Assignment #3

This assignment can be completed as a team.
Total score: 100
Due date: April 4

Objectives: Implement a heuristic algorithm to solve a robot navigation problem.

Scenario

turtle1 should start moving from the default location and within the legal boundary of the turtlesim window. The legal boundary of the turtlesim window is defined by two diagonal corners, (0, 0) and (11, 11). In addition to turtle1, there will be two additional types of turtles: **T turtles** named "T0", "T1", …"Tn" and **X turtles** called "X0", "X1", …, "Xm" where values of n and m will be determined when the T and X turtles are created. Unlike turtle1, both T and X turtles are stationary (not moving) and will land on random locations before your program starts. T turtles are not hostile to turtle1. turtle1 can capture any T turtle. X turtles however are hostile to turtle1, which means that any X turtle will capture turtle1 whenever it is "**near enough**", being **within a Euclidean distance of 0.5** (or <= 0.5). The Euclidean distance between two points (x1, y1) and (x1, y2) is calculated by $\sqrt{(x1-x2)^2+(y1-y2)^2}$. Likewise, turtle1 can only capture a T turtle by approaching within the threshold distance 0.5. To make it easy, T and X turtles will land on random locations that are far enough from turtle1 so that turtle1 will not be captured immediately as soon as it is started. You can further assume that T and X turtles will never land on the same location as one other.

Mission

The mission of turtle1 is to **capture all the T turtles**, traveling the **minimum distance** **without** being **captured** by any **X turtle** and **without** moving **out of** the **legal boundary** of the turtlesim window.

Tip: You can kill a turtle using rosservice command, "rosservice call /kill turtle-name" or using a program (as shown in "spawn_turtle.cpp" or the test program that will be available later). You may also refer to the source code of the test program posted on the course page.

The **mission** of turtle1 **fails** upon being captured by any X turtle **or** moving beyond the legal boundary of the window.

As soon as the mission is completed, **display** the names of **all the T turtles**, the total Euclidean distance traveled by the turtle1, average velocity, total time taken, and total distance traveled that is calculated by ∑velocity*Δtime on the screen, and **stop** turtle1 at current position.

Required Activities

(1) **Write a program** called "**hw3.cpp**" that **heuristically** **controls** the navigation of the ROS simulated turtle robot, "**turtle1**", to accomplish the mission. The T and X turtles will be created by the test program called "**hw3test.cpp**" that will be given to you. DO NOT code for creating T and X turtles in your program hw3.cpp.

Note: Make sure you comment the major functions or modules of your program.

(2) **Test your program by following steps: (a)** Download hw3test.cpp from the course page and put it under your package directory where your program hw3.cpp is located. **(b)** Build your package. **(c)** Open a terminal and start the turtlesim window using the command "rosrun turtlesim turtlesim_node". **(d)** Open another terminal and run the test program, hw3test.cpp using the command "rosrun hw3 hw3test" assuming the package name is "hw3". "hw3test" program if properly started, will spawn all the T and X turtles and capture turtle1 when it is near the threshold distance. **(e)** Run your own program **hw3.cpp** that will only navigate turtle1 to accomplish the mission.

Note: Again, your program should NOT include any code to create T or X turtles. Instead, focus only on controlling turtle1 to accomplish the mission. If necessary, you can change the number of T or X turtles directly modifying the values of variables MAX_TTURTLES or MAX_XTURTLES in hw3test.cpp.

(3) **Create a README file** including the detailed steps needed to build and test your program. If I am not able to build your program, you may receive a score that only considers the "effort" which is possibly the lowest score, e.g. 10 ~ 20%.

(4) **Write a brief report** in Word format including **(a)** Your team name, member name(s), contact email addresses, and also the percentage contribution to this assignment if the assignment was completed by a team (if a team cannot reach a consensus on the individual contribution, include the individual's claimed percent contribution with a brief description on specific tasks performed); **(b)** The **name** of the **search algorithm** implemented; **(c)** A **brief description** about your **heuristics** to accomplish the **mission**; **(d)** A **pseudo code** (NOT source code) for your **heuristics**. The source code should be turned in as a separate file.

Warning: Although code reuse from existing source codes on the Internet is allowed, copying code from another student or team belonging to this class is strictly prohibited. Any student or team violating this policy will receive a **ZERO** score for this assignment.

How to submit this assignment

Include (a) your report, (b) **ONLY the text file(s)** you **created/modified** in **ONE zipped or compressed file** by **your (or team's) name** and submit it to **Titanium**. For example, if your team name is "ABC", then the zip file name should be **ABC.zip**. If the assignment was completed by a team, only **ONE** of **your team members** needs to submit your team's work.

DO NOT include any other ROS files you didn't change/add. DO NOT include the entire folder of your ROS workspace.

Grading policy

Your work will be graded based on the quality of your (or team's) work, considering the completion of the requirements and the written report.