ROS----小乌龟之你追我赶

生成八个小乌龟，分别命名为/turtle1、/turtle2、/turtle3、/turtle4、/turtle5、/turtle6、/turtle7、/turtle8然后实现turtle1自己运动，然后turtle2追turtle1，记为turtle2->turtle1。最终实现turtle8->turtle7->turtle6->turtle5->turtle4->turtle3->turtle2->turtle1

这其中涉及到tf变换相关的内容:

在该项目中有三个文件pursue\_turtle\_control.cpp、pursue\_turtle.cpp、pursue\_turtle.launch

其中，pursue\_turtle.cpp用来发布某一个小乌龟和世界坐标的tf变换，根据传入的参数，确定到底是哪一个小乌龟。

pursue\_turtle\_control.cpp中产生所需要的小乌龟，同时查询tf树，确定相邻两个乌龟之间的相对方向，并发布对应乌龟的运动topic。

pursue\_turtle.launch是launch文件主要用来配置一些参数和同时启动多个节点。

在pursue\_turtle.cpp中，首先订阅传入名字小乌龟的/pose，得到对应小乌龟的位置(x，y)和角度theta。然后在对应的消息订阅回调函数中得到的信息进行相应的计算，得到平移和旋转向量，最后打包成固定的格式发布出去。

#include <ros/ros.h>

#include <tf/transform\_broadcaster.h>

#include <tf/tf.h>

#include <turtlesim/Pose.h>

std::string turtle\_name;

void poseCallback(const turtlesim::PoseConstPtr& pose)

{

static tf::TransformBroadcaster br;

tf::Transform transform;

transform.setOrigin(tf::Vector3(pose->x,pose->y,0));

tf::Quaternion q;

q.setRPY(0,0,pose->theta);

transform.setRotation(q);

br.sendTransform(tf::StampedTransform(transform,ros::Time::now(),"world",turtle\_name));

}

int main(int argc, char\*\* argv)

{

ros::init(argc,argv,"my\_life");

turtle\_name = argv[1];

ros::NodeHandle n;

ros::Subscriber sub = n.subscribe(turtle\_name+"/pose", 10, poseCallback);

ros::spin();

return 0;

}

接下来在tf树中查找相应的小乌龟之间的变换关系:

tf::TransformListener listener;

while(n.ok())

{

cl.call(e);

tf::StampedTransform transform2,transform3,transform4,transform5,transform6,transform7,transform8;

try{

listener.waitForTransform("/turtle2","/turtle1",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle2","/turtle1",ros::Time(0),transform2);

listener.waitForTransform("/turtle3","/turtle2",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle3","/turtle2",ros::Time(0),transform3);

listener.waitForTransform("/turtle4","/turtle3",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle4","/turtle3",ros::Time(0),transform4);

listener.waitForTransform("/turtle5","/turtle4",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle5","/turtle4",ros::Time(0),transform5);

listener.waitForTransform("/turtle6","/turtle5",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle6","/turtle5",ros::Time(0),transform6);

listener.waitForTransform("/turtle7","/turtle6",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle7","/turtle6",ros::Time(0),transform7);

listener.waitForTransform("/turtle8","/turtle1",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle8","/turtle1",ros::Time(0),transform8);

}catch(tf::TransformException& ex)

{

ROS\_ERROR("%s",ex.what());

ros::Duration(1.0).sleep();

continue;

}

最后将查询到的变换关系转换成相应的运动:

geometry\_msgs::Twist t;

t.angular.z = 1.0\*atan2(transform2.getOrigin().y(),transform2.getOrigin().x());;

t.linear.x =0.5\*sqrt(pow(transform2.getOrigin().x(),2)+pow(transform2.getOrigin().y(),2));;

pub2.publish(t);

t.angular.z = 1.0\*atan2(transform3.getOrigin().y(),transform3.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform3.getOrigin().x(),2)+pow(transform3.getOrigin().y(),2));;

pub3.publish(t);

t.angular.z = 1.0\*atan2(transform4.getOrigin().y(),transform4.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform4.getOrigin().x(),2)+pow(transform4.getOrigin().y(),2));;

pub4.publish(t);

t.angular.z = 1.0\*atan2(transform5.getOrigin().y(),transform5.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform5.getOrigin().x(),2)+pow(transform5.getOrigin().y(),2));;

pub5.publish(t);

t.angular.z = 1.0\*atan2(transform6.getOrigin().y(),transform6.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform6.getOrigin().x(),2)+pow(transform6.getOrigin().y(),2));;

pub6.publish(t);

t.angular.z = 1.0\*atan2(transform7.getOrigin().y(),transform7.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform7.getOrigin().x(),2)+pow(transform7.getOrigin().y(),2));;

pub7.publish(t);

t.angular.z = 1.0\*atan2(transform8.getOrigin().y(),transform8.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform8.getOrigin().x(),2)+pow(transform8.getOrigin().y(),2));;

pub8.publish(t);

完整的代码:

#include <ros/ros.h>

#include <turtlesim/Spawn.h>

#include <geometry\_msgs/Twist.h>

#include <tf/transform\_listener.h>

#include <std\_srvs/Empty.h>

#include <ctime>

int main(int argc,char\*\* argv)

{

ros::init(argc,argv,"control");

ros::NodeHandle n;

ros::service::waitForService("spawn");

// generation turtle

ros::ServiceClient client = n.serviceClient<turtlesim::Spawn>("spawn");

ros::ServiceClient cl = n.serviceClient<std\_srvs::Empty>("clear");

std\_srvs::Empty e;

turtlesim::Spawn turtle;

srand((unsigned int)time(NULL));

for(int i = 2;i < 9;i++)

