Sensing

Estimated time to completion: **12 minutes**

6.2 Lidar Plugin

The goal of this unit is to learn how to add **sensors** to your robot. This process is very similar to how the controls were added, as both use **plugins**.

Let's begin by adding a **360-degree lidar laser**.

Create a final version of URDF that has the sensors named **box_bot_control_complete_sensor.urdf**:

Execute in Terminal 1

In [ ]:
```
cd ~/ros2_ws/src
```

In [ ]:
```
cp  my_box_bot_description/urdf/box_bot_control_complete_velocity.urdf my_box_bot_description/urdf/box_bot_control_complete_sensor.urdf
```

You have to add the following code to the code you already had from the previous unit `box_bot_control_complete_velocity.urdf` :

box_bot_control_complete_sensor.urdf

In [ ]:
```
<gazebo reference="laser_scan_frame">
    <sensor name="sensor_ray" type="ray">
        <pose>0 0 0 0 0 0</pose>
        <ray>
          <scan>
            <horizontal>
              <samples>200</samples>
              <resolution>1.0</resolution>
              <min_angle>-3.14</min_angle>
              <max_angle>3.14</max_angle>
            </horizontal>
          </scan>
          <range>
            <min>0.1</min>
            <max>5.0</max>
          </range>
        </ray>
        <always_on>true</always_on>
        <visualize>true</visualize>
        <update_rate>100.0</update_rate>
        <plugin name="laser" filename="libgazebo_ros_ray_sensor.so">
          <ros>
            <namespace>/box_bot</namespace>
            <remapping>~/out:=laser_scan</remapping>
          </ros>
          <output_type>sensor_msgs/LaserScan</output_type>
        </plugin>
    </sensor>
</gazebo>
```

- Use the link named **laser_scan_frame**, connected to the **laser_scan_link** through the joint **laser_scan_frame_joint**. This joint, **laser_scan_frame_joint** can move up and down but not rotate, which you want.
- Related to the sensor **plugin and the configuration for it**. These are some elements to be noted:

- **Sensor_ray**
  - reference="laser_scan_frame" will fix the sensor to that link.
  - Set a **ray sensor type**. This is for point cloud sensors and lasers. It is based on ray-casting technology, widely used in video games.
  - Set the samples to 200 (too many will slow your PC considerably), the resolution to 1.0, and the ranges to 360 degrees (3.14 radians to the left and -3.14 radians to the right).
  - The range is important, especially the minimum because this plugin might give you false detections. Real sensors also have a minimum range of operation when too close to some collision elements.
  - Set it to visualize in Gazebo.
  - The update rate of 100.0 Hz is huge.
  - Set the plugin to **libgazebo_ros_ray_sensor.so** and set the **namespace, the output topic, and the type of message to be used.**

Create new files to launch this and see how it performs:

Execute in Terminal 1

In [ ]:
```
cd ~/ros2_ws/src
```

In [ ]:
```
touch my_box_bot_gazebo/launch/spawn_robot_ros2_sensor.launch.xml
```

In [ ]:
```
touch my_box_bot_description/launch/urdf_visualize_sensor.launch.py
```

urdf_visualize_sensor.launch.py

In [ ]:
```python
import os

from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.substitutions import Command
from launch_ros.actions import Node

# this is the function launch  system will look for
def generate_launch_description():

    ####### DATA INPUT ##########
    urdf_file = 'box_bot_control_complete_sensor.urdf'
    #xacro_file = "box_bot.xacro"
    package_description = "my_box_bot_description"

    ####### DATA INPUT END ##########
    print("Fetching URDF ==>")
    robot_desc_path = os.path.join(get_package_share_directory(package_description), "urdf", urdf_file)

    # Robot State Publisher

    robot_state_publisher_node = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        name='my_robot_state_publisher_node',
        emulate_tty=True,
        parameters=[{'use_sim_time': True, 'robot_description': Command(['xacro ', robot_desc_path])}],
        output="screen"
    )

    # RVIZ Configuration
    rviz_config_dir = os.path.join(get_package_share_directory(package_description), 'rviz', 'urdf_vis.rviz')


    rviz_node = Node(
            package='rviz2',
            executable='rviz2',
            output='screen',
            name='rviz_node',
            parameters=[{'use_sim_time': True}],
            arguments=['-d', rviz_config_dir])

    # create and return launch description object
    return LaunchDescription(
        [
            robot_state_publisher_node,
            rviz_node
        ]
    )
```

