CS 69.13, Fall 2022, Elliot Potter

Method Description

This system is decomposed into six core Python files:

initialpose_broadcaster

For each robot, this broadcasts the robot's initial pose, so that tf_broadcaster can sniff it.

tf_broadcaster

This takes those poses and publishes transforms to them from the world frame (validated in rviz)

waypoint_broadcaster

This publishes the waypoints that the robot needs to visit

simple_motion

I wrote most of this last fall. It basically just has code to rotate to and drive to a point. (I didn't want to re-write this, because I already did all the debugging work last year)

robot_follower

The follower has two main behaviors:

```
1. It listens to requests for followers, and replies on that respective service with its own
2. It instantiates an action server, and drives to the specified point if it is requested to
```

robot_leader

The leader does the following:

```
1. It listens to the waypoints channel, and sets waypoints that robots should visit
2. It requests that robots sign up using its service -- it publishes this service's name on 
3. It listens to the service, adding robot names to a list of followers
4. When it has both waypoints and followers, and waypoints==followers+1, it allocates waypoi
    Essentially, for each waypoint, it finds the closest follower (or itself).
    This is a very greedy algorithm.
5. It then tells each follower (via an action client) to drive to a selected location, and i
```

Evaluation

This system worked pretty well. The final robot positions were a little off (less than 1m), but I think this was just an issue with accuracy in my driving code. If I used a PID or some other system with feedback, this would have worked better. I had some difficulty with importing all my messages, actions and services, but fortunately I figured everything out (with your help :))