ROS2 Experimenting with a dummy robot

github code: https://github.com/gbrlb/dummy_robot_test

Tasks:

- 1. Use the dummy robot
- 2. Load it into the rviz2
- 3. Read position and orientation data of one of the robot joints
- 4. Create a node that subscribe the robot joint state and publishes one of the joints state
- 5. Create node send movement to the robot

1. Use the dummy robot

Follow the demo Experimenting with a dummy robot

```
ros2 launch dummy_robot_bringup dummy_robot_bringup2.launch.py
```

Ouput

```
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [dummy_map_server-1]: process started with pid [39586]
[INFO] [robot_state_publisher-2]: process started with pid [39588]
[INFO] [dummy_joint_states-3]: process started with pid [39590]
[INFO] [dummy_laser-4]: process started with pid [39594]
[robot_state_publisher-2] Parsing robot urdf xml string.
[robot_state_publisher-2] Link single_rrbot_link1 had 1 children
[robot_state_publisher-2] Link single_rrbot_link2 had 1 children
[robot_state_publisher-2] Link single_rrbot_link3 had 2 children
[robot_state_publisher-2] Link single_rrbot_camera_link had 0 children
[robot_state_publisher-2] Link single_rrbot_hokuyo_link had 0 children
[robot_state_publisher-2] [INFO] [1683713161.333407427]
[robot_state_publisher]: got segment single_rrbot_camera_link
[robot_state_publisher-2] [INFO] [1683713161.334157666]
[robot_state_publisher]: got segment single_rrbot_hokuyo_link
[robot_state_publisher-2] [INFO] [1683713161.334922421]
[robot_state_publisher]: got segment single_rrbot_link1
[robot_state_publisher-2] [INFO] [1683713161.335373247]
[robot_state_publisher]: got segment single_rrbot_link2
[robot_state_publisher-2] [INFO] [1683713161.335704788]
[robot_state_publisher]: got segment single_rrbot_link3
[robot_state_publisher-2] [INFO] [1683713161.335988927]
```

```
[robot_state_publisher]: got segment world
[dummy_laser-4] [INFO] [1683713161.408761680] [dummy_laser]: angle inc:
0.004363
[dummy_laser-4] [INFO] [1683713161.409600929] [dummy_laser]: scan size:
1081
[dummy_laser-4] [INFO] [1683713161.410023628] [dummy_laser]: scan time
increment:      0.000028
```

```
ros2 node list
```

Output

```
/dummy_joint_states
/dummy_laser
/dummy_map_server
/robot_state_publisher
```

```
ros2 topic list
```

Ouput

```
/joint_states
/map
/parameter_events
/robot_description
/rosout
/scan
/tf
/tf_static
```

```
ros2 topic list -t
```

Ouput

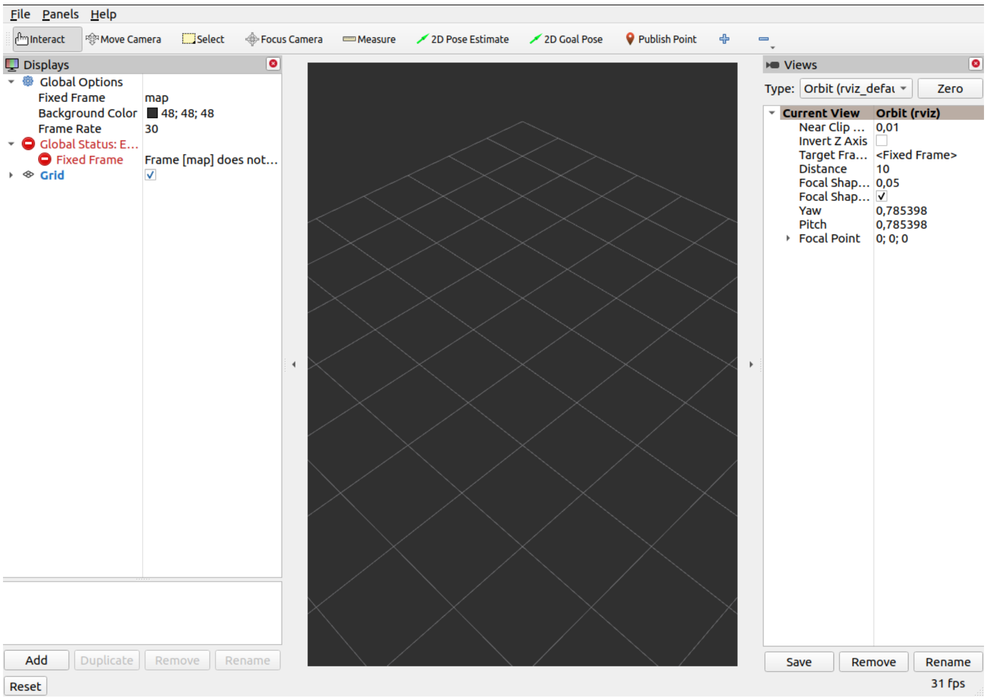
```
/clicked_point [geometry_msgs/msg/PointStamped]
/goal_pose [geometry_msgs/msg/PoseStamped]
/initialpose [geometry_msgs/msg/PoseWithCovarianceStamped]
/joint_states [sensor_msgs/msg/JointState]
/map [nav_msgs/msg/OccupancyGrid]
/map_updates [map_msgs/msg/OccupancyGridUpdate]
/parameter_events [rcl_interfaces/msg/ParameterEvent]
/robot_description [std_msgs/msg/String]
```

```
/rosout [rcl_interfaces/msg/Log]
/scan [sensor_msgs/msg/LaserScan]
/tf [tf2_msgs/msg/TFMessage]
/tf_static [tf2_msgs/msg/TFMessage]
```

