02/13/16 Turtlesim Cheat Sheet

1. $ roscore (leave running but minimize)

2. 2nd Terminal

3. $ **rosrun turtlesim turtlesim_node**
(See the turtle with Blue Background – leave terminal window running and view turtle)

4. 3rd Terminal
$ rosnode info turtlesim (Determine node information)

```
tlharmanphd@D125-43873:~$ rosnode info turtlesim
--------------------------------------------------------------------------------
Node [/turtlesim]
Publications:
 * /turtle1/color_sensor [turtlesim/Color]
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/pose [turtlesim/Pose]

Subscriptions:
 * /turtle1/cmd_vel [unknown type]

Services:
 * /turtle1/teleport_absolute
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level
 * /reset
 * /spawn
 * /clear
 * /turtle1/set_pen
 * /turtle1/teleport_relative
 * /kill


contacting node http://D125-43873:41890/ ...
Pid: 7420
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
```

NOW YOU HAVE THE INFORMATION TO CONTROL TURTLESIM!

- - - - - -
Look at Colors
$ rostopic echo /turtle1/color_sensor (r: b: g: Cntl+C to stop)
- - - -
rosparam get parameters and change color of background to Red
tlharmanphd@D125-43873:~$ **rosparam get /**

```
background_b: 255
background_g: 86
background_r: 69
rosdistro: 'indigo
roslaunch:
  uris: {host_d125_43873__60512: 'http://D125-43873:60512/'}
rosversion: '1.11.10
run_id: 2429b792-d23c-11e4-b9ee-3417ebbca982
```

rosparam set
Change the colors:
tlharmanphd@D125-43873:/$ **rosparam set background_b 0**
tlharmanphd@D125-43873:/$ **rosparam set background_g 0**
tlharmanphd@D125-43873:/$ **rosparam set background_r 255**
tlharmanphd@D125-43873:/$ **rosservice call /clear (See Red!)**

- - - -
Services Absolute and Relative Move
$ rosservice call /turtle1/teleport_absolute 1 1 0 (Move to 1,1)

$ rosservice call /turtle1/teleport_relative 1 0

- - - -
Check Turtle1's pose
$ rostopic echo /turtle1/pose

```
x: 1.0
y: 1.0
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
```

LETS CONTROL THE TURTLE- Publish to /turtle1/cmd_vel:
(roscore and turtlesim_node running)

1. Command line
2. Keyboard
3. Joystick
4. Python

Subscriptions: /turtle1/cmd_vel [unknown type]

$ rostopic type /turtle1/cmd_vel

geometry_msgs/Twist

1a. Move a bit 1 command

$ **rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'**

1b. Move in a circle repeat at frequency $ **rostopic hz /turtle1/pose**

$ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'

2. $ rosrun turtlesim turtle_teleop_key (Cntl+c to exit)

Reading from keyboard

Use arrow keys to move the turtle.

```
Up arrow       Turtle up
Down arrow     Turtle down
Right arrow    Rotate CW
Left  arrow    Rotate CCW
```

$ rqt_graph (See the namespaces, nodes and topics)

3. Joystick

4. Python Creates node /ControlTurtlesim ; Publishes t0 /turtle1/cmd_vel with Twist msg

$ python turtlesim1.py (Make Executable $chmod +x turtlesim1.py

```
[INFO] [WallTime: 1455313387.186692]  Press CTRL+c to stop TurtleBot
[INFO] [WallTime: 1455313387.188315]  Set rate 10Hz
```

```
#!/usr/bin/env python   turtlesim1.py
# Execute as a python script
# Set linear and angular values of Turtlesim's speed and turning.
import rospy                                # Needed to create a ROS node
from geometry_msgs.msg import Twist    # Message that moves base

class ControlTurtlesim():
    def __init__(self):
        # ControlTurtlesim is the name of the node sent to the master
        rospy.init_node('ControlTurtlesim', anonymous=False)

                # Message to screen
        rospy.loginfo(" Press CTRL+c to stop TurtleBot")

        # Keys CNTL + c will stop script
        rospy.on_shutdown(self.shutdown)

         # Publisher will send Twist message on topic
        # /turtle1/cmd_vel

        self.cmd_vel = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)

             # Turtlesim will receive the message 10 times per second.
        rate = rospy.Rate(10);
             # 10 Hz is fine as long as the processing does not exceed
        #   1/10 second.
        rospy.loginfo(" Set rate 10Hz")
        # Twist is geometry_msgs for linear and angular velocity
        move_cmd = Twist()
             # Linear speed in x in meters/second is + (forward) or
        # - (backwards)
        move_cmd.linear.x = 0.3    # Modify this value to change speed
             # Turn at 0 radians/s
        move_cmd.angular.z = 0
        # Modify this value to cause rotation rad/s

             # Loop and TurtleBot will move until you type CNTL+c
        while not rospy.is_shutdown():
                # publish Twist values to the Turtlesim node /cmd_vel
            self.cmd_vel.publish(move_cmd)
                # wait for 0.1 seconds (10 HZ) and publish again
            rate.sleep()



    def shutdown(self):
        # You can stop turtlebot by publishing an empty Twist message
        rospy.loginfo("Stopping Turtlesim")

        self.cmd_vel.publish(Twist())
             # Give TurtleBot time to stop
        rospy.sleep(1)

if __name__ == '__main__':
```

```
    try:
        ControlTurtlesim()
    except:
        rospy.loginfo("End of the trip for Turtlesim")
```

```
$ rosnode list
/ControlTurtlesim
/rosout
/teleop_turtle
/turtlesim
```

Change Python script turtlesim1.py to run turtle in a circle (Make executable)
In ros_ws

tlharmanphd@D125-43873:~/ros_ws$ **python turtlesim2.py**
[INFO] [WallTime: 1455383932.566053] Press CTRL+c to stop TurtleBot
[INFO] [WallTime: 1455383932.567306] Set rate 10Hz
^C[INFO] [WallTime: 1455383937.538077] Stopping Turtlesim

In turtlesim2.py Script
move_cmd = Twist()
	# Linear speed in x in meters/second is + (forward) or
	# - (backwards)
	move_cmd.linear.x = 2.0	# Modify this value to change speed
	# Turn at 1.8 radians/s
	move_cmd.angular.z = 1.8
	# Modify this value to cause rotation rad/s