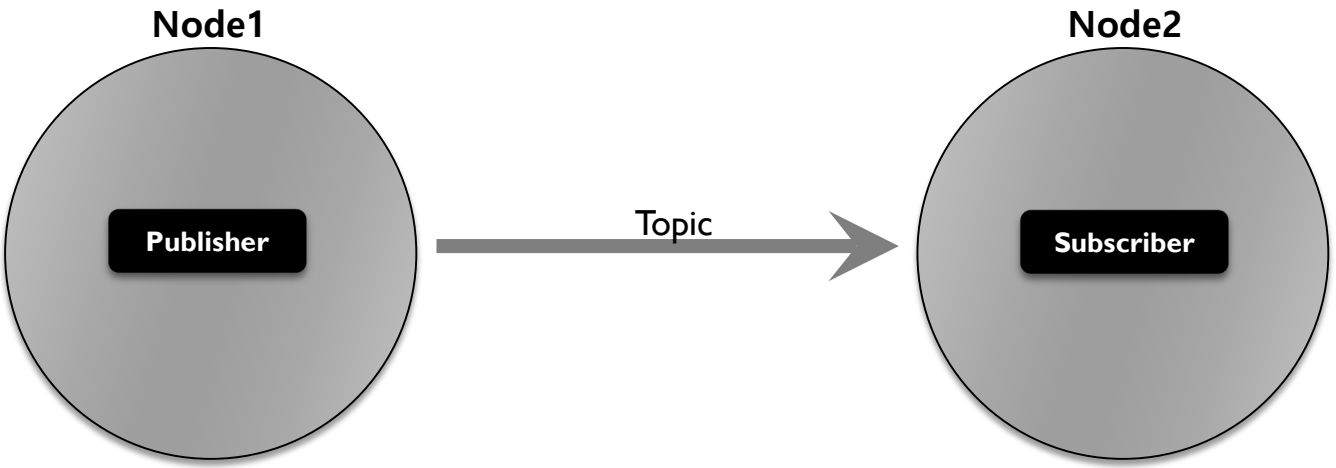
ME454
Dynamics System Programming

TA Session 6. Topics/Gazebo Messages/Pub-subscriber

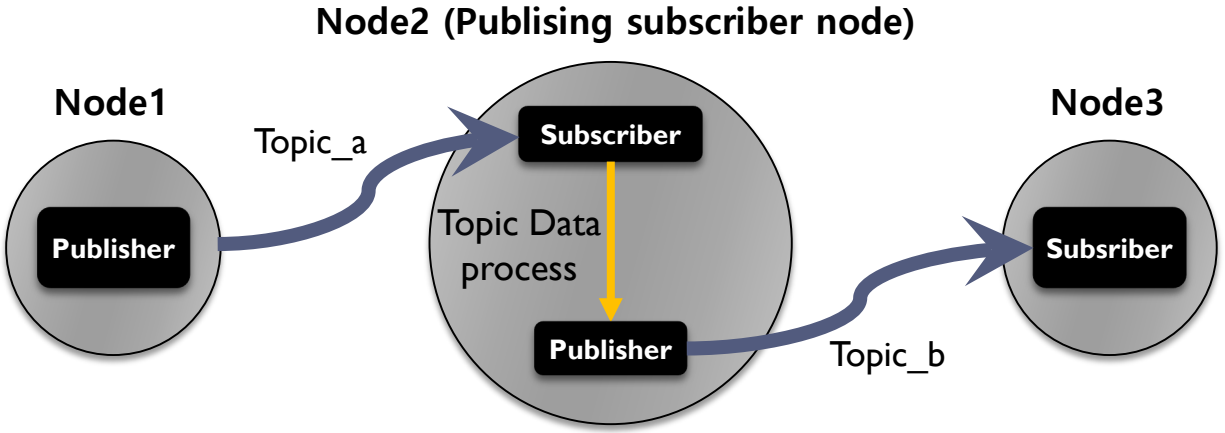
Topics



For continuous data streams and data flow

Publishing subscriber node

- We can make a node both publishes and subscribes
- https://automaticaddison.com/create-a-publisher-and-subscriber-in-c-ros-2-foxy-fitzroy/
  - Docs for foxy, but also available for ros-humble
  - Create a Publishing Subscriber Node
  - By making a node with both create_subscription and create_publisher, we can make publishing subscriber node.



Python Launch file

```
ld = LaunchDescription()
ld.add_action(declare_use_joint_state_publisher_cmd)
ld.add_action(declare_simulator_cmd)
ld.add_action(declare_urdf_model_path_cmd)
ld.add_action(declare_use_robot_state_pub_cmd)
ld.add_action(declare_use_sim_time_cmd)
ld.add_action(declare_use_simulator_cmd)
ld.add_action(declare_world_cmd)
ld.add_action(start_gazebo_server_cmd)
ld.add_action(start_gazebo_client_cmd)

ld.add_action(spawn_coin_cmd)
ld.add_action(start_robot_state_publisher_cmd)

return ld
```

Launch → pack multiple commands in a single file

Python Launch file

```
declare_urdf_model_path_cmd = DeclareLaunchArgument(
name='urdf_model',
default_value=default_urdf_coin_model_path,
description='Absolute path to robot urdf file')
```

```
spawn_coin_cmd = Node(
package='gazebo_ros',
executable='spawn_entity.py',
arguments=['-entity', 'coin',
                '-topic', 'robot_description',
                    '-x', spawn_x_val_coin,
                    '-y', spawn_y_val_coin,
                    '-z', spawn_z_val_coin,
                    '-R', spawn_roll_val_coin],
                   output='screen')
```