2. Load it into the rviz2

To open opens Rviz2 run in another terminal `rviz2` or

```
ros2 run rviz2 rviz2
```



Displays

Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
Global Status: E...	
Fixed Frame	Frame [map] does not...
Grid	☑

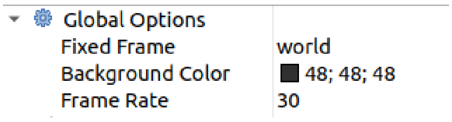
Views

Type: Orbit (rviz_defau

Current View	Orbit (rviz)
Near Clip ...	0,01
Invert Z Axis	□
Target Fra...	<Fixed Frame>
Distance	10
Focal Shap...	0,05
Focal Shap...	☑
Yaw	0,785398
Pitch	0,785398
Focal Point	0; 0; 0

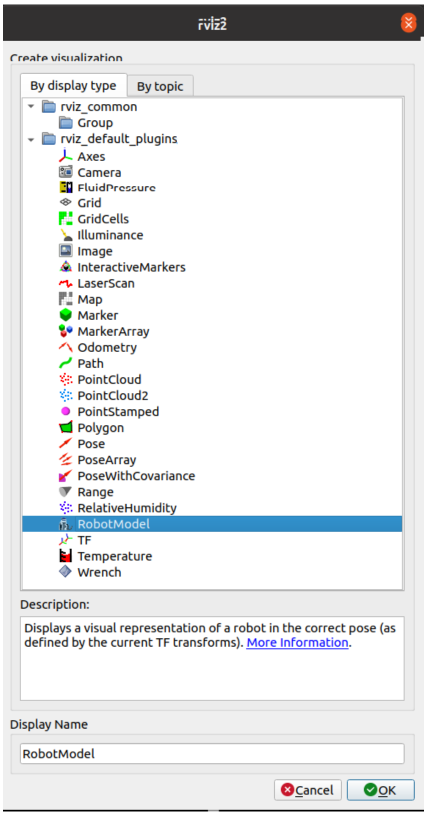
31 fps

1. Change `Global Options\Fixed Frame` map -> world:



Global Options	
Fixed Frame	world
Background Color	■ 48; 48; 48
Frame Rate	30

2. Add `RobotModel` and `TF` visualizations.



rviz2

Create visualization

By display type | By topic

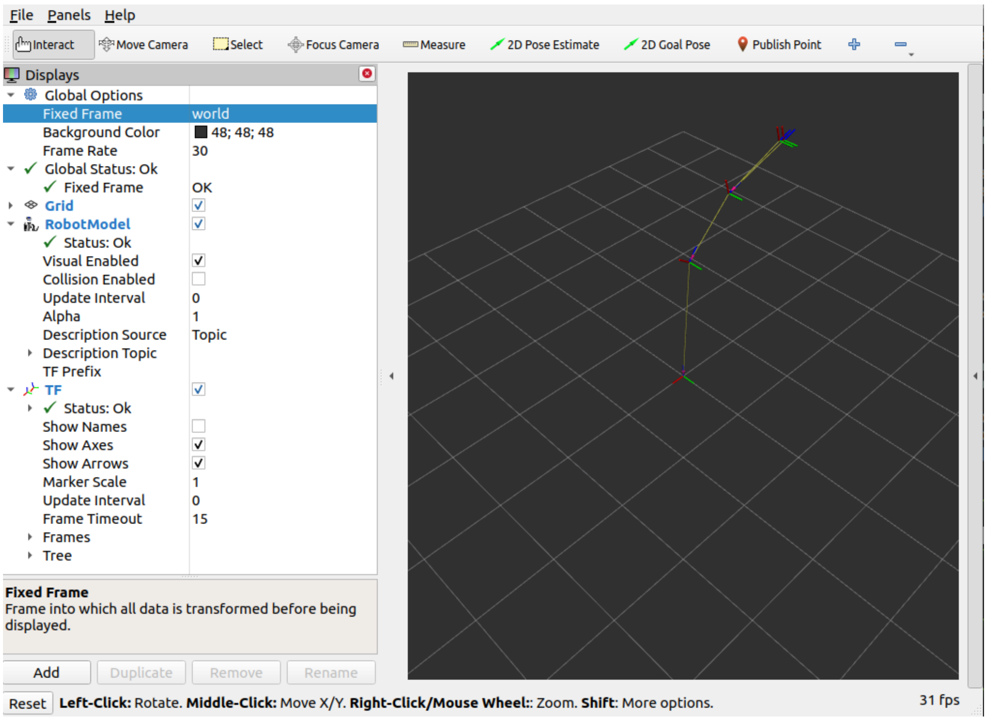
- rviz_common
  - Group
- rviz_default_plugins
  - Axes
  - Camera
  - FluidPressure
  - Grid
  - GridCells
  - Illuminance
  - Image
  - InteractiveMarkers
  - LaserScan
  - Map
  - Marker
  - MarkerArray
  - Odometry
  - Path
  - PointCloud
  - PointCloud2
  - PointStamped
  - Polygon
  - Pose
  - PoseArray
  - PoseWithCovariance
  - Range
  - RelativeHumidity
  - RobotModel
  - TF
  - Temperature
  - Wrench

