

# RQT Plugins

```
$ roscore
$ rosrunc turtlesim turtlesim_node
$ rosrunc turtlesim turtle_teleop_key
$ rqt #RQT graphical user interface
```

Available RQT plugins

<http://wiki.ros.org/rqt/Plugins>

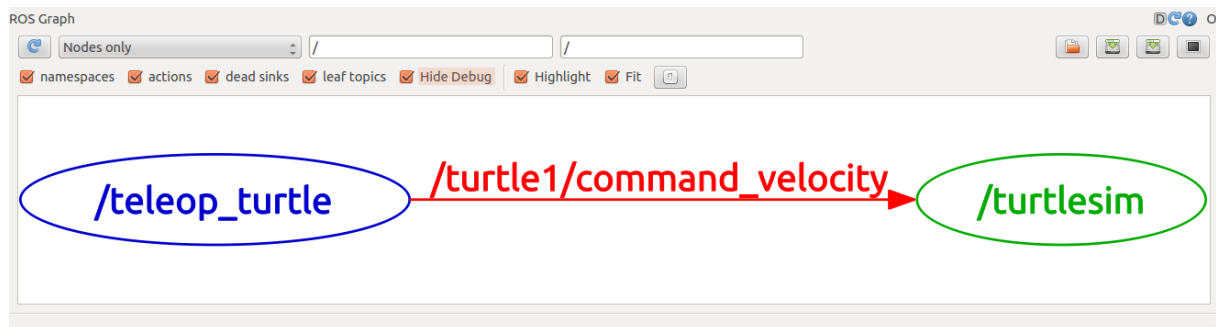
- **Node Graph**
- **Topic Monitor**
- **Message publisher**
- **Service caller**
- **rqt\_plot**
- **Launch (experimental)**
- **Logger Level**
- **Console**
- **Tf tree (tf\_echo, rviz)**

## Dynamic Reconfigure

- Vortrag Rodion Marynych, 11.1.2016

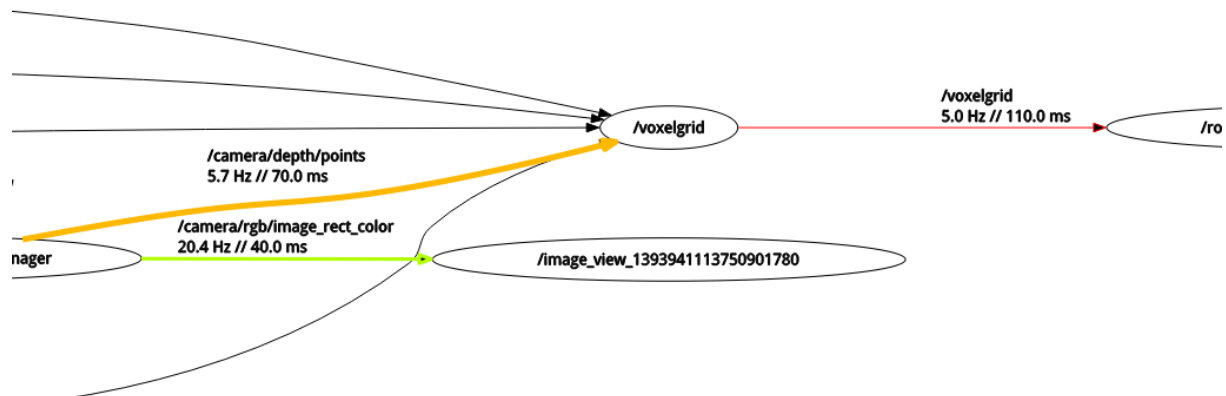
## Node Graph (\$ rqt\_graph)

- Choose Grape type
  - o „Nodes Only“
  - o “Nodes/Topics (active)”
  - o “Nodes/Topics (all)”
- Namespace Filter
- Topic Filter
- Group Namespaces (/turtle1/...)
- Highlight
  - o ROS nodes (here blue and green)
  - o Topics (here red)



