3806ICT - Week 3 Lab

Nick van der Merwe - s5151332 - nick.vandermerwe@griffithuni.edu.au

March 25, 2021

1 Task 1 - Programming Exercise

Figure 1: The services file

services.cpp

```cpp
#include "ros/ros.h"
#include "week3/set_target.h"
#include "week3/distance_to_target.h"
#include "geometry_msgs/Vector3.h"
#include "turtlesim/Pose.h"
#include <cmath>
#include <iostream>

geometry_msgs::Vector3 Vector3Ctor(double x, double y, double z){
    // For some reason the vector3 ctor is broken so here's a custom one
    geometry_msgs::Vector3 toReturn;
    toReturn.x = x;
    toReturn.y = y;
    toReturn.z = z;
    return toReturn;
}

geometry_msgs::Vector3 target;

bool setTargetFunction(week3::set_target::Request & req,
        week3::set_target::Response & res){
    // Set the target to always stay the same
    // World runs from 0->11.0 in both x and y
    target.x = static_cast<double>(rand()) / static_cast<double>(RAND_MAX) * 11;
    target.y = static_cast<double>(rand()) / static_cast<double>(RAND_MAX) * 11;
    res.location = Vector3Ctor(target.x, target.y, -1);

    ROS_INFO("Set target to %f %f", res.location.x, res.location.y);
    return true;

}


double euclideanDistance(geometry_msgs::Vector3 & point,
        turtlesim::Pose & pose){
    return sqrt(
        pow(point.x - pose.x, 2) +
        pow(point.y - pose.y, 2)
    );
}

turtlesim::Pose turtlePose;

void setTurtlePose(const turtlesim::Pose & msg){
    turtlePose = msg;
}

void updateTurtlePose(){
    ros::NodeHandle nodeHandle;
    ros::Subscriber turtlePoseSubscriber =
        nodeHandle.subscribe("turtle1/pose", 50, setTurtlePose);
    ros::spinOnce();
}

bool distanceToTargetFunction(week3::distance_to_target::Request & req,
    week3::distance_to_target::Response & res){
    res.relative_distance = euclideanDistance(target, turtlePose);
    res.robot_location = Vector3Ctor(turtlePose.x, turtlePose.y, -1);
    res.target_location = target;
    return true;
}


int main(int argc, char **argv){
    srand(time(NULL));
    ros::init(argc, argv, "services_server");
    ros::NodeHandle nodeHandle;

    ros::ServiceServer setTargetService =
        nodeHandle.advertiseService("set_target", setTargetFunction);
    ros::ServiceServer setDistanceToTargetService =
        nodeHandle.advertiseService("distance_to_target", distanceToTargetFunction);
    ros::Subscriber turtlePoseSubscriber =
        nodeHandle.subscribe("turtle1/pose", 50, setTurtlePose);
    ROS_INFO("Ready to manage erequests");

    ros::spin();

    return 0;
}
```

Figure 2: The rosservice calls