Subscriber整合

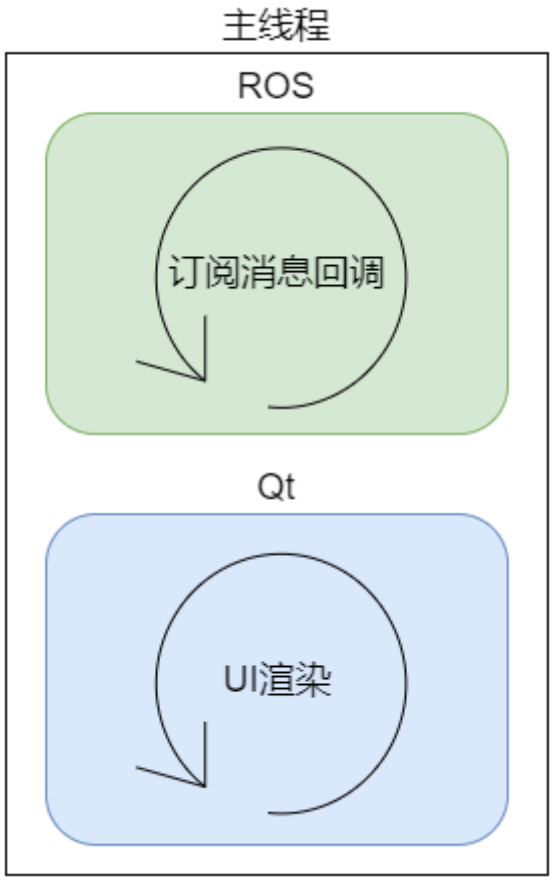
1. 添加subscriber创建

```
poseTopicName = "/turtle1/pose"
rospy.Subscriber(poseTopicName, Pose, self.poseCallback)
```

2. 添加订阅回调

```
def poseCallback(self, msg):
    print msg
```

3. 调试



与C++不同，调试结果为可以正常接收到订阅的消息。但是关于窗体关闭事件方面只能通过关闭QT界面来关闭进程，不同通过命令行来。

Python可以正常接收到订阅消息的原因，在于Python采用的SIP方式进行操作QT的界面UI的中间是一套异步策略。

但是为了更好的优化体验，我们也是可以和C++实现的代码逻辑想通的。

接管渲染：

```
updateTimer = QTimer(self)
updateTimer.setInterval(16)
updateTimer.start()

updateTimer.timeout.connect(self.onUpdate)
```

渲染逻辑:

```
def onUpdate(self):
    self.update()
    if rospy.is_shutdown():
        self.close()
```

完整示例代码

MainWindow

```
#!/usr/bin/env python
# coding: utf-8

from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
import rospy
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose
from math import radians, degrees


class MainWindow(QWidget):
    def __init__(self):
        super(MainWindow, self).__init__()

        # 自定义刷新
        updateTimer = QTimer(self)
        updateTimer.setInterval(16)
        updateTimer.start()

        updateTimer.timeout.connect(self.onUpdate)

        # 设置title
        self.setWindowTitle("小乌龟控制")
        self.resize(400, 120)

        # 设置布局
        layout = QFormLayout()
        self.setLayout(layout)

        # 添加控件
        self.editLinear = QLineEdit("0")
        layout.addRow("线速度", self.editLinear)

```

```
        self.editAngular = QLineEdit("0")
        layout.addRow("角速度", self.editAngular)

        self.labelX = QLabel()
        layout.addRow("当前X坐标", self.labelX)

        self.labelY = QLabel()
        layout.addRow("当前Y坐标", self.labelY)

        self.labelLinear = QLabel()
        layout.addRow("当前线速度", self.labelLinear)

        self.labelAngular = QLabel()
        layout.addRow("当前角速度", self.labelAngular)

        self.labelDegrees = QLabel()
        layout.addRow("当前角度", self.labelDegrees)

        self.btnSend = QPushButton("发送")
        layout.addRow(self.btnSend)

        # 添加事件
        self.btnSend.clicked.connect(self.clickSend)

        # 创建publisher
        topicName = "/turtle1/cmd_vel"
        self.publisher = rospy.Publisher(topicName, Twist, queue_size=1000)

        poseTopicName = "/turtle1/pose"
        rospy.Subscriber(poseTopicName, Pose, self.poseCallback)

    def clickSend(self):
        linearX = float(self.editLinear.text())
        angularZ = radians(float(self.editAngular.text()))

        # 构建消息
        twist = Twist()
        twist.linear.x = linearX
        twist.angular.z = angularZ
        # 发布
        self.publisher.publish(twist)

    def poseCallback(self, msg):
        if not isinstance(msg, Pose):
            return
        self.labelX.setText(str(msg.x))
        self.labelY.setText(str(msg.y))
        self.labelLinear.setText(str(msg.linear_velocity))
        self.labelAngular.setText(str(msg.angular_velocity))
        self.labelDegrees.setText(str(degrees(msg.theta)))

    def onUpdate(self):
        self.update()

        if rospy.is_shutdown():
            self.close()
```