{

turtle.request.x = rand()%8+1;

turtle.request.y = rand()%8+1;

client.call(turtle);

}

ros::Publisher pub1 = n.advertise<geometry\_msgs::Twist>("/turtle1/cmd\_vel", 10);

ros::Publisher pub2 = n.advertise<geometry\_msgs::Twist>("/turtle2/cmd\_vel", 10);

ros::Publisher pub3 = n.advertise<geometry\_msgs::Twist>("/turtle3/cmd\_vel", 10);

ros::Publisher pub4 = n.advertise<geometry\_msgs::Twist>("/turtle4/cmd\_vel", 10);

ros::Publisher pub5 = n.advertise<geometry\_msgs::Twist>("/turtle5/cmd\_vel", 10);

ros::Publisher pub6 = n.advertise<geometry\_msgs::Twist>("/turtle6/cmd\_vel", 10);

ros::Publisher pub7 = n.advertise<geometry\_msgs::Twist>("/turtle7/cmd\_vel", 10);

ros::Publisher pub8 = n.advertise<geometry\_msgs::Twist>("/turtle8/cmd\_vel", 10);

tf::TransformListener listener;

while(n.ok())

{

cl.call(e);

tf::StampedTransform transform2,transform3,transform4,transform5,transform6,transform7,transform8;

try{

listener.waitForTransform("/turtle2","/turtle1",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle2","/turtle1",ros::Time(0),transform2);

listener.waitForTransform("/turtle3","/turtle2",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle3","/turtle2",ros::Time(0),transform3);

listener.waitForTransform("/turtle4","/turtle3",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle4","/turtle3",ros::Time(0),transform4);

listener.waitForTransform("/turtle5","/turtle4",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle5","/turtle4",ros::Time(0),transform5);

listener.waitForTransform("/turtle6","/turtle5",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle6","/turtle5",ros::Time(0),transform6);

listener.waitForTransform("/turtle7","/turtle6",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle7","/turtle6",ros::Time(0),transform7);

listener.waitForTransform("/turtle8","/turtle1",ros::Time(0),ros::Duration(3.0));

listener.lookupTransform("/turtle8","/turtle1",ros::Time(0),transform8);

}catch(tf::TransformException& ex)

{

ROS\_ERROR("%s",ex.what());

ros::Duration(1.0).sleep();

continue;

}

geometry\_msgs::Twist t;

t.angular.z = 1.0\*atan2(transform2.getOrigin().y(),transform2.getOrigin().x());;

t.linear.x =0.5\*sqrt(pow(transform2.getOrigin().x(),2)+pow(transform2.getOrigin().y(),2));;

pub2.publish(t);

t.angular.z = 1.0\*atan2(transform3.getOrigin().y(),transform3.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform3.getOrigin().x(),2)+pow(transform3.getOrigin().y(),2));;

pub3.publish(t);

t.angular.z = 1.0\*atan2(transform4.getOrigin().y(),transform4.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform4.getOrigin().x(),2)+pow(transform4.getOrigin().y(),2));;

pub4.publish(t);

t.angular.z = 1.0\*atan2(transform5.getOrigin().y(),transform5.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform5.getOrigin().x(),2)+pow(transform5.getOrigin().y(),2));;

pub5.publish(t);

t.angular.z = 1.0\*atan2(transform6.getOrigin().y(),transform6.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform6.getOrigin().x(),2)+pow(transform6.getOrigin().y(),2));;

pub6.publish(t);

t.angular.z = 1.0\*atan2(transform7.getOrigin().y(),transform7.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform7.getOrigin().x(),2)+pow(transform7.getOrigin().y(),2));;

pub7.publish(t);

t.angular.z = 1.0\*atan2(transform8.getOrigin().y(),transform8.getOrigin().x());;

t.linear.x = 0.5\*sqrt(pow(transform8.getOrigin().x(),2)+pow(transform8.getOrigin().y(),2));;

pub8.publish(t);

t.angular.z = rand()%3;

t.linear.x = rand()%7;

pub1.publish(t);

ros::spinOnce();

}

return 0;

}

最后看一下pursue\_turtle.launch文件:

<launch>

//用来启动显示小乌龟的界面

<node pkg="turtlesim" type="turtlesim\_node" name="star"/>

//启动pursue\_turtle\_control节点，产生小乌龟

<node pkg="base" type="pursue\_turtle\_control" name="road"/>

//启动八个节点，发布每个小乌龟和世界坐标之间的变换关系。

<node pkg="base" type="pursue\_turtle" args="/turtle1" name="turtle1\_world\_broadcaster"/>

<node pkg="base" type="pursue\_turtle" args="/turtle2" name="turtle2\_world\_broadcaster"/>

<node pkg="base" type="pursue\_turtle" args="turtle3" name="turtle3\_world\_broadcaster"/>

<node pkg="base" type="pursue\_turtle" args="turtle4" name="turtle4\_world\_broadcaster"/>

<node pkg="base" type="pursue\_turtle" args="turtle5" name="turtle5\_world\_broadcaster"/>

<node pkg="base" type="pursue\_turtle" args="turtle6" name="turtle6\_world\_broadcaster"/>

<node pkg="base" type="pursue\_turtle" args="turtle7" name="turtle7\_world\_broadcaster"/>

<node pkg="base" type="pursue\_turtle" args="turtle8" name="turtle8\_world\_broadcaster"/>

</launch>