Description:

Displays a visual representation of a robot in the correct pose (as defined by the current TF transforms). More Information.

Display Name

RobotModel



Displays

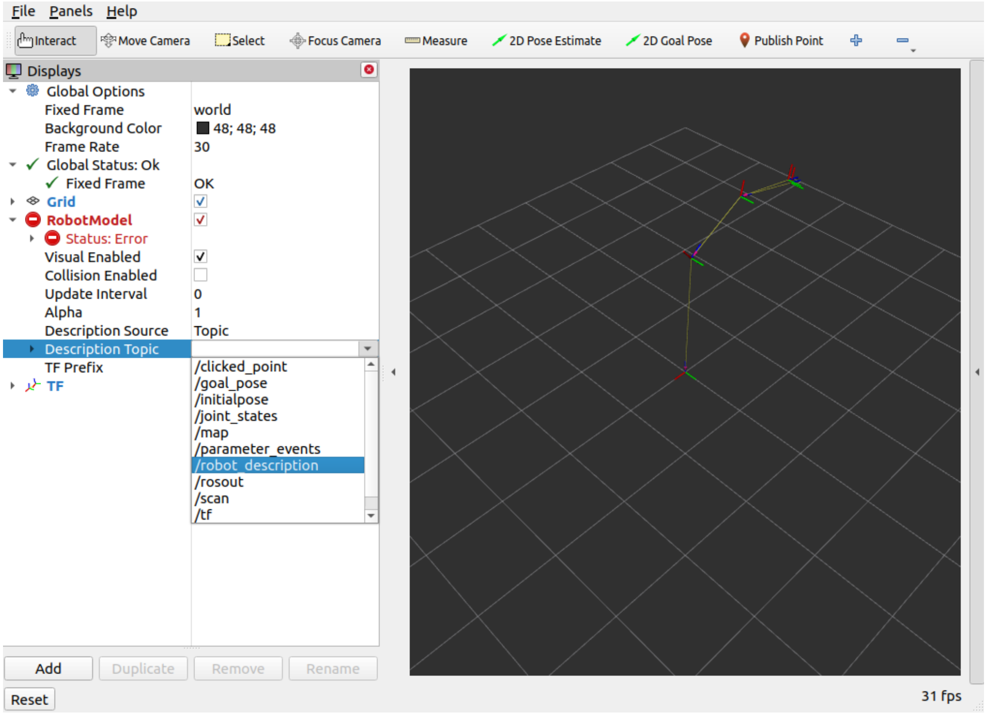
Global Options	
Fixed Frame	world
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	☑
RobotModel	☑
✓ Status: Ok	
Visual Enabled	☑
Collision Enabled	□
Update Interval	0
Alpha	1
Description Source	Topic
Description Topic	
TF Prefix	
TF	☑
✓ Status: Ok	
Show Names	□
Show Axes	☑
Show Arrows	☑
Marker Scale	1
Update Interval	0
Frame Timeout	15
Frames	
Tree	

Fixed Frame
Frame into which all data is transformed before being displayed.