Spawn the model in urdf to gazebo
-entity : name of the model in gazebo
-file : urdf file containing the model
x, y, z, R, P, Y: position and orientation of the spawned model

Python Launch file

```
coin_subscriber_cmd = Node(
    package='coin_rolling',
    executable='coin_subscriber',
    output='screen',
    name="coin_subscriber"
)
```

Running a node in the launch file
Package : package name
Executable : name of the executable file on CmakeLists.txt
Name : name of the node

ROS & Gazebo tutorials

- Coin rolling

Github Download

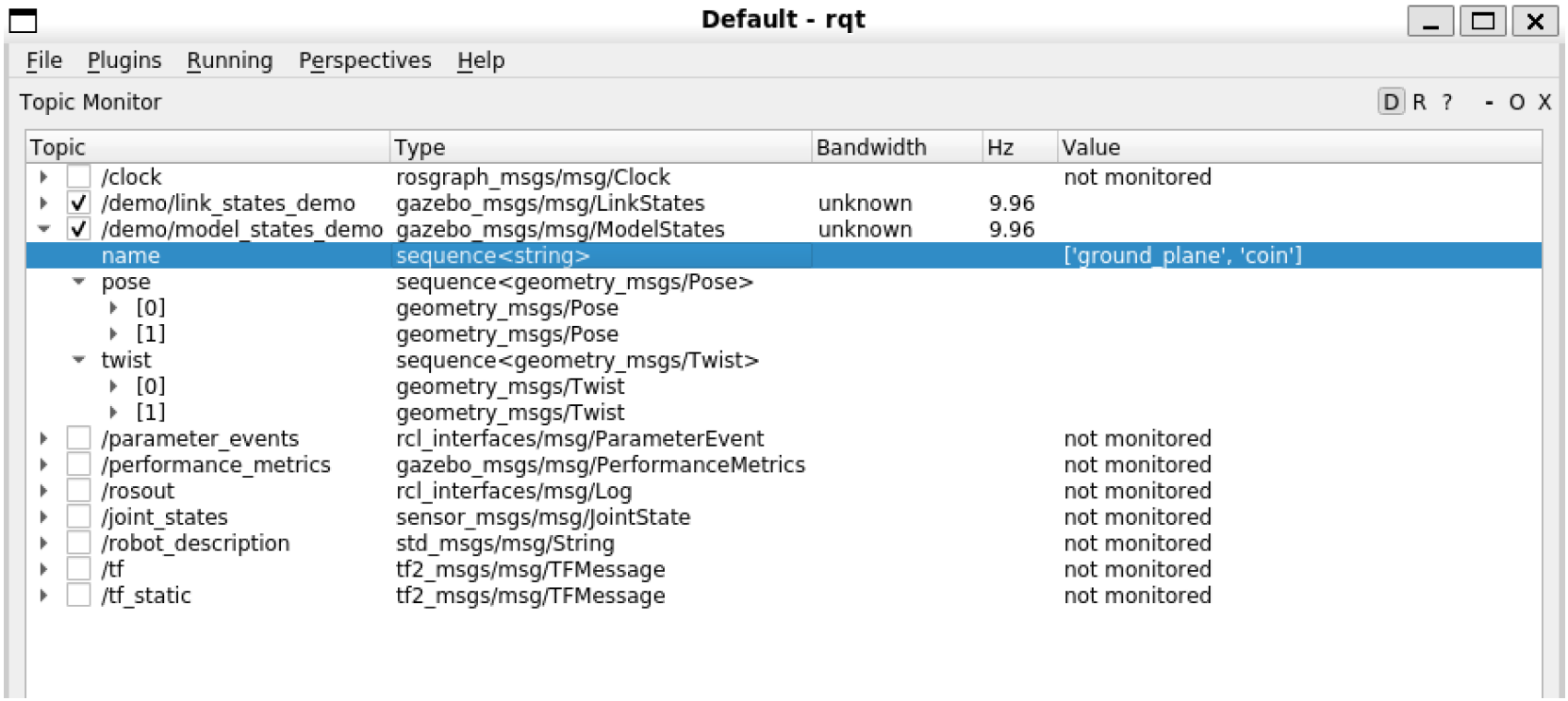
```
cd ~/repos/ME454_2024
git pull

cd ~/ros2_ws/src
cp -r ~/repos/ME454_2024/coin_rolling ~/ros2_ws/src
cp -r ~/repos/ME454_2024/coin_rolling_msg ~/ros2_ws/src
```

Running Gazebo

```
cd ~/ros2_ws
colcon build
. install/local_setup.bash
ros2 launch coin_rolling coin_spawn_launch.py
```

Gazebo_msgs

- Message and service data structures for interacting with Gazebo from ROS



World file

- Coin_rolling.world

```
<?xml version="1.0"?>
<sdf version="1.6">
  <world name="world">

    <plugin name="gazebo_ros_state" filename="libgazebo_ros_state.so">
      <ros>
        <namespace>/demo</namespace>
        <argument>model_states:=model_states_demo</argument>
        <argument>link_states:=link_states_demo</argument>
      </ros>

      <update_rate>10.0</update_rate>
    </plugin>
```

- Gazebo plugins publishes the gazebo_msgs which we can subscribe
- In this plugin, the name of the topic is
  - /demo/model_states_demo
  - /demo/link_states_demo
- Can change the name of the topic

