Design Challenge 2: The Great Escape

1 The Challenge

Oh no! The robots have been trapped in a pen and need to escape. We already know how to move forward, but have no way of avoiding walls. This is easy to fix, however, by writing the behavior avoidWalls(). Thanks to statistics, if we continually move forward and avoid walls our robot will eventually escape the pen.

2 Behavior Control

You will need to use three functions sucessfully escape the pen. You wrote behControl.goalStop() and behControl.moveForward() earlier today. You will also need to write the function behControl.wallAvoid(). Remember to think about the highest to lowest hierarchy presented in todays activity.

3 Avoid Walls

3.1 Overview

Each robot has 8 IR transmitters and 8 IR receivers. The transmitters send out messages which, if close enough to a wall, will bounce back to the receivers. We can use these reflected signals to determine if the robot is near a wall and which way to turn. Thankfully, all of the messy IR communication and direction calculation is taken care of by the following functions:

1. behControl.ObsFront(): Determines if there is an obstacle in front of the robot. Input: None, Output: True if an obstacle is in front of the robot. False if otherwise.

2. behControl.ObsRight(): Determines if there is an obstacle to the right of the robot. Input: None, Output: True if an obstacle is to the right of the robot. False if otherwise.

3. behControl.ObsLeft(): Determines if there is an obstacle to the left of the robot. Input: None, Output: True if an obstacle is to the left of the robot. False if otherwise.

We are writing the function avoidWalls(). The function will tale no arguments and return tv, rv, and active. You will need to employ if, elif, and else statements to make decisions for which direction your robot needs to turn. You will get information about obstacles from behControl.dectObsLeft, behControl.dectObsRight,
and `behControl.detectObsFront` functions which you use as arguments in your if statements. Because your inputs will be either True or False you need to check for one of these in your if statements. Within each if statement you will need to use your `turnLeft` and `turnRight` functions from yesterday to guide your robot away from the walls.

### 3.2 Pseudocode

Just like yesterday, we want to think about what we’re doing before we actually do it. Your pseudocode needs to outline each if statement, its argument, and what the robot should do when the if statement evaluates to true.

### 3.3 Writing `wallAvoid()`

Turn your pseudocode into code! Note: You will not be able to test wall avoid until you set up your behaviors correctly!

### 4 Scoring

+50 Robot exits the maze  
-10 Robot doesn’t stop when it sees the goal robot  
-5 Every time your robot hits a wall