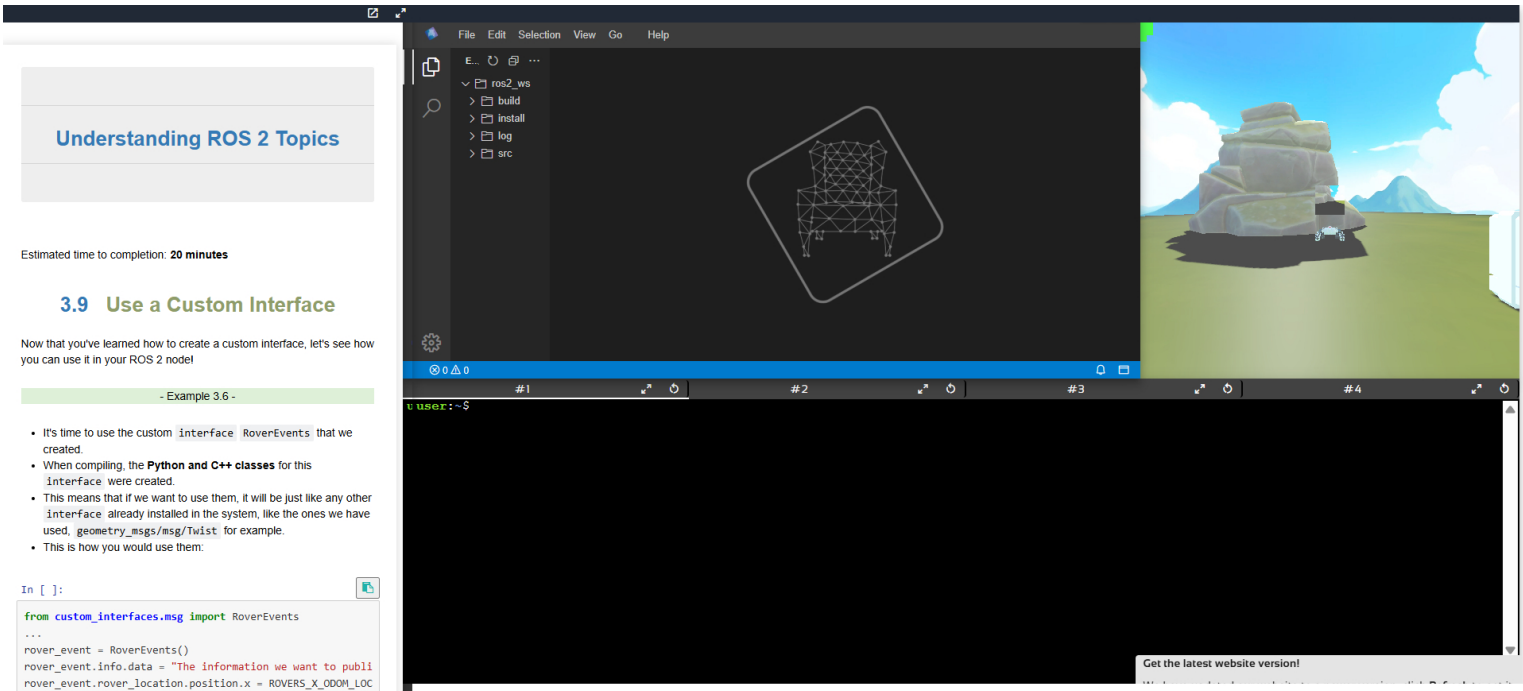


Understanding ROS 2 Topics

Estimated time to completion: **20 minutes**

3.9 Use a Custom Interface

Now that you've learned how to create a custom interface, let's see how you can use it in your ROS 2 node!

- Example 3.6 -

- It's time to use the custom `interface` `RoverEvents` that we created.
- When compiling, the **Python and C++ classes** for this `interface` were created.
- This means that if we want to use them, it will be just like any other `interface` already installed in the system, like the ones we have used, `geometry_msgs/msg/Twist` for example.
- This is how you would use them:

In [ ]:

```
from custom_interfaces.msg import RoverEvents
...
rover_event = RoverEvents()
rover_event.info.data = "The information we want to publi
rover_event.rover_location.position.x = ROVERS_X_ODOM_LOC
rover_event.rover_location.position.y = ROVERS_Y_ODOM_LOC
```

- Let's create a modification of the `plant_detector_node.py` to publish whatever is detected and retrieve the odometry location at that point.
- We will publish this data in a new `topic` named `mars_rover_events`.
- We will need to create a `subscriber` **for the odometry** `topic` `/odom`:

In [ ]:

```
# Subscribe to the odometry topic
self.odom_subscription = self.create_subscription(
    Odometry,
    '/odom',
    self.odom_callback,
    10)
self.odom_subscription  # prevent unused variable warning

...

def odom_callback(self, msg):
    # Store the current odometry data
    self.current_odom = msg
```

- Here we save the latest **odometry data** in the variable `self.current_odom`.
- This is because this way we can retrieve that data in another `callback`.