## Topic Statistics

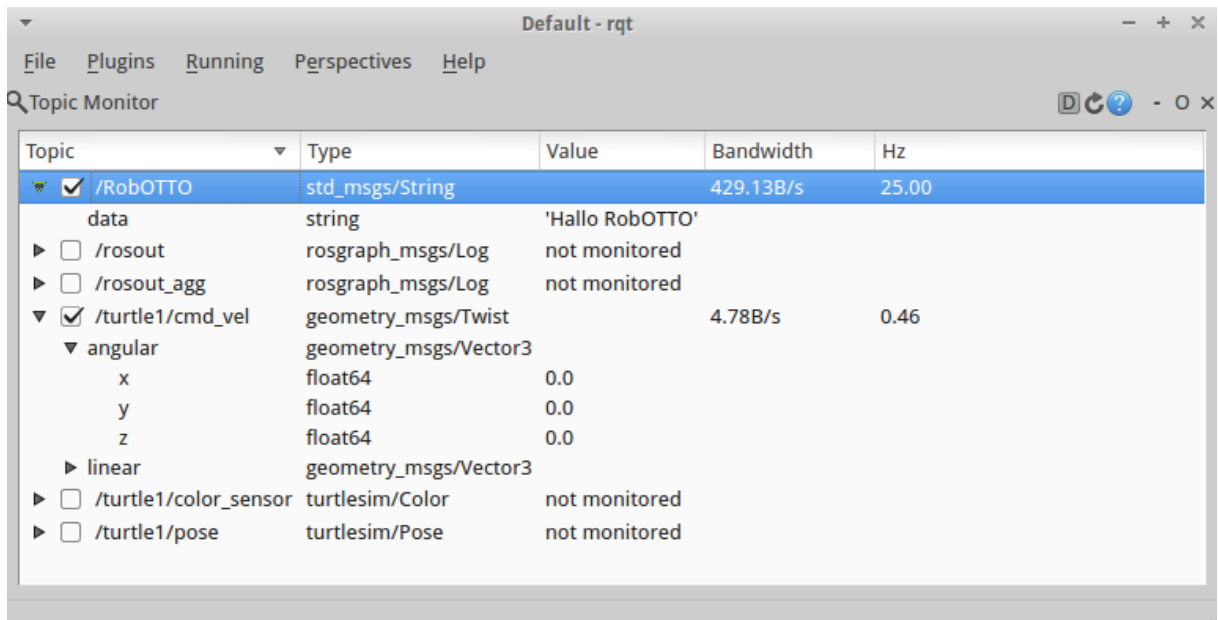
```
$ rosparam set enable_statistics true  
  
$ rostopic pub /RobOTTO std_msgs/String "data: 'Hallo  
  RobOTTO' " -r 25  
  
$ rostopic bw /RobOTTO
```



## Topic Monitor (\$ rosrun rqt\_topic rqt\_topic)

```
$ rostopic pub /RobOTTO std_msgs/String "data: 'Hallo  
RobOTTO' " -r 25
```

- topic monitor list
  - o Topic
  - o Type
  - o Value
  - o Bandwidth (Bytes per second)
  - o Frequency