spawn_robot_ros2_sensor.launch.xml

In [ ]:
```
<?xml version='1.0' ?>
<launch>
    <!-- Publish URDF file in robot_description topic -->
    <include file="$(find-pkg-share my_box_bot_description)/launch/urdf_visualize_sensor.launch.py"/>
    <!-- Read robot_description and spawn in gazebo running sim -->
    <include file="$(find-pkg-share my_box_bot_gazebo)/launch/spawn_robot_description.launch.py"/>
    <!-- Load the controllers -->
    <include file="$(find-pkg-share my_box_bot_gazebo)/launch/control_position_velocity.launch.py"/>
</launch>
```

Execute in Terminal 1

In [ ]:
```
cd ~/ros2_ws; colcon build; source install/setup.bash
```

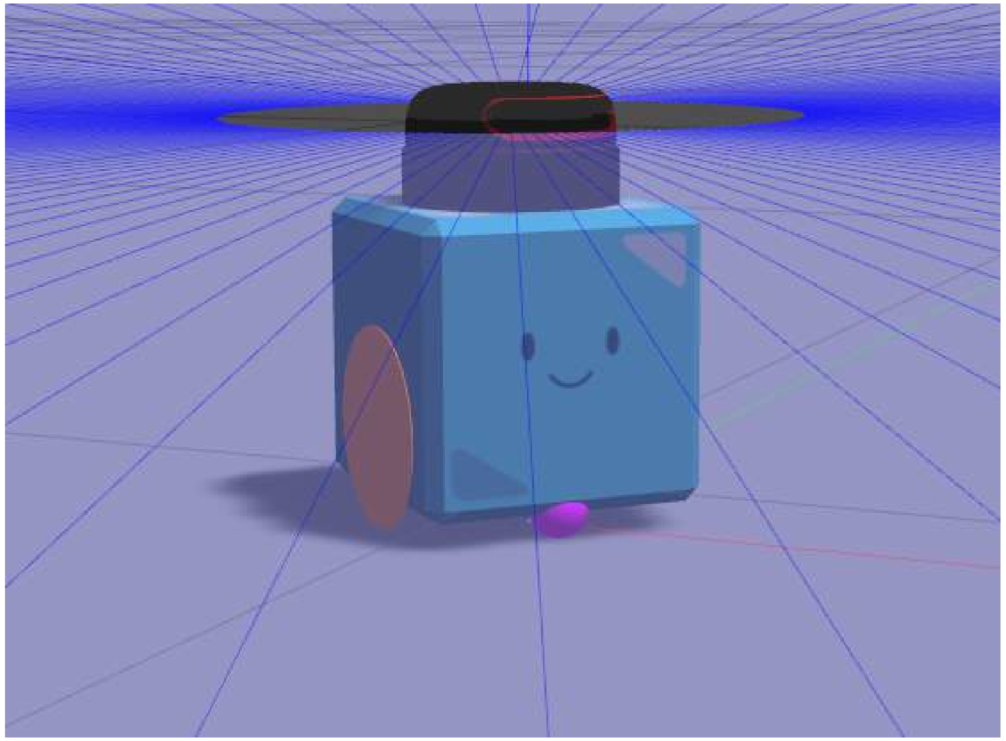
In [ ]:
```
ros2 launch my_box_bot_gazebo start_world.launch.py
```