In [ ]:

```
def image_callback(self, msg):
    # Convert ROS Image message to OpenCV image
    cv_image = self.bridge.imgmsg_to_cv2(msg, "bgr8")

    # Use the PlantDetector to make a prediction
    prediction = self.plant_detector.predict(cv2.cvtColor

    # Create a RoverEvents message
    rover_event = RoverEvents()

    # Determine the result message based on the predictio
    if prediction > 0.5:
        rover_event.info.data = f"Plant detected with con
        self.get_logger().warning(rover_event.info.data)
        self.get_logger().warning("Publishing mars rover
        rover_event.info.data = f"Plant detected with con
        # If the odometry data is available, include the
        if self.current_odom:
            rover_event.rover_location = self.current_odo

        # Publish the RoverEvents message
        self.publisher_.publish(rover_event)
    else:
        rover_event.info.data = f"No plant detected. Conf
        self.get_logger().info(rover_event.info.data)
```

- Now we **ONLY** publish into the `mars_rover_events` when a plant is detected.
- We retrieve the odometry data and publish it.

- At the end you should get something similar to this:

plant_detector_node.py

In [ ]:

```
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Image
from nav_msgs.msg import Odometry  # Import Odometry mess
from custom_interfaces.msg import RoverEvents  # Import R
```

```
from cv_bridge import CvBridge
import cv2
from mars_rover_tasks.plant_detector import PlantDetector

class PlantDetectorNode(Node):
    def __init__(self):
        super().__init__('plant_detector_node')

        # Initialize the CvBridge
        self.bridge = CvBridge()

        # Initialize the PlantDetector
        path_to_model = "/home/user/ros2_ws/src/basic_ros
        self.plant_detector = PlantDetector(model_path=pa

        # Subscribe to the image topic
        self.subscription = self.create_subscription(
            Image,
            '/leo/camera/image_raw',
            self.image_callback,
            10)
        self.subscription  # prevent unused variable warn

        # Subscribe to the odometry topic
        self.odom_subscription = self.create_subscription
            Odometry,
            '/odom',
            self.odom_callback,
            10)
        self.odom_subscription  # prevent unused variable

        # Initialize the Publisher for rover events
        self.publisher_ = self.create_publisher(RoverEven

        # Variable to store the latest odometry message
        self.current_odom = None

    def odom_callback(self, msg):
        # Store the current odometry data
        self.current_odom = msg

    def image_callback(self, msg):
        # Convert ROS Image message to OpenCV image
        cv_image = self.bridge.imgmsg_to_cv2(msg, "bgr8")

        # Use the PlantDetector to make a prediction
        prediction = self.plant_detector.predict(cv2.cvtC

        # Create a RoverEvents message
        rover_event = RoverEvents()

        # Determine the result message based on the predi
        if prediction > 0.5:
            rover_event.info.data = f"Plant detected with
            self.get_logger().warning(rover_event.info.da
            self.get_logger().warning("Publishing mars ro
            rover_event.info.data = f"Plant detected with
            # If the odometry data is available, include
            if self.current_odom:
                rover_event.rover_location = self.current

            # Publish the RoverEvents message
            self.publisher_.publish(rover_event)
        else:
            rover_event.info.data = f"No plant detected.
            self.get_logger().info(rover_event.info.data)



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

    plant_detector_node = PlantDetectorNode()

    rclpy.spin(plant_detector_node)

    # Destroy the node explicitly (optional)
    plant_detector_node.destroy_node()
    rclpy.shutdown()

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

- Lets compile and run this node:

Build and Run

Execute in Terminal #1

In [ ]:

```
cd ~/ros2_ws
colcon build --packages-select mars_rover_tasks
source install/setup.bash
```

- And run it:

Execute in Terminal #1

- Start the `plant_detector_node`:

In [ ]:

```
cd ~/ros2_ws
source install/setup.bash
ros2 run mars_rover_tasks image_plant_detector_executable
```

Execute in Terminal #2

- Move the Mars rover around to test whether the camera detects the plant when it sees it.

In [ ]:

```
cd ~/ros2_ws
```

```
source install/setup.bash
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

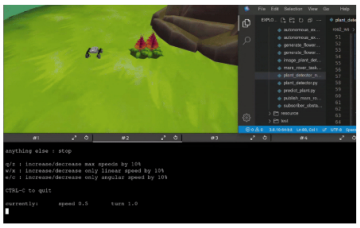
Execute in Terminal #3

- `Echo` the `topic` `/mars_rover_events` to see the publication of these detections.
- **BE CAREFUL** to `source install/setup.bash`, because otherwise in that terminal ROS2 won't understand what the new custom type `RoverEvents`.

In [ ]:

```
cd ~/ros2_ws
source install/setup.bash
ros2 topic echo /mars_rover_events
```

- You should see something like so, try moving to different points to see how the odometry changes:



- End of Example 3.5 -

24/09/2024