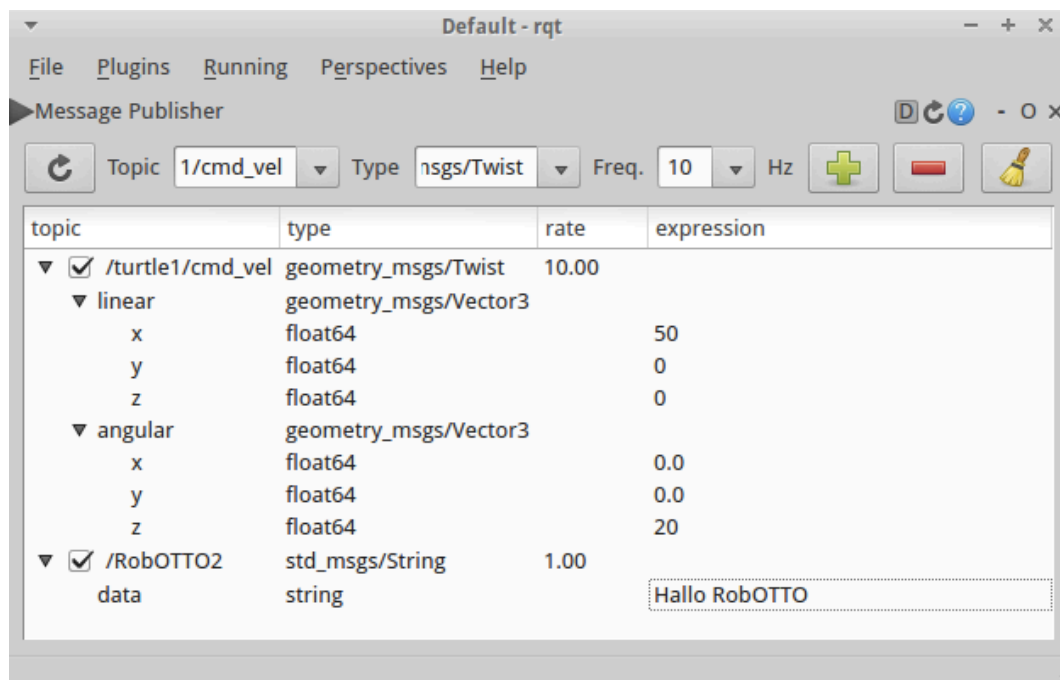
The screenshot shows the 'rqt' window with the 'Topic Monitor' panel. The table displays the following data:

Topic	Type	Value	Bandwidth	Hz
<input checked="" type="checkbox"/> /RobOTTO	std_msgs/String		429.13B/s	25.00
data	string	'Hallo RobOTTO'		
<input type="checkbox"/> /rosout	rosgraph_msgs/Log	not monitored		
<input type="checkbox"/> /rosout_agg	rosgraph_msgs/Log	not monitored		
<input checked="" type="checkbox"/> /turtle1/cmd_vel	geometry_msgs/Twist		4.78B/s	0.46
angular	geometry_msgs/Vector3			
x	float64	0.0		
y	float64	0.0		
z	float64	0.0		
linear	geometry_msgs/Vector3			
<input type="checkbox"/> /turtle1/color_sensor	turtlesim/Color	not monitored		
<input type="checkbox"/> /turtle1/pose	turtlesim/Pose	not monitored		