World file

- Ground plane
  - Set physical properties of the ground plane
- Gravity
  - Control the gravity

```
<model name='ground_plane'>
<static>1</static>
<link name='link'>
  <collision name='collision'>
    <geometry>
      <plane>
        <normal>0 0 1</normal>
        <size>100 100</size>
      </plane>
    </geometry>
    <surface>
      <contact>
        <collide_bitmask>65535</collide_bitmask>
        <ode/>
      </contact>
      <friction>
        <ode>
          <mu>100</mu>
          <mu2>50</mu2>
        </ode>
        <torsional>
          <ode/>
        </torsional>
      </friction>
      <bounce/>
    </surface>
    <max_contacts>10</max_contacts>
  </collision>
  <visual name='visual'>
    <cast_shadows>0</cast_shadows>
    <geometry>
      <plane>
        <normal>0 0 1</normal>
        <size>100 100</size>
      </plane>
    </geometry>
    <material>
      <script>
        <name>Gazebo/Grey</name>
      </script>
    </material>
  </visual>
  <self_collide>0</self_collide>
  <enable_wind>0</enable_wind>
  <kinematic>0</kinematic>
</link>
```

URDF gazebo property

- coin.urdf

```
<gazebo reference="base_link">
    <collision name='base_link_collision'>
    <surface>
    <friction>
    <ode>
    <mu>0.9</mu>
    <mu2>0.9</mu2>
    <fdir1>0 0 0</fdir1>
    </ode>
    </friction>
    <contact>
    <ode>
    <kp>1e+13</kp>
    <kd>1</kd>
    <max_vel>10.0</max_vel>
    </ode>
    </contact>
    <bounce>
    <restitution_coefficient>1.0</restitution_coefficient>
    <threshold>0.001</threshold>
    </bounce>
    </surface>
    </collision>
</gazebo>
```

URDF gazebo property

- http://classic.gazebosim.org/tutorials?tut=ros_urdf&cat=connect_ros

`<gazebo>` Elements For Links

List of elements that are individually parsed:

Name	Type	Description
material	value	Material of visual element
gravity	bool	Use gravity
dampingFactor	double	Exponential velocity decay of the link velocity - takes the value and multiplies the previous link velocity by (1-dampingFactor).
maxVel	double	maximum contact correction velocity truncation term.
minDepth	double	minimum allowable depth before contact correction impulse is applied
mu1	double	Friction coefficients μ for the principal contact directions along the contact surface as defined by the Open Dynamics Engine (ODE) (see parameter descriptions in ODE's user guide)
mu2		
fdir1	string	3-tuple specifying direction of mu1 in the collision local reference frame.
kp	double	Contact stiffness k_p and damping k_d for rigid body contacts as defined by ODE (ODE uses erp and cfm but there is a mapping between erp/cfm and stiffness/damping)
kd		
selfCollide	bool	If true, the link can collide with other links in the model.
maxContacts	int	Maximum number of contacts allowed between two entities. This value overrides the max_contacts element defined in physics.
laserRetro	double	intensity value returned by laser sensor.

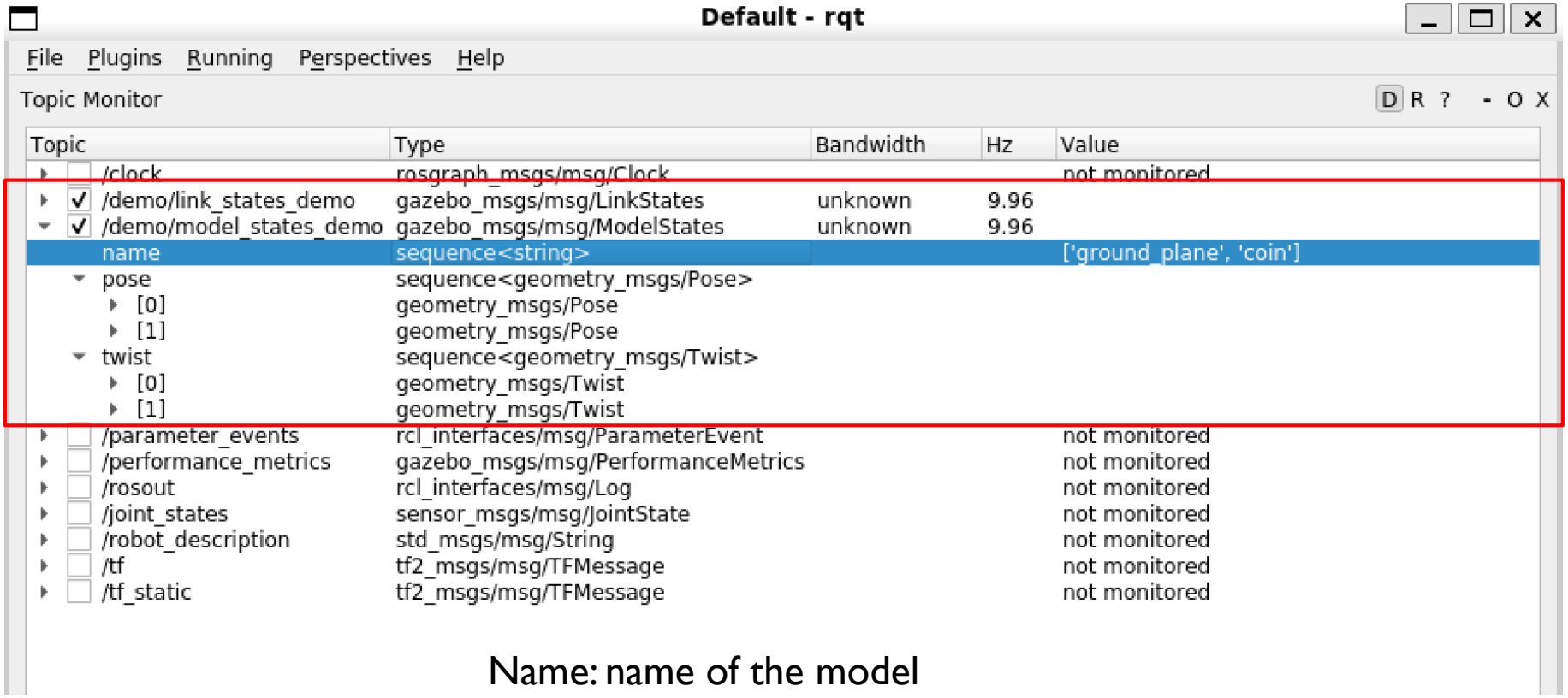
Similar to `<gazebo>` elements for `<robot>`, any arbitrary blobs that are not parsed according to the table above are inserted into the the corresponding `<link>` element in the SDF. This is particularly useful for plugins, as discussed in the ROS Motor and Sensor Plugins tutorial.

