Assignment 1

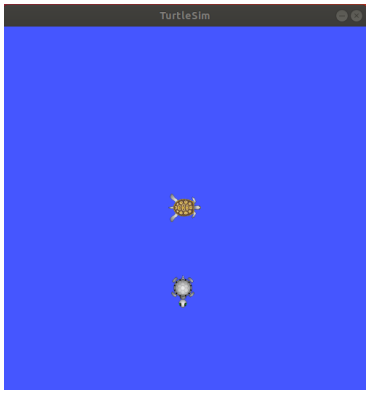
Due: Sun 29th March 2020 at 23:59 (IST – Irish Standard Time)

Introduction

In this assignment, you will mimic a "factory floor" situation using the turtlesim simulator. The goal is to implement the following functionality:

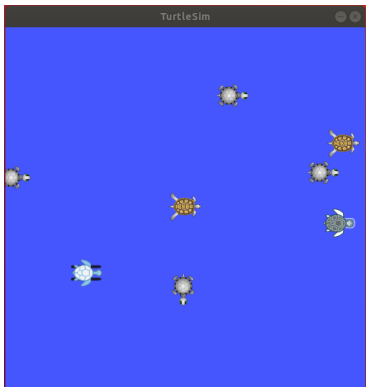
- A mobile robot that collects objects scattered throughout the workspace.
- A fixed-trajectory robot that collects objects delivered via a conveyor belt.

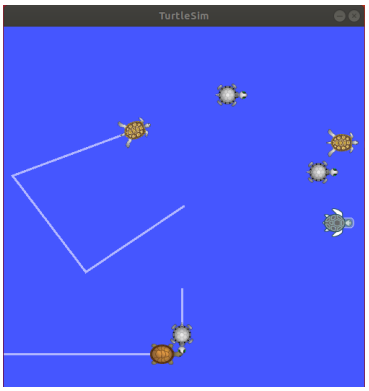
The initial state of your turtlesim simulation should look as follows:



This is your workspace before any objects are spawned, i.e. only "robots" present. The turtle in the centre of the workspace (turtle1) represents the mobile robot that can move freely about the workspace. The "south-facing" turtle (turtle2) represents the fixed-trajectory robot that will collect objects from the "conveyor belt".

The following illustrates the workspace after the objects to be collected (also represented as turtles) have been spawned, and a snapshot of the simulation in progress:





When determining the trajectory of turtle1, you may adopt a "nearest neighbour" (greedy algorithm) approach, i.e. when determining the next object to be collected, simply choose the closest one. If you wish to instead solve for the shortest overall route*, you may do so. This is not a requirement, however, and no additional marks will be awarded for this.

In addition, note the following:

- Objects (i.e. turtles) should be removed from the simulation once collected (but only once successfully collected).
- The mobile robot (turtle1) should return to its starting point once all objects have been collected.
- The fixed-trajectory robot (turtle2) should return to its starting point after collecting each object from the "conveyor belt".
- Aim to have turtle2 arrive at the collection point at the same time as the object to be collected, i.e. turtle2 should not spend an unnecessary amount of time at the "conveyor belt" waiting for the object to arrive.
- You do not need to change the orientation of turtle2. It should simply move directly forwards and backwards.

Create a ROS package for this assignment. Once completed, archive† your entire package directory and upload to Canvas.

Tasks

1. Create a ROS service server that will spawn a given number of turtles at random coordinates throughout the workspace. The number of turtles to be spawned should be passed to the server as the message request. The server response will depend on how you choose to implement the remainder of the assignment. [10%]

2. Create a ROS service server that determines the next object to be collected by turtle1. Alternatively, determine the entire route in advance by solving the travelling salesperson problem. Note: turtle2 and "conveyor belt" objects should not be included in the calculations here. [15%]

3. Create a ROS action server that moves turtle1 in a straight line to a specified location. The action server should provide the following feedback during execution:
   - An estimate of the remaining journey time.
   - A progress bar (in the form of a string) indicating the percentage of the journey completed, e.g. if the turtle has travelled 60% of the distance to the specified location, the progress bar should be `|======....|`

   [25%]

*This is a classic combinatorial optimisation problem known as the travelling salesperson problem. If you choose to solve for the shortest route, do not do so by brute force.

†You can archive in any format you choose, e.g. zip, tar, etc.

4. Spawn new "conveyor belt" turtles at random intervals (but spaced at least five seconds apart). These turtles should be spawned at coordinates (0, 1) and should move in a horizontal path at a speed of 1 m/s. [10%]

5. Ensure that "conveyor belt" turtles are collected by turtle2 when they reach the collection point at coordinates (5.5, 1). [10%]

6. Create a launch file that will start your simulation. When executed, it should:
   - Start the turtlesim simulator. [2%]
   - Spawn turtle2 (the "fixed-trajectory robot") at coordinates (5.5, 3) and orientation $-\pi/2$. [2%]
   - Spawn a random number of additional turtles (between 5 and 10), randomly distributed over the workspace. These are the objects to be collected by turtle1 (the "mobile robot"). [2%]
   - Activate turtle1, i.e. set it on a trajectory to collect all required objects and return to its starting point (providing progress reports on its journey between objects as outlined above). [7%]
   - Spawn "conveyor belt" turtles at random intervals and set them on a horizontal trajectory ("along the conveyor belt") as outlined above. [2%]
   - Ensure that objects (turtles) are removed from the simulation once collected. Note: you should verify in your code that the object has been successfully collected before removing it from the simulation, i.e. verify that it has been reached by the relevant "robot" (turtle1 or turtle2). [15%]