## Message publisher (\$ rosrun rqt\_publisher rqt\_publisher)

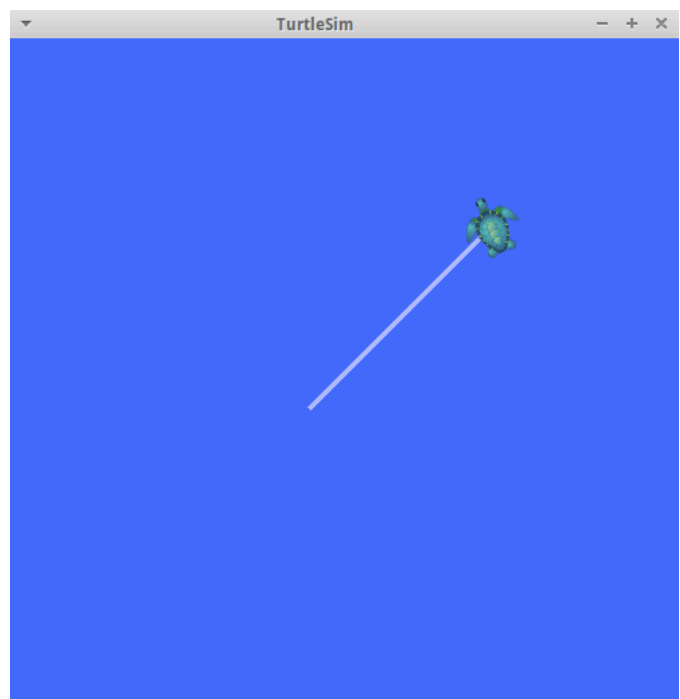
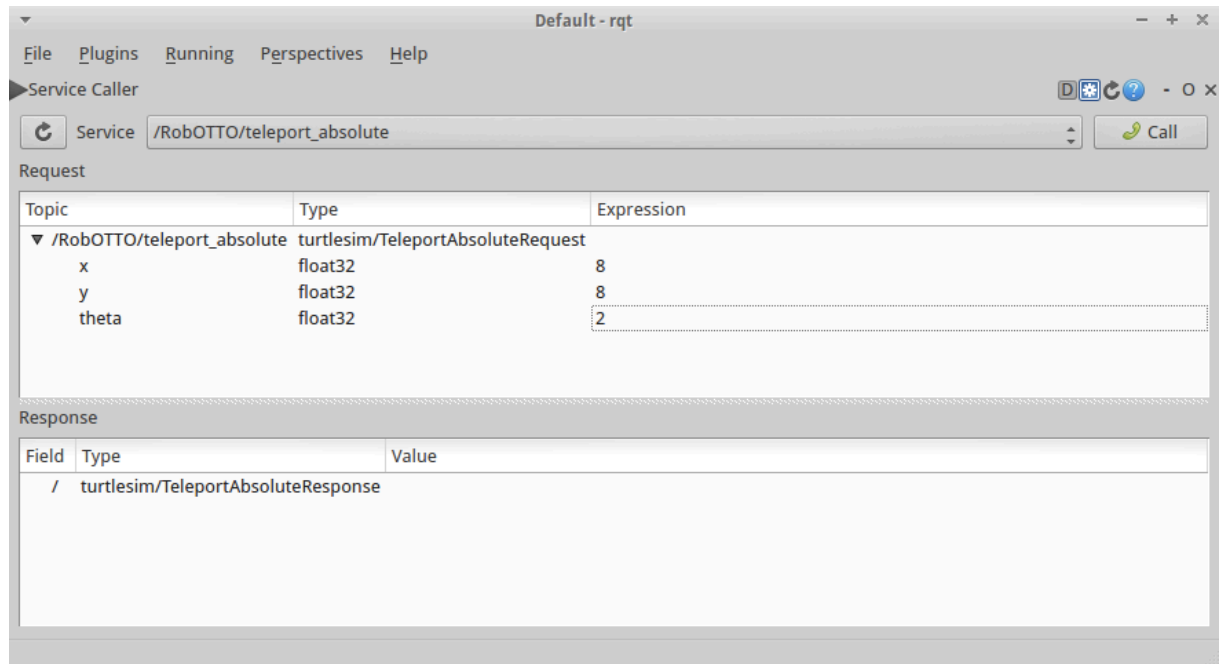
```
$ rostopic echo /RobOTTO2
```

- Create a Publisher
  - o Topic name to publish on
  - o Message Type to publish
  - o Frequency for periodic publishers in Hz (set to 0 for manual publishing)
  - o Add new publisher
  - o Remove selected publisher
  - o Clear all publishers



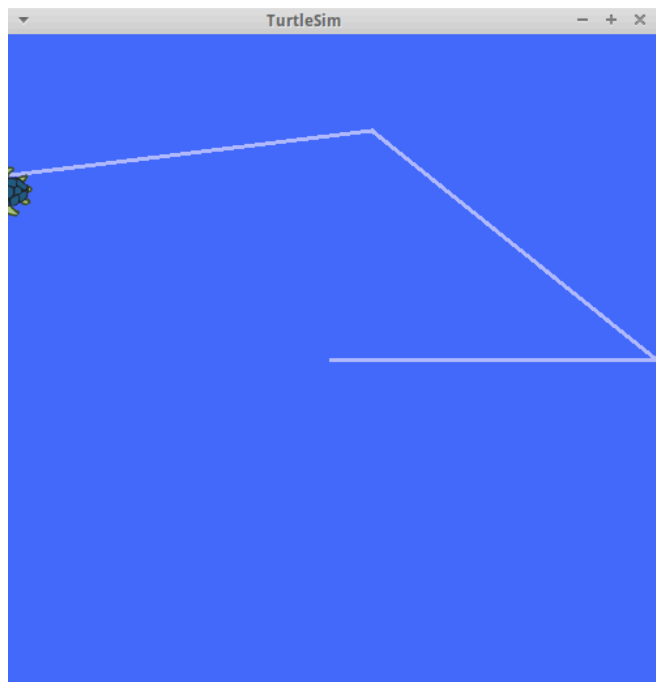
## Service caller (\$ rosrn rqt\_service\_caller rqt\_service\_caller)