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

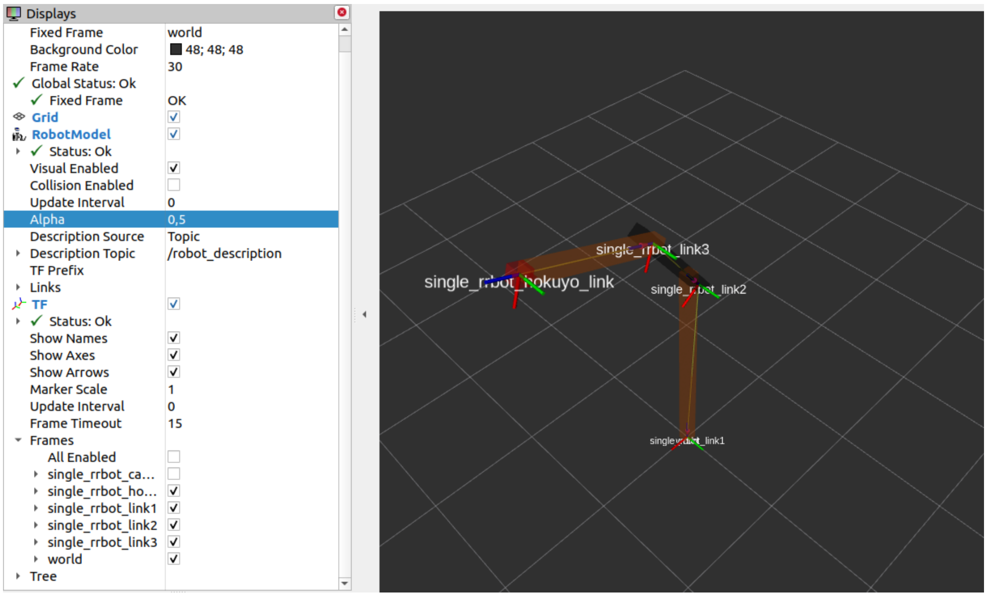
31 fps

3. Change `RobotModel/Description Topic` to `/robot_description`



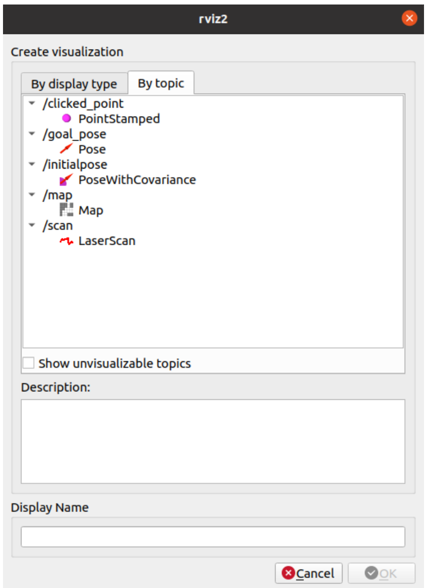
Displays
- Global Options
  - Fixed Frame: world
  - Background Color: 48; 48; 48
  - Frame Rate: 30
- ✓ Global Status: Ok
  - ✓ Fixed Frame: OK
- Grid ☑
- RobotModel ☑
  - Status: Error
  - Visual Enabled ☑
  - Collision Enabled □
  - Update Interval: 0
  - Alpha: 1
  - Description Source: Topic
  - Description Topic
  - TF Prefix
- TF

/clicked_point
/goal_pose
/initialpose
/joint_states
/map
/parameter_events
/robot_description
/rosout
/scan
/tf



Displays
- Fixed Frame: world
- Background Color: 48; 48; 48
- Frame Rate: 30
- ✓ Global Status: Ok
  - ✓ Fixed Frame: OK
- Grid ☑
- RobotModel ☑
  - ✓ Status: Ok
  - Visual Enabled ☑
  - Collision Enabled □
  - Update Interval: 0
  - Alpha: 0,5
  - Description Source: Topic
  - Description Topic: /robot_description
  - TF Prefix
  - Links
- TF ☑
  - ✓ Status: Ok
  - Show Names ☑
  - Show Axes ☑
  - Show Arrows ☑
  - Marker Scale: 1
  - Update Interval: 0
  - Frame Timeout: 15
  - Frames
    - All Enabled □
    - single_rrbot_ca… □
    - single_rrbot_ho… ☑
    - single_rrbot_link1 ☑
    - single_rrbot_link2 ☑
    - single_rrbot_link3 ☑
    - world ☑
  - Tree

