

## 坐标系的广播&监听/ CPP的龟实验

2022年11月23日 星期三 21:13

## 创建功能包

- 创建功能包

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg learning_tf roscpp rospy tf turtlesim
```



编译

- 配置tf广播器与监听器代码编译规则

```
# Specify libraries to link a library or executable target against
# target_link_libraries($PROJECT_NAME_node
#   ${catkin_LIBRARIES}
# )
```

## 如何配置CMakeLists.txt中的编译规则

- 设置需要编译的代码和生成的可执行文件;
- 设置链接库;

编译可执行文件

```
add_executable(turtle_tf_broadcaster src/turtle_tf_broadcaster.cpp)
target_link_libraries(turtle_tf_broadcaster ${catkin_LIBRARIES})
```

```
add_executable(turtle_tf_listener src/turtle_tf_listener.cpp)
target_link_libraries(turtle_tf_listener ${catkin_LIBRARIES})
```

CMakeLists.txt

- 创建tf广播器代码 (C++)

```

# 1. 设置画布大小和坐标轴，并设置 Turtle 画笔的初始位置
turtle.setup(600, 600)
turtle.penup()
turtle.goto(-300, -300)
turtle.pendown()

# 2. 设置画笔的粗细和颜色
turtle.pensize(2)
turtle.pencolor('red')

# 3. 开始绘制正方形
for i in range(4):
    turtle.forward(300)
    turtle.right(90)

# 4. 结束绘制并显示结果
turtle.done()

```

## 如何实现一个tf广播器

- 定义TF广播器 (TransformBroadcaster)
- 创建坐标变换值;
- 发布坐标变换 (sendTransform)

turtle\_tf\_broadcaster.cpp

- 创建tf监听器代码 (C++)

[illegible]

## 如何实现一个TF监听器

- 定义TF监听器;  
(TransformListener)
- 查找坐标变换;  
(waitForTransform、lookupTransform)

turtle\_tf\_listener.cpp

- 编译并运行

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ roslaunch turtlesim turtlesim.launch
$ roslaunch learning_tf turtle.launch
$ roslaunch learning_tf turtle.launch
$ roslaunch learning_tf turtle.launch
$ roslaunch turtlesim turtle.launch
```

坐标系: Eucled

f\_broadcaster /turtle1  
f\_broadcaster /turtle2 } 运行可执行files

查2座标高 同808类系  
命令 turtle2 跟附

Key ~~to~~ turb

同时出现  
turtle)