Execute in Terminal 2

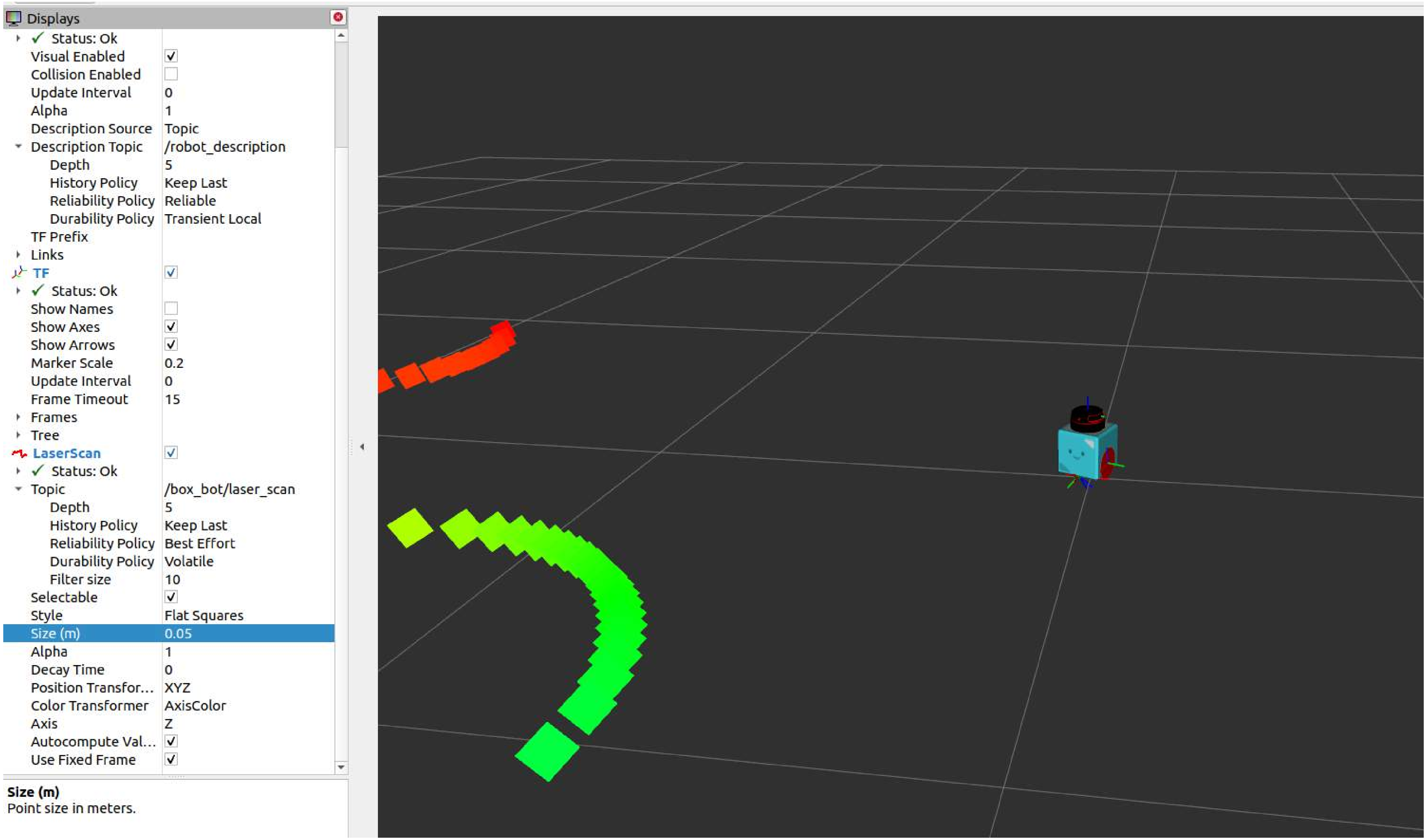
In [ ]:
```
cd ~/ros2_ws; source install/setup.bash
```

In [ ]:
```
ros2 launch my_box_bot_gazebo spawn_robot_ros2_sensor.launch.xml
```

You should see something like this in **Gazebo**:



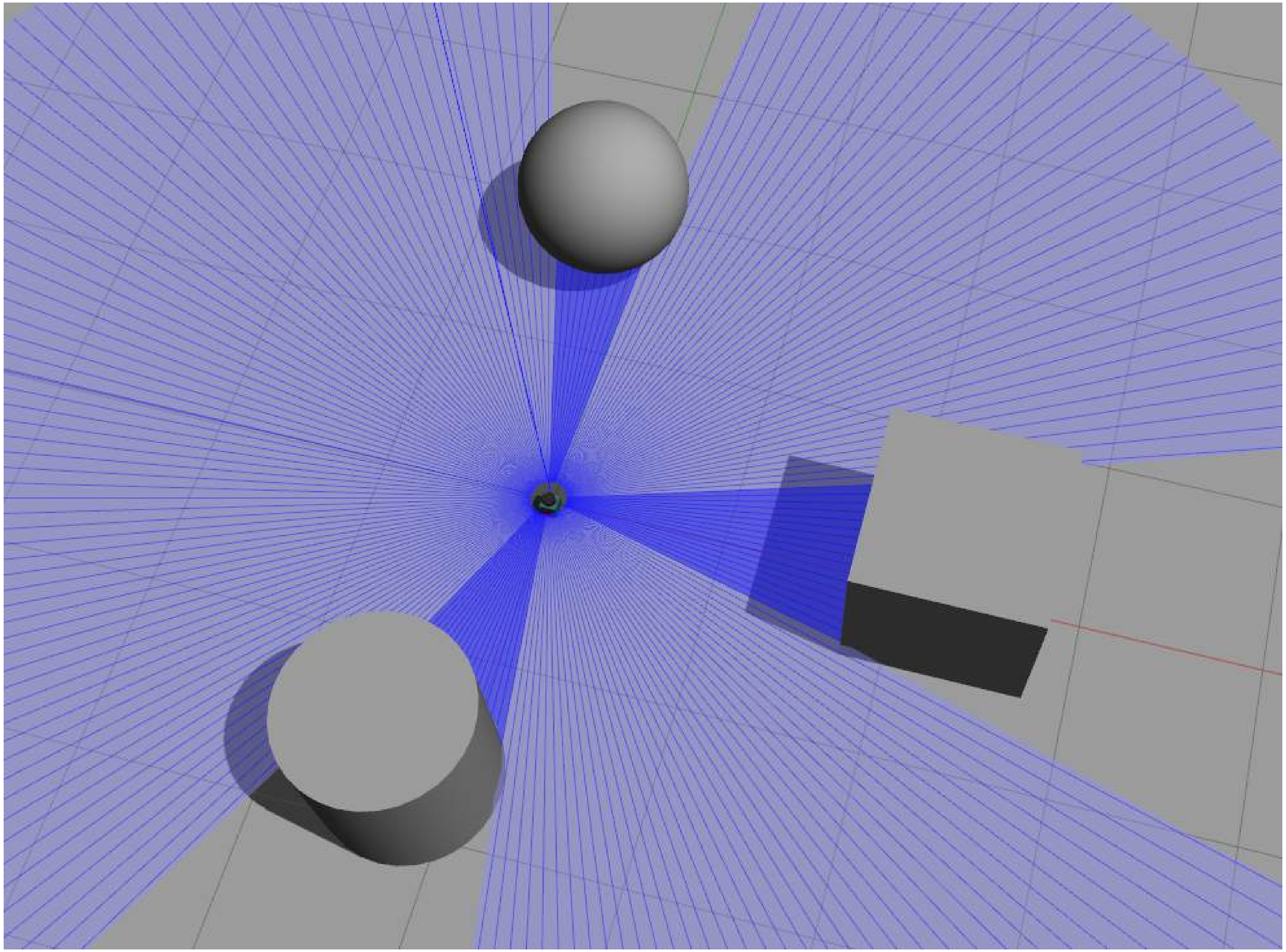
In RVIZ, add the **laser element**, select the topic **/box_bot/laser_scan**, and **VERY IMPORTANT**, select the correct **QoS**; in this case, **Reliability = Best effort**.