- Call Service
  - o Select service from drop down menu
  - o Change the expressions in the request
  - o Call selected service
  - o Get the response in the lower window



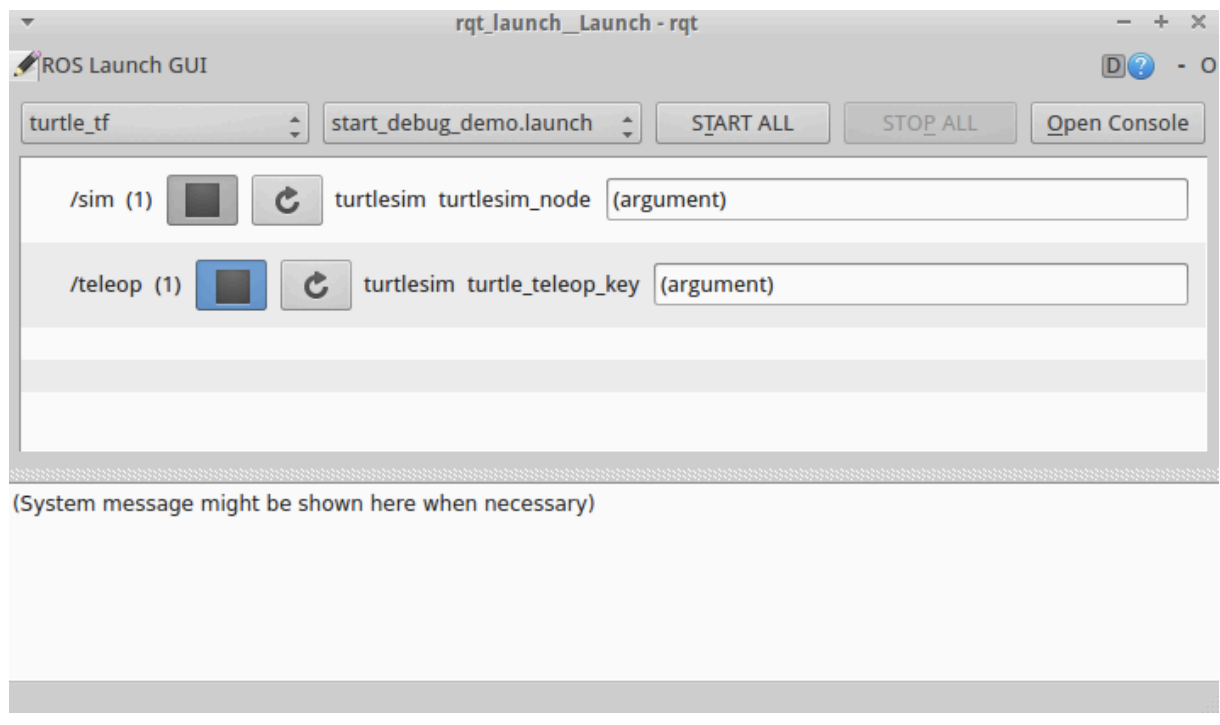
**rqt\_plot** (\$ rosrun rqt\_plot rqt\_plot)

- select the topic that is to be plotted
- autoscroll
- pause plot
- clear plot
- Pan axes with left mouse, zoom with right
- save the figure as .png
- edit curves and axes parameters



## **Launch (experimental)** (\$ rosrun rqt\_launch rqt\_launch)

- select a .launch files on the local file system
- See all nodes defined in the .launch file
- Run and stop nodes individually
- Start and stop all nodes at once
- Add an argument value before the node gets started