Give velocity to the coin

```
ros2 service call /demo/set_entity_state
'gazebo_msgs/SetEntityState' '{state: {name: "coin", pose:
{position: {x: 2.6472, y: 0.0, z: 0.5}, orientation: {x: 0.0, y:
0.7071068, z: 0.0, w: 0.7071068}}, twist: {linear: {x: 0.0, y: 2.0,
z: 0.0}, angular: {x: -1.0, y: 0.0, z: 0.0}}}}'
```

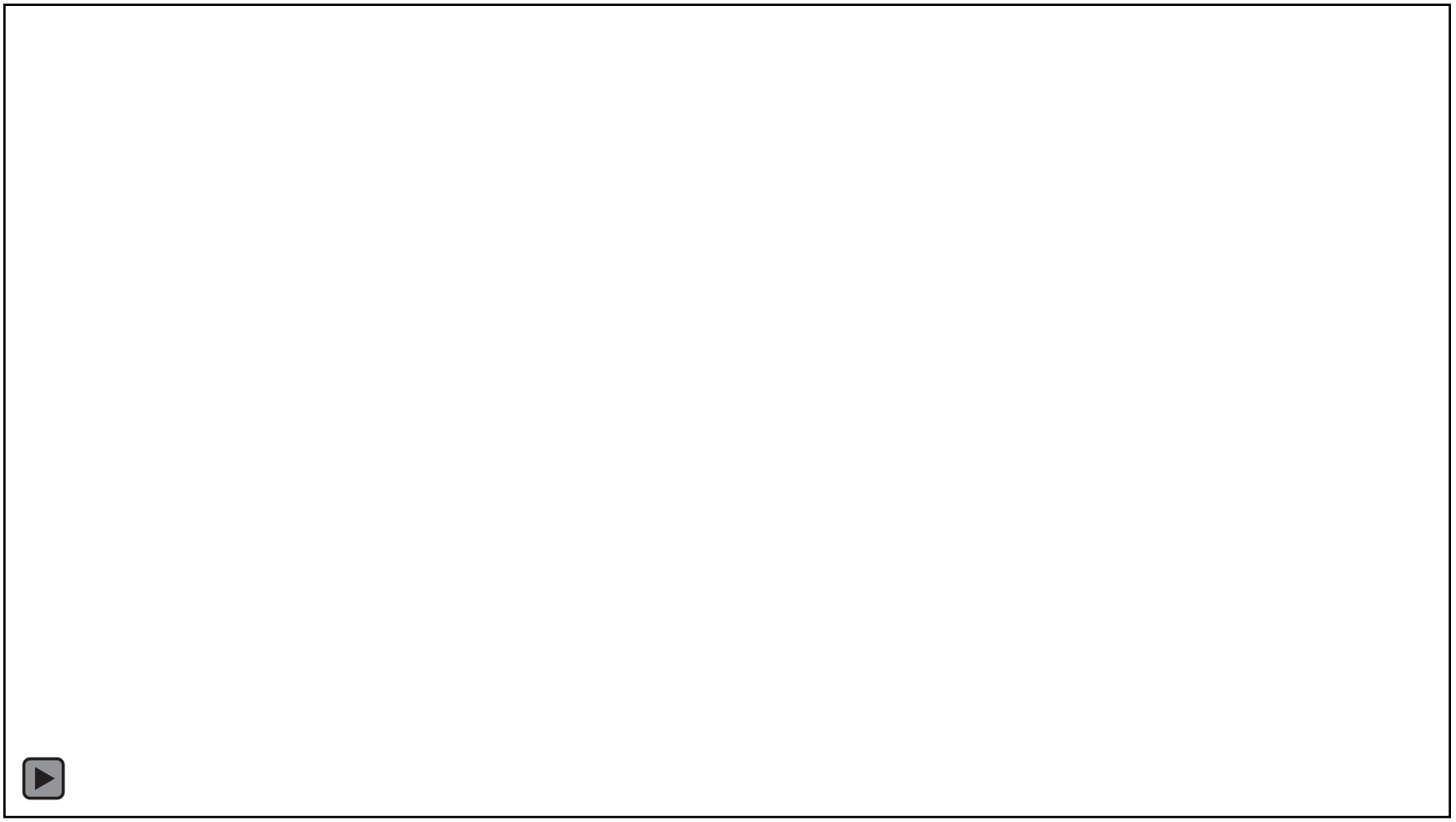
Gazebo_msgs

- Check the msgs property using rqt

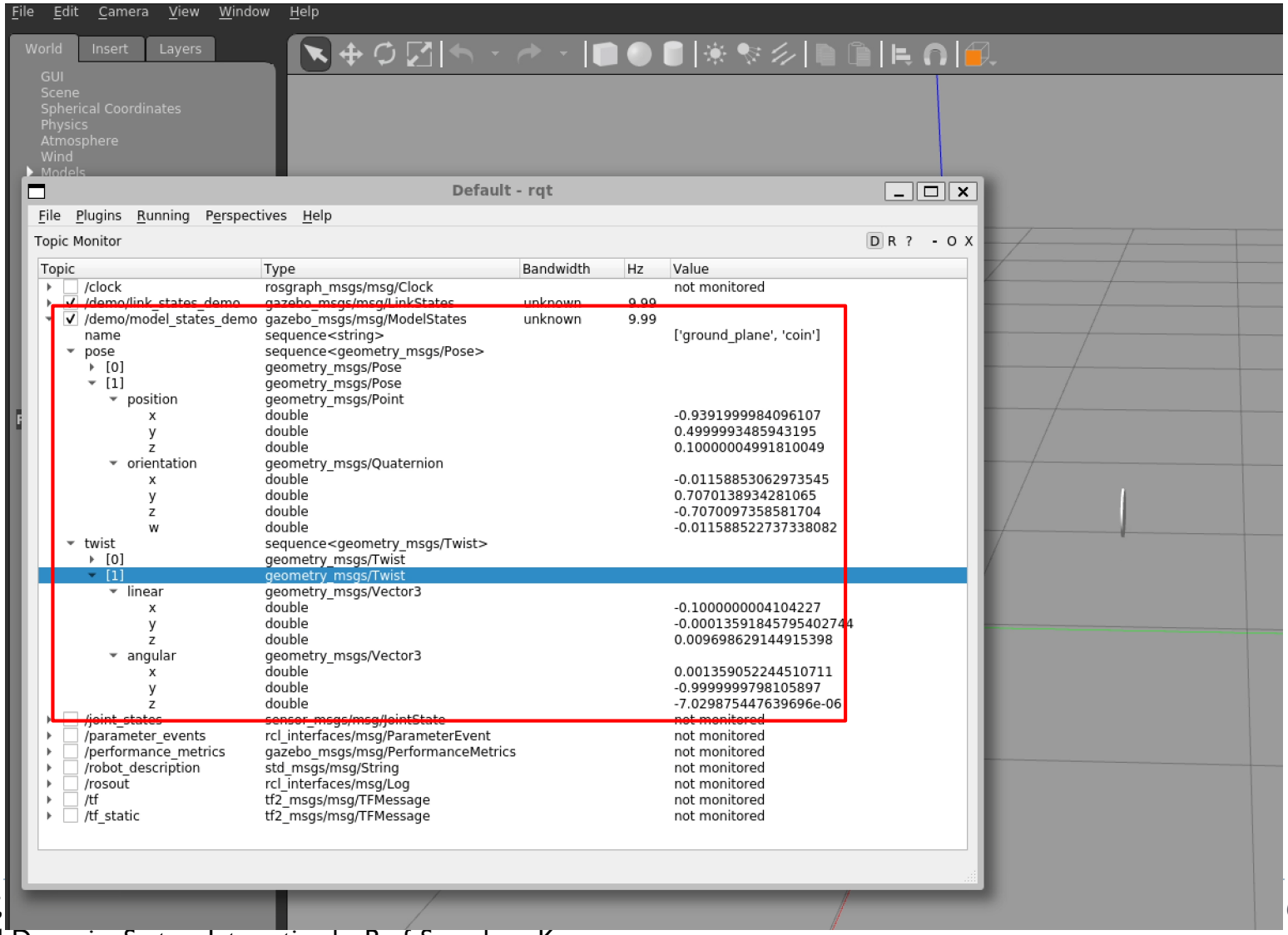


Name: name of the model
Pose : position and orientation
Twist : linear and angular velocity