4. Add `/map/Map` and `/scan/LaserScan` visualization.



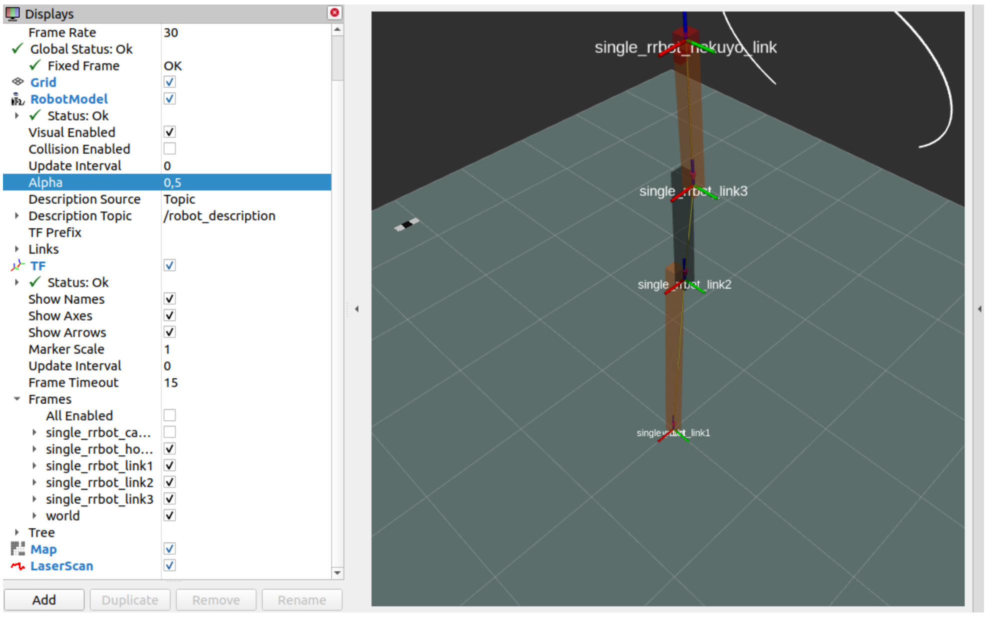
Create visualization

- /clicked_point
  - PointStamped
- /goal_pose
  - Pose
- /initialpose
  - PoseWithCovariance
- /map
  - Map
- /scan
  - LaserScan

□ Show unvisualizable topics

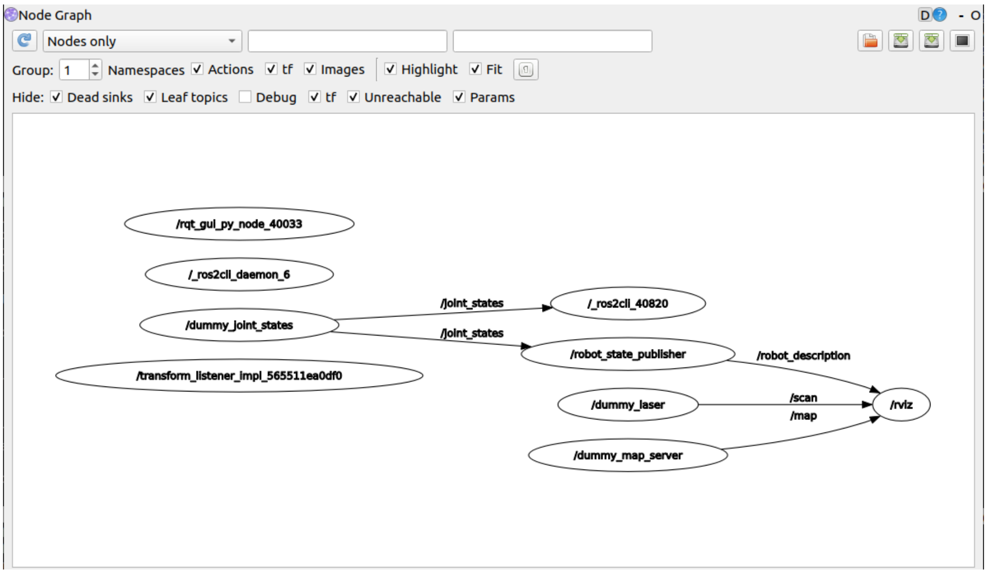
Description:

Display Name



Displays

Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	☑
RobotModel	☑
✓ Status: Ok	
Visual Enabled	☑
Collision Enabled	□
Update Interval	0
Alpha	0,5
Description Source	Topic
Description Topic	/robot_description
TF Prefix	
Links	
TF	☑
✓ Status: Ok	
Show Names	☑
Show Axes	☑
Show Arrows	☑
Marker Scale	1
Update Interval	0
Frame Timeout	15
Frames	
All Enabled	□
single_rrbot_ca...	□
single_rrbot_ho...	☑
single_rrbot_link1	☑
single_rrbot_link2	☑
single_rrbot_link3	☑
world	☑
Tree	
Map	☑
LaserScan	☑



Node Graph

Nodes only

Group: 1 ☑ Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☑ Dead sinks ☑ Leaf topics □ Debug ☑ tf ☑ Unreachable ☑ Params

> Quote
>
> ## What’s happening?
>
> If you have a closer look at the launch file, we start a couple of nodes at the same time.
>
> - `dummy_map_server`
> - `dummy_laser`
> - `dummy_joint_states`
> - `robot_state_publisher`
>
> ⓘ **dummy_robot_bringup.launch.py** ›
>
> ```
> import os
>
> from launch import LaunchDescription
> from launch_ros.actions import Node
> ```

```
from launch_ros.substitutions import FindPackageShare


def generate_launch_description():
    pkg_share = FindPackageShare('dummy_robot_bringup').find('dummy_robot_bringup')
    urdf_file = os.path.join(pkg_share, 'launch', 'single_rrbot.urdf')
    print(urdf_file)
    with open(urdf_file, 'r') as infp:
        robot_desc = infp.read()
    rsp_params = {'robot_description': robot_desc}

    return LaunchDescription([
        Node(package='dummy_map_server', executable='dummy_map_server', output='screen'),
        Node(package='robot_state_publisher', executable='robot_state_publisher',
             output='screen', parameters=[rsp_params]),
        Node(package='dummy_sensors', executable='dummy_joint_states', output='screen'),
        Node(package='dummy_sensors', executable='dummy_laser', output='screen')
    ])
```

The first two packages are relatively simple. The `dummy_map_server` constantly publishes an empty map with a periodic update.

