Patrol Robot Research

Project Structure

In order to get the patrol simulation going, you need to type:

```
roslaunch turtlebot_sim multi_patrol.launch
```

What this does is launch 8 `proj3_randGoal_patrol.launch` robots (disused below), Turtlebot_multi.rviz, the topology__patrol__generator from the go2goal package, the position rebroadcaster (so the turtlebots know the location of each other), as well as a network emulator node (figure out what this does).

`proj3_randGoal_patrol`

This file contains the launching information to bring up `proj3_patrol.launch` (discussed below). It also loads up the random goal generator.

proj3__patrol

This file is (finally) the one that brings up the turtlebot. This is what loads Rviz, creates the vehicle, and the go to goal node.

Nodes of Concern

Go to Goal Control

- `controllers/patrol_g2g/`
  - Sets up the Finite State Machines (pose and no pose?)
    - Rotate
    - Go to Goal
    - Wait for vehicle
    - Converge to Goal
    - Stop
  - Publishing state -> why?
- `go2goal/topology_graph/`
- `go2goal/rand_goal_generator`
- `network_topology_emulator/delta_disk_emulator`
- `turtlebot_sim/simple_map_tf`

Messages of Concern

- mv__msgs/VehiclePose.msg
- mv__msgs/VehiclePoses.msg

misc

- Eigen Class
  - When using « (bitwise left bitshift), you are just moving x bits (x being the size of the data type) and inserting the variable specified in the next index of the matrix.