CS148 Midterm: RoboTag

Due: Thursday, March 25, 2004, in class

Contents

1 Overview

Think tag. But not quite tag. Think a variation on the game of tag. Your robot will be pitted against another robot, designed and built by your peers, and this will be the test to see who can build a better bot. It will also be a good chunk of your semester grade.

RoboTag is a game of two robots. The point of the game is to score as many points as possible. The robot with the highest score at the end of the game wins.

During game play, one robot is on "offense" and one robot is on "defense". Only the offensive robot can score points. When the defensive robot tags the offensive robot, the roles switch and the defensive robot becomes the offensive robot, and visa versa. Roles will also switch if a significant amount of time has passed without the defensive robot tagging the offensive robot.

The offensive robot scores points by tagging end zones. The first end zone that the offensive robot tags does not matter, it will get points regardless of which end zone it tags. Following that, the offensive robot must tag alternating end zones in order to be awarded points.

Each robot will be equipped with two lego lights that is to remain on throughout the duration of the game. These lights consume a fairly significant amount of electricity, so watch your battery levels.

Commands will be issued to the robots via IR signals sent using LNP. Support code is provided to take care of most of this functionality for you.

At the beginning of each match, each robot will be given a calibration phase to allow the robot to do such things as calibrate its light sensor readings. During the calibration phase, each robot will be assigned an ID, either 0 or 1. Robot 0 will always start the match as being on defense.

After the calibration phase is over, the TAs will place the robots on the playing field. At this point, the robots are on their own and human intervention will be allowed only under extenuating circumstances.

2 The Game

2.1 Calibration

At the beginning of the calibration phase, you will be told whether your robot is identified as robot 0 or robot 1. If your robot is robot 0, then it will start out on defense, so you should configure it accordingly (either by pushing a button or doing some hand waving or whatever). You are only allowed to have one program on your RCX, so this program must be able to configure itself at startup.

You will be allowed to calibrate the light sensors so that your robot is able to detect which zone it is in. This should be done by placing your robot

The playing field is a $8' \times 4'$ board divided into sections as follows:

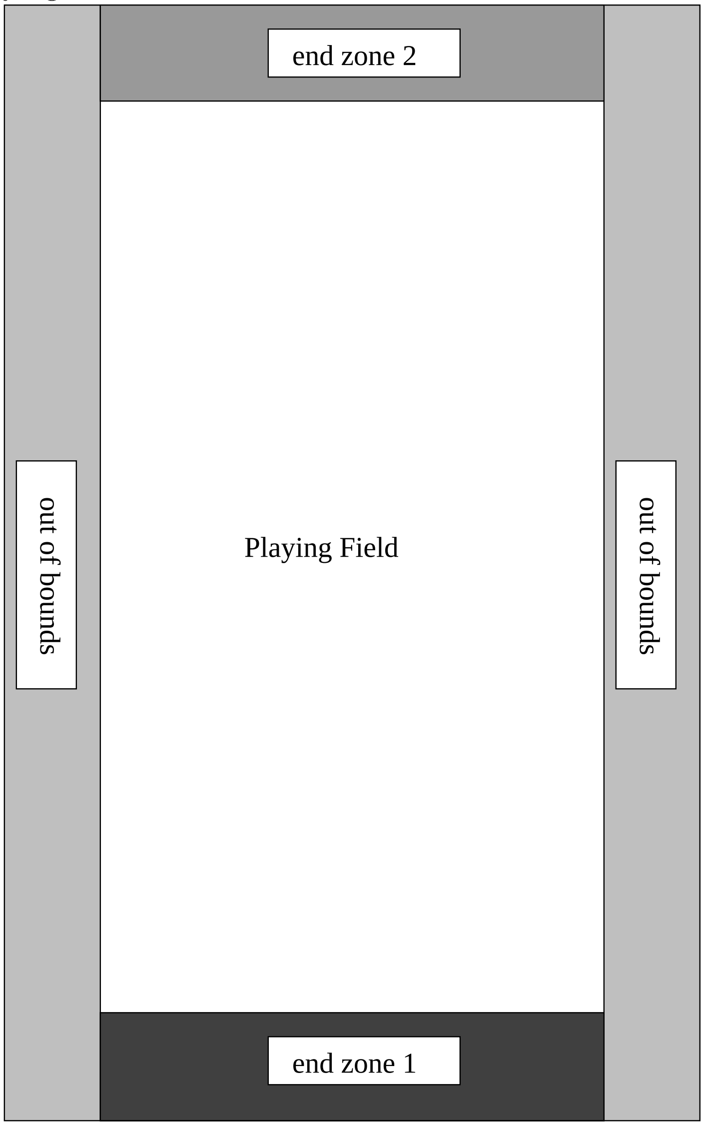

Diagram not drawn to scale.
The out of bounds zones are the same color, as are the end zones.
The board will be divided into 2 color zones down the center line (not pictured). Each side of the line will be a different color.

Figure 1: The playing board

over each zone and pressing a button or key or such.