The `dummy_laser` does basically the same; publishing dummy fake laser scans.

The `dummy_joint_states` node is publishing fake joint state data. As we are publishing a simple RRbot with only two joints, this node publishes joint states values for these two joints.

The `robot_state_publisher` is doing the actual interesting work. It parses the given URDF file, extracts the robot model and listens to the incoming joint states. With this information, it publishes TF values for our robot which we visualize in RViz.

3. Read position and orientation data of one of the robot joints

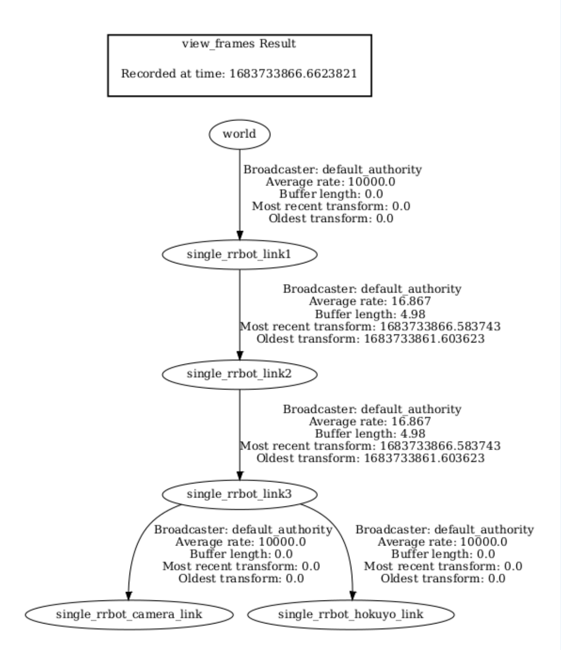
1. Install `view_frames.py`

```
sudo apt-get install ros-foxy-turtle-tf2-py ros-foxy-tf2-tools ros-foxy-tf-transformations
```

2. Run `view_frames.py`

```
ros2 run tf2_tools view_frames.py
```

ⓘ **Output** ›


view_frames Result
Recorded at time: 1683733866.6623821
world
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
single_rrbot_link1
Broadcaster: default_authority
Average rate: 16.867
Buffer length: 4.98
Most recent transform: 1683733866.583743
Oldest transform: 1683733861.603623
single_rrbot_link2
Broadcaster: default_authority
Average rate: 16.867
Buffer length: 4.98
Most recent transform: 1683733866.583743
Oldest transform: 1683733861.603623
single_rrbot_link3
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
single_rrbot_camera_link
single_rrbot_hokuyo_link

3. Use `tf2_echo` to read the position of a frame:

Example

ros2 run tf2_ros tf2_echo [reference_frame] [target_frame]

```
ros2 run tf2_ros tf2_echo world single_rrbot_link3
```

ⓘ **Output** ›

```
ros2 run tf2_ros tf2_echo world single_rrbot_link3
At time 1683756760.890708201
- Translation: [-0.026, 0.200, 2.850]
- Rotation: in Quaternion [0.000, -0.492, 0.000, 0.871]
```

4. Create a node that subscribe the robot joint state and publishes one of the joints state

0. Review the tutorials:

- Using `colcon` to build packages
- Creating a workspace
- Creating a package
- Writing a simple publisher and subscriber (Python)
- Writing a simple service and client (Python)

1. create python packages

```
ros2 pkg create --build-type ament_python dummy_robot_test
```

2. Create the python code in `dummy_robot_test/dummy_robot_test/one_joint_state_publisher.py`

one_joint_state_publisher.py

```
import rclpy
from rclpy.node import Node

from sensor_msgs.msg import JointState

class OneJointState(Node):

    def __init__(self):
        super().__init__('OneJointState')
        # create subscriber
        self.subscription = self.create_subscription(
            JointState,
            '/joint_states',
            self.listener_callback,
            10)

        # create publisher
        self.publisher_ = self.create_publisher(
            JointState,
            'one_joint_state',
            10)

        self.msg = JointState()
        timer_period = 1  # seconds
        self.timer = self.create_timer(timer_period,
self.timer_callback)
        self.i = 0
        self.subscription  # prevent unused variable warning

    def listener_callback(self, msg):
        self.msg.header = msg.header
        self.msg.name = [msg.name[0]]
        self.msg.position = [msg.position[0]]
        # self.msg.velocity = [msg.velocity[0]]
        # self.msg.effort = [msg.effort[0]]
        self.publisher_.publish(self.msg)
        self.i += 1
        pass

    def timer_callback(self):
        self.get_logger().info(f"{self.i}: {self.msg.position[0]}")
        pass

def main(args=None):
    rclpy.init(args=args)

    one_joint_state = OneJointState()
    rclpy.spin(one_joint_state)

    one_joint_state.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

3. Modify and add `'one_joint_state_publisher = dummy_robot_test.one_joint_state_publisher:main'` to `setup.py`
4. Modify and add dependencies to `package.xml` :

