Moving rules

$$\begin{cases} v & = & const \\ v & = & \sqrt{(y_{target} - y_{self})^2 + (x_{target} - x_{self})^2} \end{cases}$$

$$\Delta\theta = \arctan\left(\frac{y_{target} - y_{self}}{x_{target} - x_{self}}\right) - \theta_{self}$$

$$\omega = \frac{\Delta\theta}{\Delta t} = \frac{\arctan\left(\frac{y_{target} - y_{self}}{x_{target} - x_{self}}\right) - \theta_{self}}{\Delta t}$$

Run a demo

In one terminal

```
  $ rosrun turtlesim turtlesim_node
```

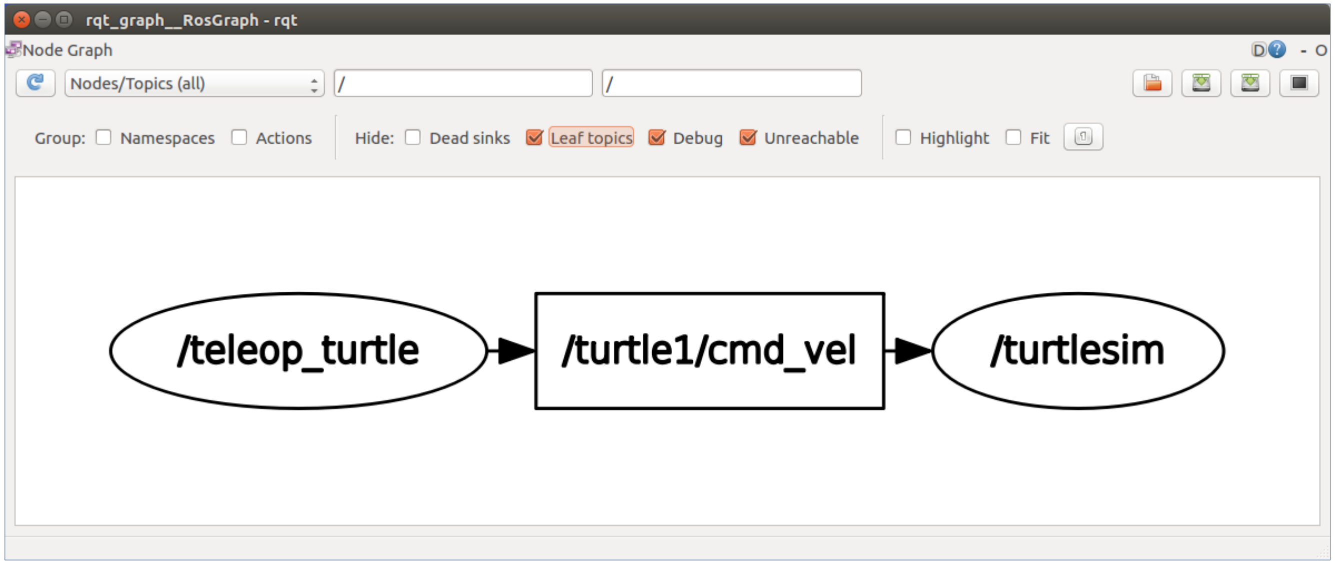
This opens a window with a turtle

In another teminal

```
  $ rosrun turtlesim turtle_teleop_key
```

This runs a tool for controlling a turtle (with arrows)

Nodes & Topics


rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (all)
/
/
Group: Namespaces Actions
Hide: Dead sinks [✓ Leaf topics] [✓ Debug] [✓ Unreachable]
Highlight Fit
/teleop_turtle
/turtle1/cmd_vel
/turtlesim

```
$ rqt_graph
```

Useful commands

```
$ rosnode list
$ rosnode info <nodename>

$ rostopic list
$ rostopic info <topicname>
```

Create, make, set paths & run

```
$ catkin_create_pkg <package> <depends>
```

Notes:

- Execute from <worksapce>/src

```
$ catkin_make
```

- Execute from <worksapce>

```
$ source <workspace>/devel/setup.bash
```

```
$ rosrun <package> <type> __name:=<name>
```

File system structure

```
workspace_folder        ←
  ┣bin    (auto-generated)
  ┣devel  (auto-generated)
  ┗src                  ←
    ┣package#1_folder
    ┗package#2_folder
      ┣CmakeLists.txt (auto-generated)
      ┣package.xml    (auto-generated)
      ┗<other staff>
```

Publish to topic:

```
$ rostopic pub /turtle1/color_sensor
turtlesim/Color "{r: 0, g: 0, b: 255}"
```

```
#! /usr/bin/env python
import rospy
from turtlesim.msg import Color

rospy.init_node("turtle_color_blue")
pub=rospy.Publisher(“/turtle1/color_sensor”,
                    Color,
                    queue_size=10)
msg=Color(r=0,g=0,b=255)
pub.publish(msg)
```

Subscribe to topic:

```
$ rostopic echo /turtle1/color_sensor
```

```
#! /usr/bin/env python
import rospy
from turtlesim.msg import Color

def callback(msg):
  print(“turtle color: ” + str(msg))

rospy.init_node("turtle_get_color")
rospy.Subscriber(“/turtle1/color_sensor”,
                 Color,
                 callback)
rospy.spin()
```

Call a service:

```
$ rosservice call /spawn "{x: 0.0, y: 0.0,
theta: 0.0, name: 'victim'}"
```

```
#! /usr/bin/env python
import rospy
from turtlesim.srv import Spawn

rospy.init_node("spawn_caller")
rospy.wait_for_service(“/spawn”)
spawn_func=rospy.ServiceProxy(“/spawn”,
                              Spawn)
res = spawn_func(4.0, 4.0, 0.0, ”victim”)
# isinstance(res, SpawnResponse) == True
```

Handle a service:

```
#! /usr/bin/env python
import rospy
from turtlesim.srv import Spawn
from turtlesim.srv import SpawnResponse

def callback(req):
  # isinstance(req, SpawnRequest) == True
  print(“spawn request: ” + str(req))
  return SpawnResponse(req.name)

rospy.init_node("spawn_handler")
rospy.Service(“/spawn”,
              Spawn,
              callback)
rospy.spin()
```

Launch file

```
$ roslaunch <package> <file>.launch
```

```
<launch>
  <param name="a" value="0"/>
  <node pkg="turtlesim"
        type="turtlesim_node"
        name="simulator"
        output="screen"/>
  <node pkg="p" type="a.py" name="v">
    <param name="p_param" value="p"/>
    <remap from="old_topic_name"
           to="new_topic_name"/>
  </node>
</launch>
```