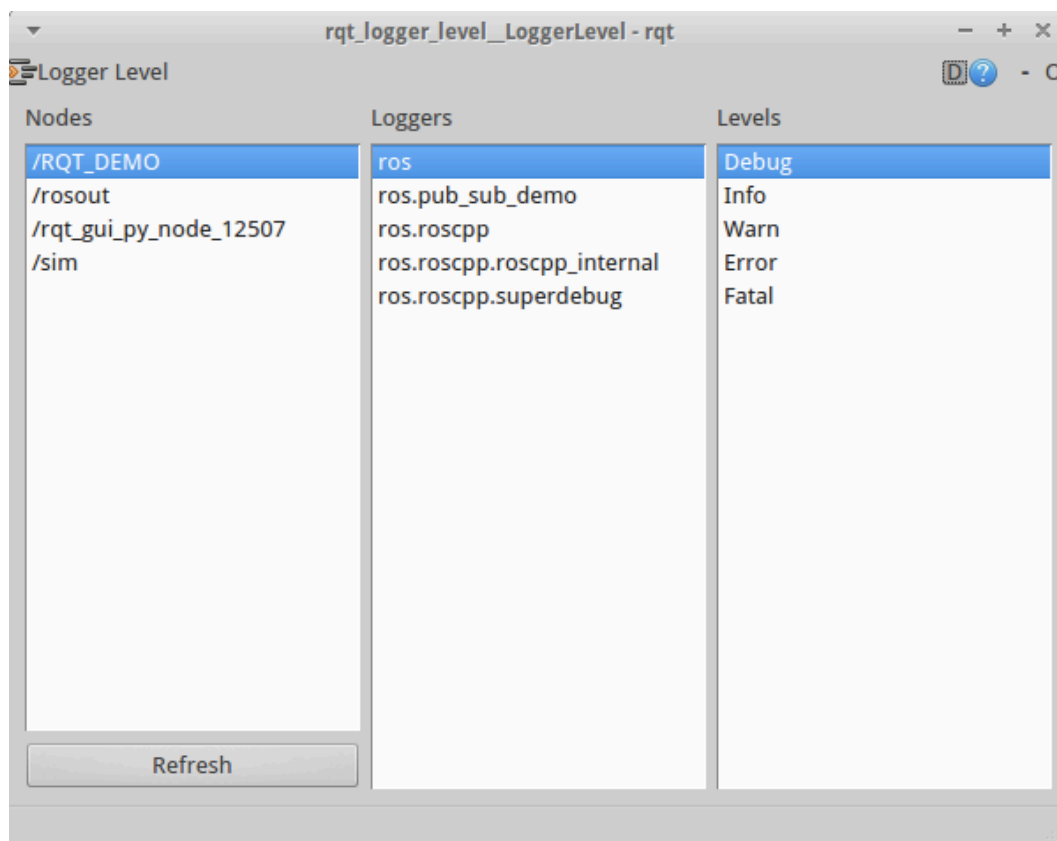
## Logger Level (\$ rosrun rqt\_logger\_level rqt\_logger\_level)

- rqt\_logger\_level is an application for adjusting the logger level of ros nodes.

```
$ rosrun pub_sub_demo node1
```

inside the node

```
ROS_INFO("hallo ich bin eine INFO");  
ROS_DEBUG("mich nennt man DEBUG");  
ROS_WARN("ich WARN dich!");  
ROS_ERROR("ich bin ein roter ERROR");  
ROS_FATAL("ich bin auch rot aber FATAL");
```





## Console (\$ rqt\_console)

Logging levels are prioritized in the following order:

- Fatal
- Error
- Warn
- Info
- Debug

- Pause/Resume the message stream
- Configuration
  - o Logger Level
- Exclude Messages
  - o ...containing
  - o ...from time range
  - o ...from topic
  - o Custom
- Highlight Messages
  - o ...containing
  - o ...from time range
  - o ...from topic
  - o Custom

The screenshot shows the rqt\_console interface with a table of messages and configuration panels. The table displays 5 messages, with 4 currently visible. The messages are:

#	Message	Severity	Node	Stamp	Topics	Location
#5	ich bin auch rot aber FATAL	Fatal	/RQT_DEMO	15:23:21.56...	/rosout	/home/...
#4	ich bin ein roter ERROR	Error	/RQT_DEMO	15:23:21.56...	/rosout	/home/...
#2	mich nennt man DEBUG	Debug	/RQT_DEMO	15:23:21.56...	/rosout	/home/...
#1	hallo ich bin eine INFO	Info	/RQT_DEMO	15:23:21.56...	/rosout	/home/...