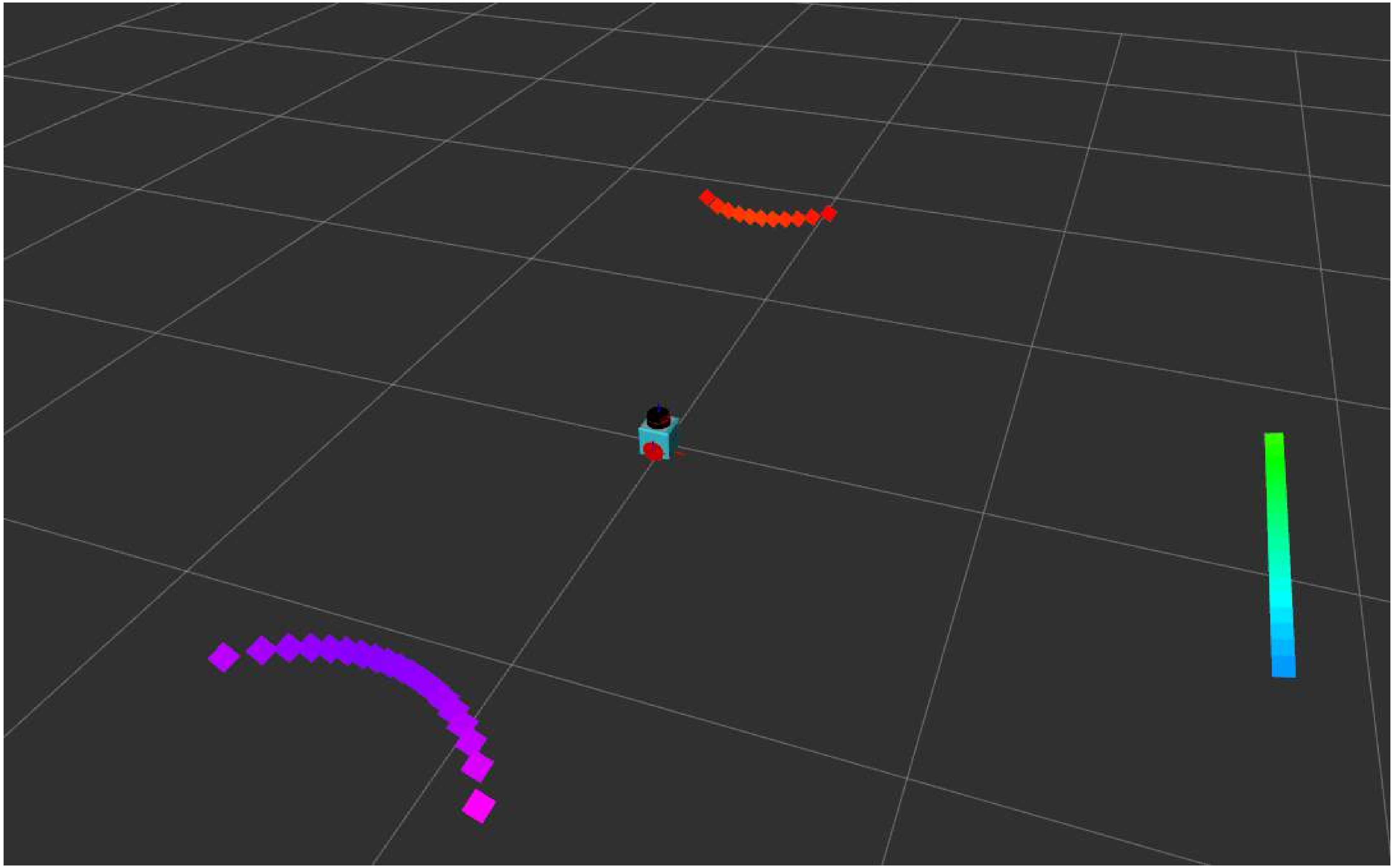


✓ Status: Ok	
Visual Enabled	✓
Collision Enabled	
Update Interval	0
Alpha	1
Description Source	Topic
Description Topic	/robot_description
Depth	5
History Policy	Keep Last
Reliability Policy	Reliable
Durability Policy	Transient Local
TF Prefix	
Links	
TF	✓
✓ Status: Ok	
Show Names	
Show Axes	✓
Show Arrows	✓
Marker Scale	0.2
Update Interval	0
Frame Timeout	15
Frames	
Tree	
LaserScan	✓
✓ Status: Ok	
Topic	/box_bot/laser_scan
Depth	5
History Policy	Keep Last
Reliability Policy	Best Effort
Durability Policy	Volatile
Filter size	10
Selectable	✓
Style	Flat Squares
Size (m)	0.05
Alpha	1
Decay Time	0
Position Transfor...	XYZ
Color Transformer	AxisColor
Axis	Z
Autocompute Val...	✓
Use Fixed Frame	✓

Size (m)
Point size in meters.