Gazebo_msgs on rqt



Gazebo_msgs on rqt



Gazebo_msgs subscriber

- coin_subscriber.cpp
- Need to include header file
- Header file naming
  - All lowercase
- gazebo_msgs::msg::ModelStates → gazebo_msgs/msg/model_states.hpp
- geometry_msgs::msg::Vector3 → geometry_msgs/msg/vector3.hpp
- std_msgs::msg::Float32 → std_msgs/msg/float32.hpp

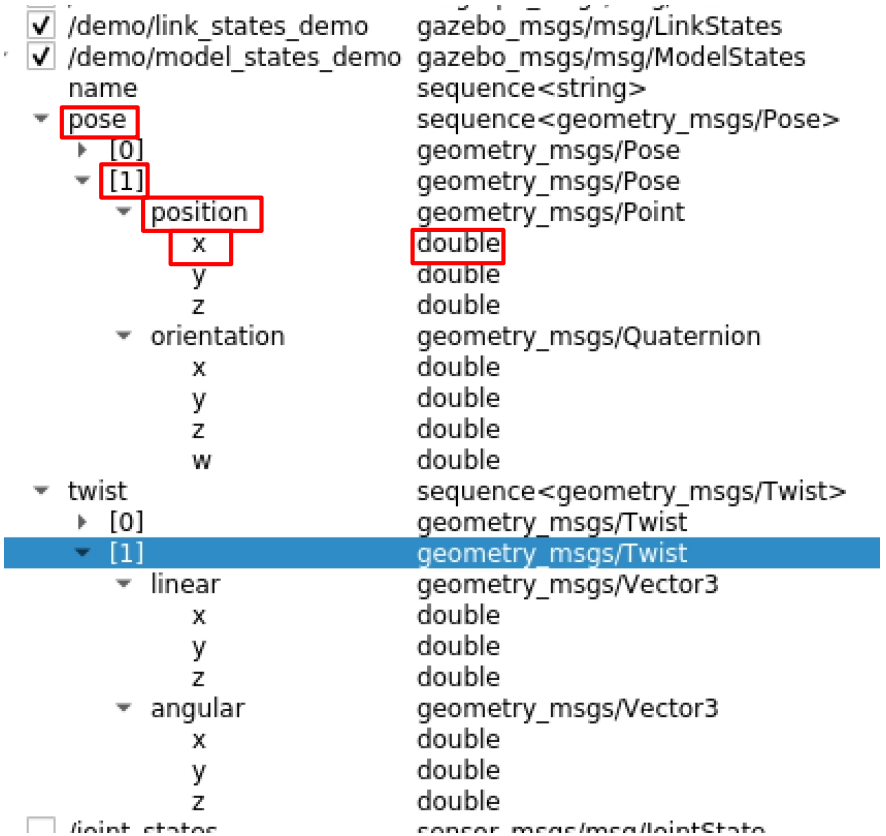
```
#include <memory>
#include "rclcpp/rclcpp.hpp"
#include "gazebo_msgs/msg/model_states.hpp"
#include "std_msgs/msg/string.hpp"
#include "std_msgs/msg/int32.hpp"
```

