<!-- Gazebo diff drive plugin -->
<gazebo>
<plugin name="diff_drive_controller" filename="libgazebo_ros_diff_drive.so">
<update_rate>50</update_rate>
<left_joint>base_left_wheel_joint</left_joint>
<right_joint>base_right_wheel_joint</right_joint>
<wheel_separation>0.4</wheel_separation>
```

```
        <wheel_diameter>0.2</wheel_diameter>
        <publish_odom>true</publish_odom>
        <publish_odom_tf>true</publish_odom_tf>
        <publish_wheel_tf>true</publish_wheel_tf>
        <odometry_topic>odom</odometry_topic>
        <odometry_frame>odom</odometry_frame>
        <robot_base_frame>base_footprint</robot_base_frame>
        <command_topic>cmd_vel</command_topic>
        </plugin>
        </gazebo>

</robot>
```