You can add basic models through the **Gazebo GUI** and see that the laser is working properly:





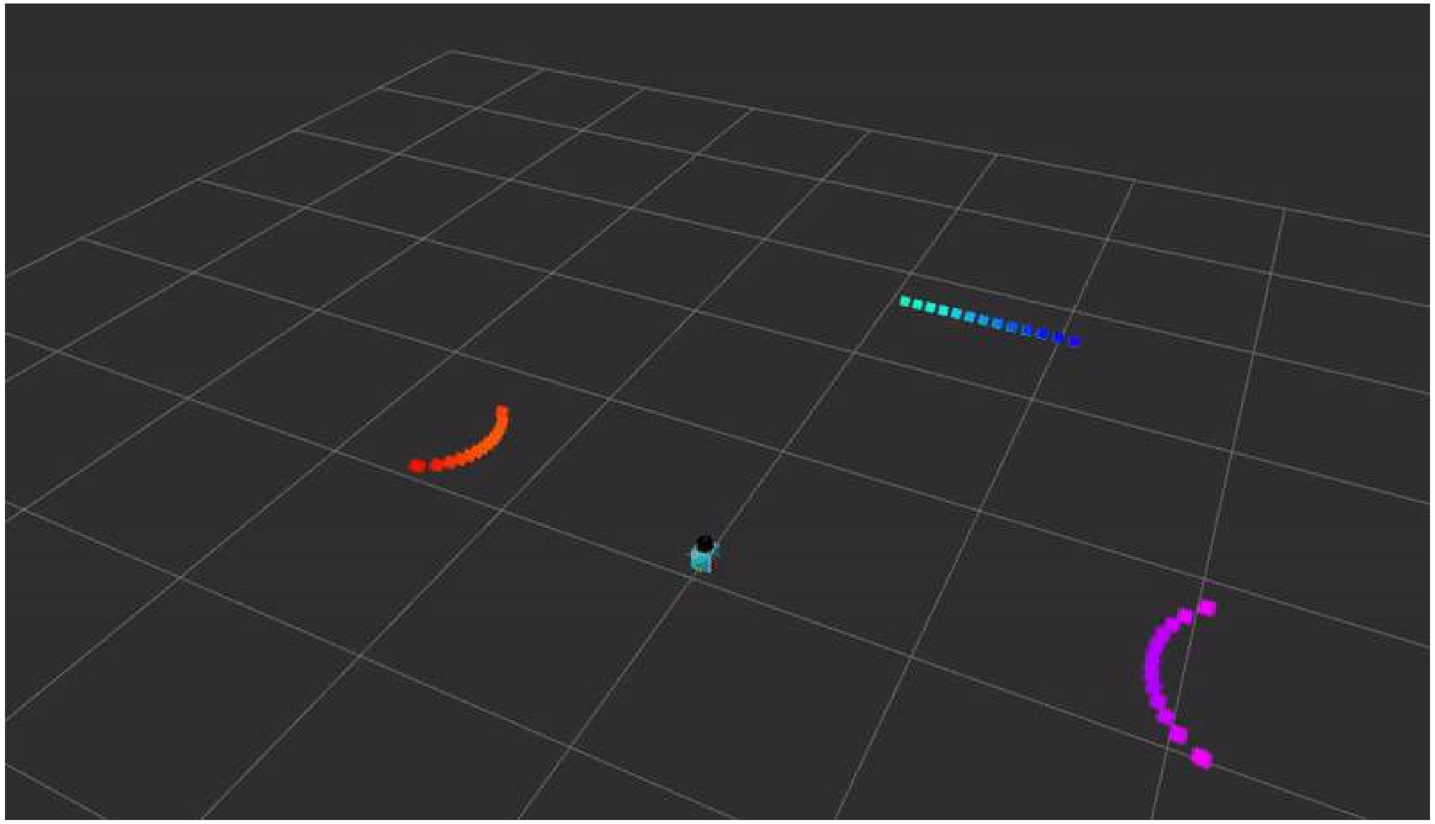
And move around to see this in action:

Execute in Terminal 3

In [ ]:
```
cd ~/ros2_ws; source install/setup.bash
```

In [ ]:
```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

You should see something similar to this:



16/11/2023