Gazebo_msgs subscriber

- Check the structure

- {msg}->{pose}[1].position.x
  - Type double

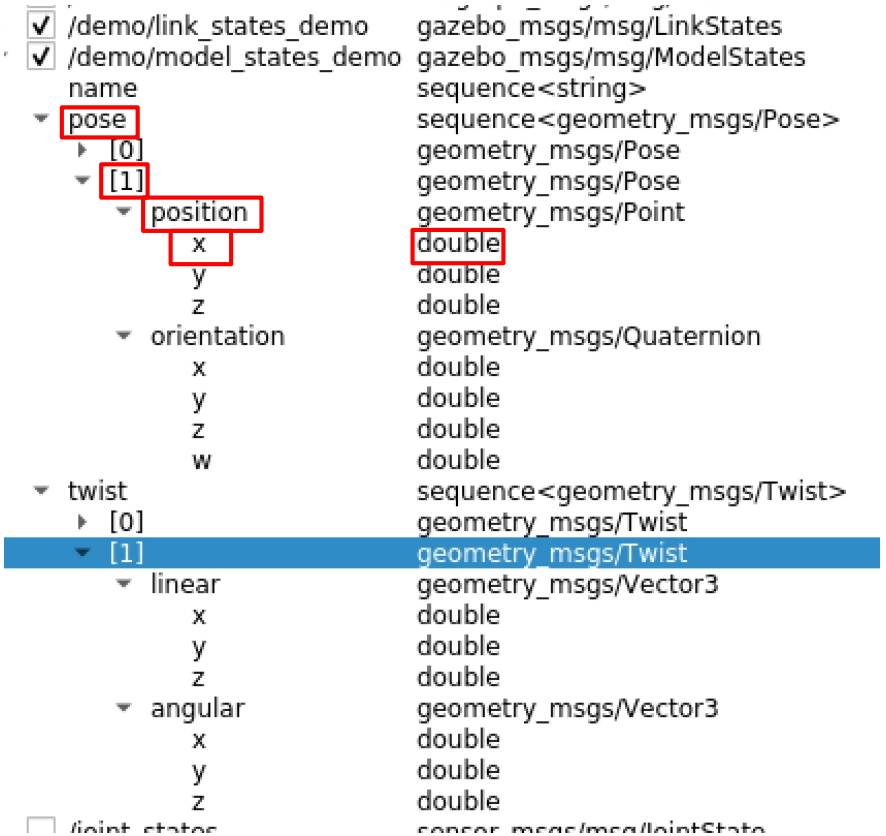
- Can access the date in the message by editing the sections




```
int current_model_idx = 1;
double wy = msg->twist[current_model_idx].angular.y;
double vx = msg->twist[current_model_idx].linear.x;
```