```
<exec_depend>launch</exec_depend>
<exec_depend>launch_ros</exec_depend>
<exec_depend>std_msgs</exec_depend>
<exec_depend>geometry_msgs</exec_depend>
<exec_depend>python3-numpy</exec_depend>
<exec_depend>rclpy</exec_depend>
<exec_depend>dummy_robot_bringup</exec_depend>
<exec_depend>dummy_map_server</exec_depend>
<exec_depend>dummy_sensors</exec_depend>
```

6. Build and source:

```
ros2 pkg create --build-type ament_python dummy_robot_test
colcon build --symlink-install
source ~/ros2_ws/install/setup.bash
```

7. Launch `dummy_robot_bringup`:

```
ros2 launch dummy_robot_bringup dummy_robot_bringup.launch.py
```

8. Run `one_joint_state_publisher`

```
$ ros2 run dummy_robot_test one_joint_state_publisher
[INFO] [1683815233.101406142] [OneJointState]: 50: 0.4747952280852878
```

5. Create node send movement to the robot

Assuming the same package of the last example.

1. Create the python code `/src/dummy_robot_test/dummy_robot_test/dummy_joint_states.py`

ⓘ **dummy_joint_states.py** ›

```
import rclpy
from rclpy.node import Node

from sensor_msgs.msg import JointState
import numpy as np

class DummyJointStates(Node):

    def __init__(self):
        super().__init__('DummyJointStates')
        # create joint_state  publisher
        self.publisher_ = self.create_publisher(
            JointState,
            'joint_states',
            10)

        self.msg = JointState()
        self.msg.name.append("single_rrbot_joint1")
        self.msg.name.append("single_rrbot_joint2")
        self.msg.position = [.0,.0]
        timer_period = 0.05  # seconds
        self.timer = self.create_timer(timer_period,
self.timer_callback)
        self.i = 0

    def timer_callback(self):
        self.i += 1
        self.msg.position[0] = np.sin(self.i/100)*np.pi/2
        self.msg.position[1] = np.cos(self.i/10)*np.pi
        self.msg.header.stamp = self.get_clock().now().to_msg()
        self.publisher_.publish(self.msg)
        self.get_logger().info(f"{self.i}:
{self.msg.header.stamp}")
        self.get_logger().info(f"{self.i}: {self.msg.name}")
        self.get_logger().info(f"{self.i}: {self.msg.position}")
        pass

def main(args=None):
    rclpy.init(args=args)

    dummy_joint_states = DummyJointStates()
    rclpy.spin(dummy_joint_states)

    dummy_joint_states.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

2. Modify `setup.py` to add the executable:

ⓘ **setup.py** ›

```
import os
from glob import glob
from setuptools import setup

package_name = 'dummy_robot_test'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        # Include all launch files.
        (os.path.join('share', package_name, 'launch'),
glob(os.path.join('launch', '*launch.[pxy][yma]*'))),
        # # Include all conf files.
        (os.path.join('share', package_name, 'conf'),
glob(os.path.join('conf', '*'))),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='gabrielbermudez',
    maintainer_email='gabrielbermudez@gmail.com',
    description='Package for test',
    license='Apache License 2.0',
    tests_require=['pytest'],
    entry_points={
            'console_scripts': [
                    'one_joint_state_publisher =
dummy_robot_test.one_joint_state_publisher:main',
                    'dummy_joint_states =
dummy_robot_test.dummy_joint_states:main',
            ],
    },
)
```

3. Create Rviz2 conf file in `dummy_robot_test/conf/dummy_robot.rviz`
4. Create a launch file in `dummy_robot_test/launch/dummy_robot_test_launch.py`

ⓘ **dummy_robot_test_launch.py** ›

```
import os

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

def generate_launch_description():
    pkg_share =
FindPackageShare('dummy_robot_bringup').find('dummy_robot_bringup')
    urdf_file = os.path.join(pkg_share, 'launch',
'single_rrbot.urdf')
    with open(urdf_file, 'r') as infp:
        robot_desc = infp.read()
    rsp_params = {'robot_description': robot_desc}
    rviz_config = os.path.join(
        get_package_share_directory('dummy_robot_test'),
        'conf',
        'dummy_robot.rviz'
        )
    print(rviz_config)
    return LaunchDescription([
        Node(package='dummy_map_server',
executable='dummy_map_server', output='screen'),
        Node(package='robot_state_publisher',
executable='robot_state_publisher',
             output='screen', parameters=[rsp_params]),
        Node(package='dummy_sensors', executable='dummy_laser',
output='screen'),
        Node(
            package='dummy_robot_test',
            executable='dummy_joint_states',
            output='screen'
        ),
        Node(
            package='rviz2',
            executable='rviz2',
            name='rviz2',
            arguments=['-d', rviz_config],
            output='screen'
        )
    ])
```