Below the table, there are two configuration panels:

**Exclude Messages...**

- ...with severities: Debug Info Warn Error Fatal
- ...from node: /RQT\_DEMO

**Highlight Messages...**

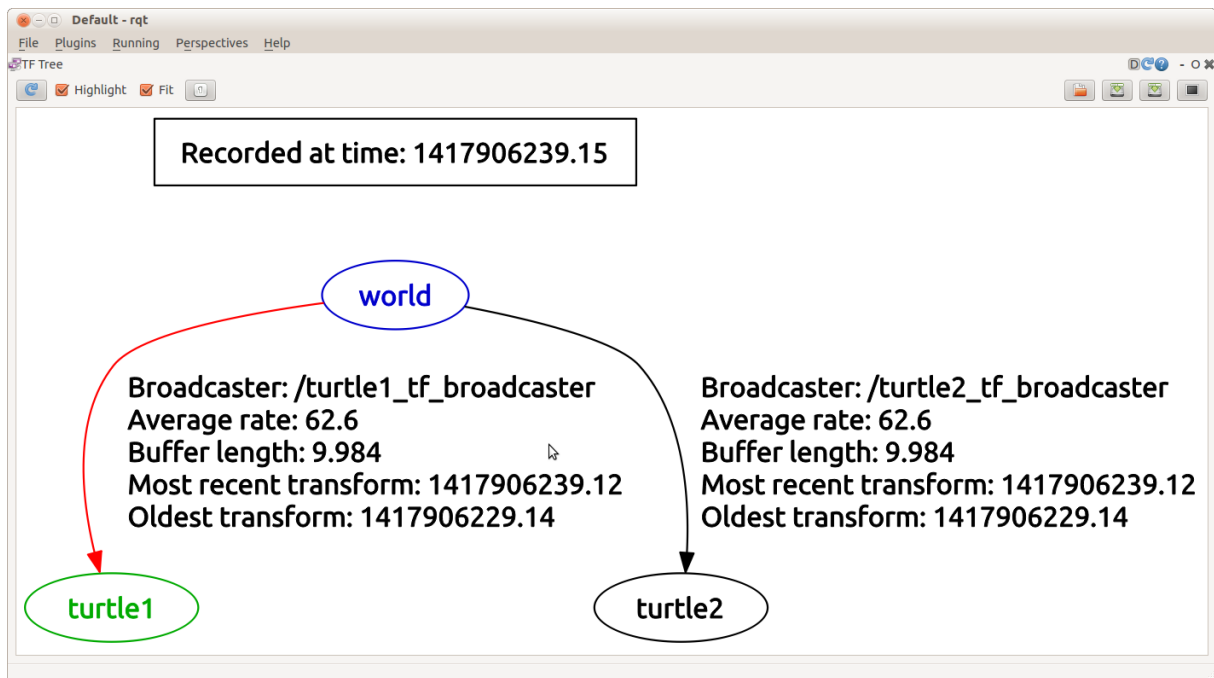
- ...containing: ich bin  Regex
- ...from node: /RQT\_DEMO
- ...with severities: Debug Info Warn Error Fatal

**Tf tree** (`$ rosrun rqt_tf_tree rqt_tf_tree`)

```
$ roslaunch turtle_tf turtle_tf_demo.launch
```

### Ros Wiki:

This demo is using the tf library to create three coordinate frames: a world frame, a turtle1 frame, and a turtle2 frame. This tutorial uses a tf broadcaster to publish the turtle coordinate frames and a tf listener to compute the difference in the turtle frames and move one turtle to follow the other.



We can see that "world" is the parent of the "turtle1" and "turtle2" frames and some more informations for debugging purposes.

**tf\_echo** (not a rqt plugin)

```
$ rosrun tf tf_echo world turtle1  
$ rosrun tf tf_echo turtle1 world
```

**rviz** (rosrun rviz rviz)

```
$ rosrun rviz rviz -d `rospack find  
turtle_tf`/rviz/turtle_rviz.rviz
```