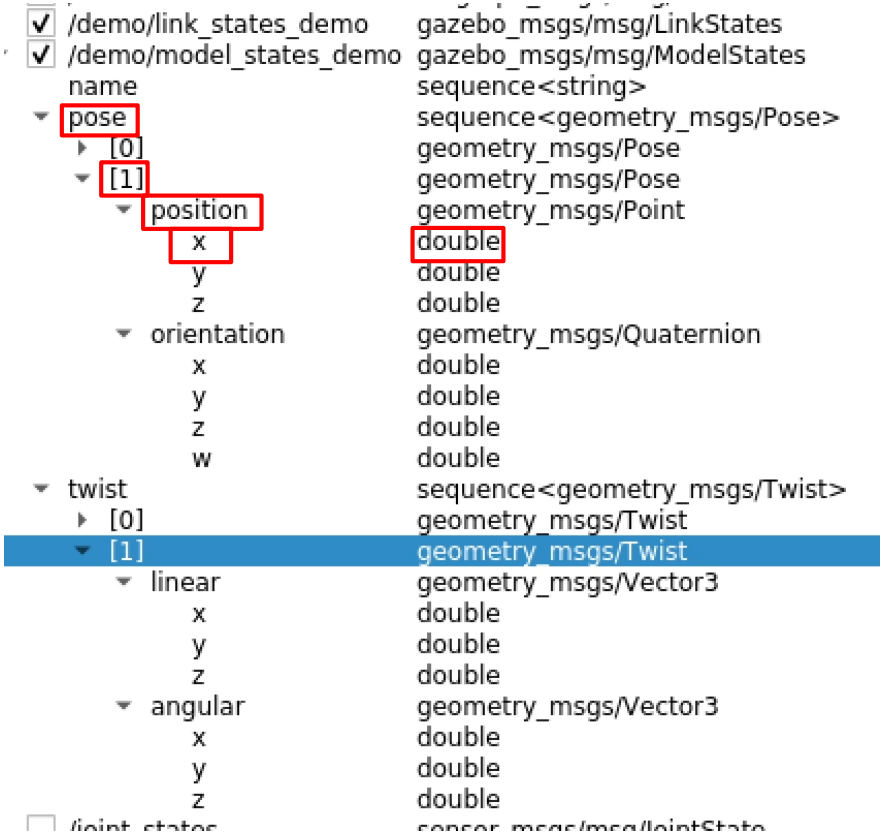
Gazebo_msgs subscriber

- geometry_msgs/Point
  - Composed of x, y, z
- geometry_msgs/Quaternion
  - Composed of x, y, z, w(real part)
- geometry_msgs/Pose
  - Composed of position, orientation
  - Position is geometry_msgs/Point element
  - orientation is geometry_msgs/Quaternion element



Gazebo_msgs subscriber

- geometry_msgs/Vector3
  - Composed of x, y, z
- geometry_msgs/Twist
  - Composed of linear, angular
  - Linear and angular is geometry_msgs/Vector3 element



Publishing subscriber node

- Define the subscription_ and publisher_, and make a constructor of both.
- When there is a subscription, topic_callback function is called

```
public:
    MinimalPubSub()
    : Node("coin_pubsub")
    {
        subscription_ = this->create_subscription<gazebo_msgs::msg::ModelStates> (
                "demo/model_states_demo",
                10,
                std::bind(&MinimalPubSub::topic_callback, this, _1)
                );

        publisher_ = this->create_publisher<std_msgs::msg::Float32>("coin/inclined_coin",10);
    }
private:
rclcpp::Subscription<gazebo_msgs::msg::ModelStates>::SharedPtr subscription_;
rclcpp::Publisher<std_msgs::msg::Float32>::SharedPtr publisher_;
```

Publishing subscriber node

- Topic_callback function
  - Get the angular velocity around y-axis on gazebo_msgs
  - Make a message with the angular velocity
  - Publish the message

```
void topic_callback(const gazebo_msgs::msg::ModelStates::SharedPtr msg) const
{
    double wy = msg->twist[1].angular.y;
    auto message = std_msgs::msg::Float32();
    message.data = wy;
    publisher_->publish(message);
}
```

Tasks

- Edit the inclined_coin_pubsub.cpp file

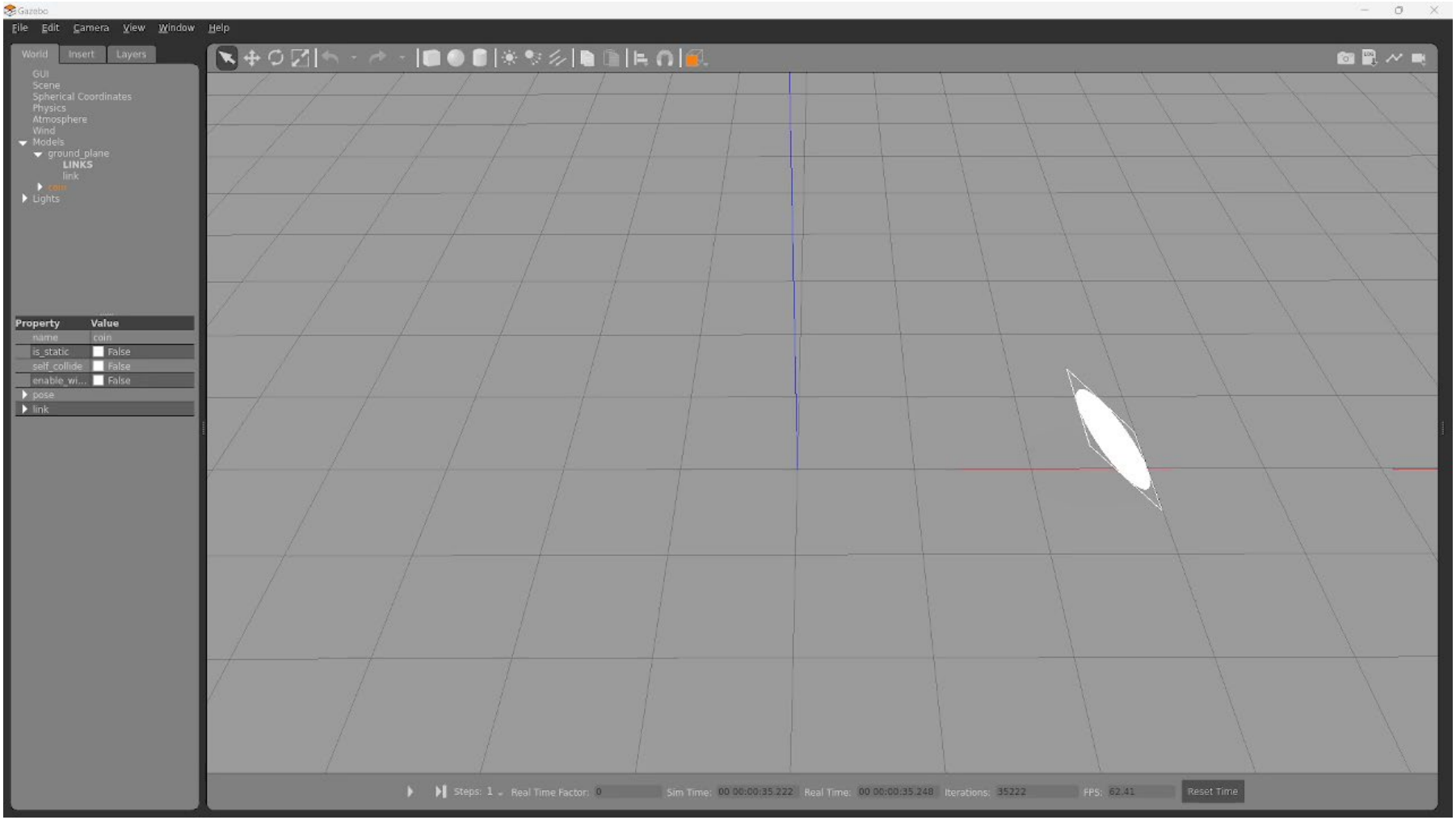
- Calculate the angular momentum of the rolling coin.
  - Angular velocity is given on the world coordinate, and the mass moment of inertia is given on the body coordinate.
  - Convert the quaternion which expresses the orientation of the coin to rotation matrix
  - Calculate the angular momentum in the body(coin) coordinate.
  - Publish the calculated results