5. Modify and add dependencies to `package.xml` :

package.xml

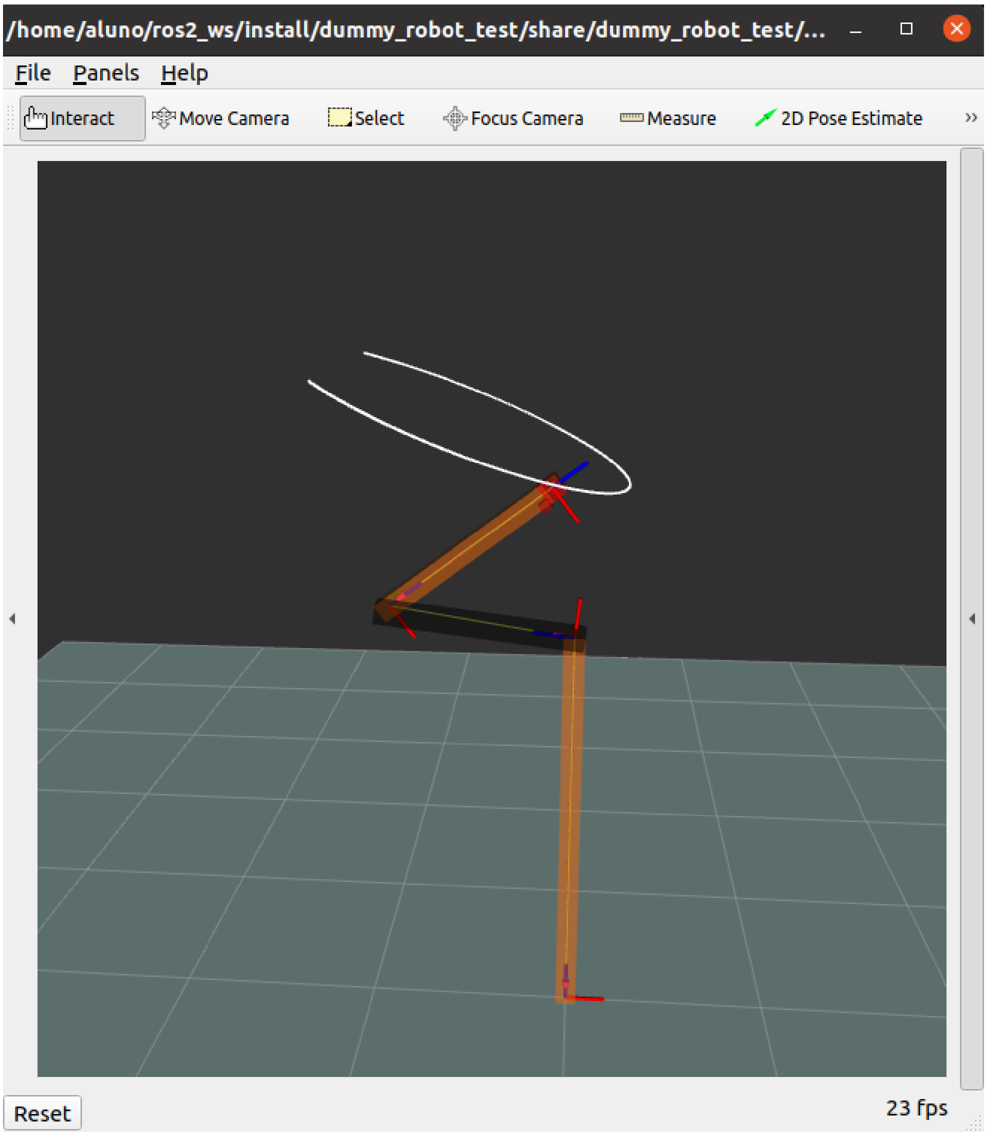
```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>dummy_robot_test</name>
  <version>0.0.0</version>
  <description>Package for test</description>
  <maintainer email="gabrielbermudez@usp.br">Gabriel
Bermudez</maintainer>
  <license>Apache License 2.0</license>
  <exec_depend>rclpy</exec_depend>
  <exec_depend>launch</exec_depend>
  <exec_depend>launch_ros</exec_depend>
  <exec_depend>std_msgs</exec_depend>
  <exec_depend>geometry_msgs</exec_depend>
  <exec_depend>python3-numpy</exec_depend>
  <exec_depend>rclpy</exec_depend>
  <exec_depend>dummy_robot_bringup</exec_depend>
  <exec_depend>dummy_map_server</exec_depend>
  <exec_depend>dummy_sensors</exec_depend>

  <test_depend>ament_copyright</test_depend>
  <test_depend>ament_flake8</test_depend>
  <test_depend>ament_pep257</test_depend>
  <test_depend>python3-pytest</test_depend>

  <export>s
    <build_type>ament_python</build_type>
  </export>
</package>
```

6. Build and source:

```
ros2 pkg create --build-type ament_python dummy_robot_test
colcon build --symlink-install
source ~/ros2_ws/install/setup.bash
```

7. Launch `dummy_robot_test_launch.py`:

```
ros2 launch dummy_robot_test dummy_robot_test_launch.py
```


/home/aluno/ros2_ws/install/dummy_robot_test/share/dummy_robot_test/...
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
Reset
23 fps