Tasks

- Edit the launch file (coin_spawn_launch.py) to run the "inclined_coin_pubsub" node.
- Launch the coin_spawn_launch.py
- Set the velocity of the coin by the command below

ros2 service call /demo/set_entity_state 'gazebo_msgs/SetEntityState' '{state: {name: "coin", pose: {position: {x: 2.6472, y: 0.0, z: 0.3536}, orientation: {x: 0.0, y: -0.9238795, z: 0.0, w: 0.3826834}}, twist: {linear: {x: 0.0, y: 4.1603, z: 0.0}, angular: {x: -5.8835, y: 0.0, z: -4.3115}}}}'

Rolling inclined coin



Launched world

ME454
Dynamics System Programming

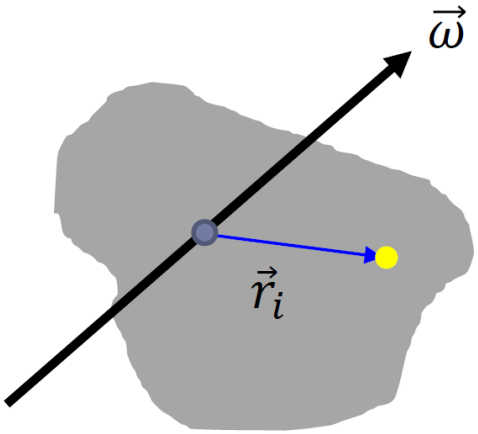
TA Session 6 Suppliments

Quaternion to rotation matrix conversion

$$q = w + xi + yj + zk, \ |q| = 1$$

$$\Rightarrow \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2wz & 2xz + 2wy \\ 2xy + w2z & 1 - 2x^2 - 2z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & 1 - 2x^2 - 2y^2 \end{bmatrix}$$

Moment of inertia



$$\vec{a} \times \left(\vec{b} \times \vec{c}\right) = (\vec{a} \cdot \vec{c})\vec{b} - \left(\vec{a} \cdot \vec{b}\right)\vec{c}$$

$$\vec{r}_i = x\hat{\imath} + y\hat{\jmath} + z\hat{k}$$
$$\vec{\omega} = \omega_x\hat{\imath} + \omega_z\hat{\jmath} + \omega_z\hat{k}$$

Here, $(\hat{\imath}, \hat{\jmath}, \hat{k})$ are unit vectors of a frame of our choice (or body frame).
We may want to set the origin of the frame at the center of mass of a body of interest and the frame orientation along the principal axis of body inertial tensor

$$\mathbf{L}_{in} = \sum_i \mathbf{L}_i$$
$$= \sum_i (\mathbf{R}_i \times m_i\mathbf{V}_i)$$
$$= \mathbf{R} \times M\mathbf{V} + \sum (\mathbf{r}_i \times m_i\mathbf{v}_i)$$

$$\sum(\vec{r}_i \times m_i\vec{v}_i)$$
$$= \sum(\vec{r}_i \times m_i(\vec{\omega} \times \vec{r}_i))$$
$$= \sum m_i\{(\vec{r}_i \cdot \vec{r}_i)\vec{\omega} - (\vec{r}_i \cdot \vec{\omega})\vec{r}_i\}$$
$$= \begin{bmatrix} y^2+z^2 & -xy & -xz \\ -xy & x^2+z^2 & -yz \\ -xz & -yz & x^2+y^2 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$
$$= \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

https://www.youtube.com/watch?v=HxH7fAQjt70
https://www.youtube.com/watch?v=BkszeogHGD0

Angular momentum

- Angular velocity in world coordinate, inertia of the body in the body coordinate

- Two options
  - Convert the angular velocity in world coordinate to body coordinate
    → Calculate angular momentum in body coordinate
    → Convert the angular momentum to world coordinate
  - Convert the inertia in the body coordinate to the world coordinate
    → Calculate angular momentum in world coordinate
- Don't forget the linear velocity term.

Inertia coordinate transfer

- https://en.wikipedia.org/wiki/Moment_of_inertia

Inertia tensor of rotation [edit]

Let $\mathbf{R}$ be the matrix that represents a body's rotation. The inertia tensor of the rotated body is given by:[27]

$$\mathbf{I} = \mathbf{R}\mathbf{I}_0\mathbf{R}^\mathsf{T}$$

mymat.hpp library

- For students, we provided mymat.hpp library, same as HW #2. You can use Vec3 and Mat33 class for dynamic calculation.