Your robot should also take some ambient light level readings, as this will be important for determining approximately where the other robot is. You are guaranteed that the lego lights will have a higher light reading than the ambient lights.

2.2 Game

After the calibration phase they should go into the game playing mode. The first thing the robot should do is wait for the `GameStart` signal (section 3.2). Each robot will be placed in an arbitrary position and direction on the playing field. When the `GameStart` signal is received, your robot should turn its light on and then engage its game playing algorithm.

The general gameplay is that the offensive robot is trying to score at the end zones while avoiding the defensive robot. The defensive robot is trying to "tag" (bump into) the offensive robot. When the defensive robot successfully tags the offensive robot, the two robots switch roles; the offensive robot is now on defense and the defensive robot is now on offense.

A tag occurs when the defensive robot runs into the offensive robot. The defensive robot must indicate that it has tagged the other robot (e.g., by playing a short tune, which should *not* be based on the Simpsons or Les Misérables)

When a tag occurs, your robot will receive a `Tag` signal (section 3.2). This signal will also indicate which robot is on defense. (You should use this for sanity checks.)

If the defensive robot does not manage to tag the offensive robot within one minute of entering defensive mode, then a `ManualSwitch` signal will be issued. Upon receiving the `ManualSwitch` signal, the robots should switch roles.

Since the gameplay can get rather chaotic at times, there is also a `Pause` signal. When the pause signal is issued, your robot should immediately stop and maintain its current state. While paused, your robot should only respond to `ManualSwitch`, `GameOver`, and `Restart` signals. When the `Restart` signal is issued, your robot should immediately start moving again. The `Restart` signal will also carry a sanity check bit, which you are required to make use of.

Each game will last for approximately 4 minutes.

2.2.1 Offensive Robot

When a robot switches to offensive mode, it should try to score points and avoid the defensive robot.

The offensive robot scores points by entering alternating end zones. The robot will be awarded points for the first end zone it enters. There are two end zones on the playing field. The robot may not score in the same end zone twice in a row. For example, if the robot scores in end zone 1, it must score in end zone 2 next before it can score again in end zone 1.

If any part of the offensive robot enters a scoring zone while the robot is still considered in bounds, it will be counted as a score and points will be awarded.

When the offensive robot receives the `GameStart` signal, it should try to score points by finding an end zone. Upon scoring (by reaching an end zone) it should make some indication that it has scored.

The offensive robot should also be avoiding the defensive robot at the same time. How you want to avoid the defensive robot is up to you and your partner to decide.

2.2.2 Defensive Robot

When the defensive robot receives the `GameStart` signal, it should try to find the offensive robot and tag it by activating its own bump sensor. (Basically running into the offensive robot).

In order to prevent "tagback" scenarios, the defensive robot, upon switching to defensive mode, is not allowed to move for the duration of one second. This is not required when the game starts. This should happen *when the signal is received*, not when your robot detects the change.

When the defensive robot tags the offensive robot, it should receive a `Tag` signal from the TA. Upon receiving confirmation from the TAs, the robot should switch to offensive mode.

When the robots receive the IR signal (either `Tag` or `ManualSwitch`), the newly defensive robot (the robot that just got tagged) must stay still for one second to allow the other robot to run away. If no tagging occurs after a minute, the TAs will switch it up by sending the `ManualSwitch` signal. If the robot receives the `ManualSwitch` signal during a pause, it must stay still for a second after the `Restart` signal is issued.

2.2.3 Out of Bounds

The defensive robot is allowed to go out of bounds for as long as it wants. The offensive robot is not allowed to go out of bounds at all. The robot is out of bounds if the entire robot is in the out of bounds zone. Thus, it should not try to ride the edge of the playing field. When the offensive robot is determined to be out of bounds, then it will become the defensive robot. This will be indicated by a `ManualSwitch` signal. If both of the robots are out of bounds, nothing will happen.

3 Details

3.1 Rules and Regulations

- robots must never exceed $1' \times 1' \times 1'$at any point before, during, or after the game
- No hardware or software changes are allowed to any robot after the competition begins. *Only one program is allowed to be on your RCX during the competition.* (You cannot choose a program based on the opponent's algorithm.) This program may be restarted at the discretion of the judges.
- No "damaging" foreign objects may be added to the robot (i.e., *no battle bots*) TAs reserve the right to deny objects which they feel may be dangerous or inappropriate.
- Your robots should be programmed with good sportsmanship in mind. We will not look favorably on aggressive or potentially damaging strategies.
- Neither you nor your robot may emit any IR signals during any part of the game.
- If any piece of your robot falls onto the playing field during the competition or your robot becomes immobile in some way that is not a direct result of malicious behavior by your opponents' bot, your robot will receive one penalty point, and if it is on offense, it will be switched to defense. The game will be paused to allow you to repair your robot. Three such violations will result in a forfeit.

- You will not be allowed to touch the robot during the course of gameplay except in the situation described in the previous rule.
- The lego lights must remain on from the time that the `GameStart` signal is issued until the time that the `GameOver` signal is issued. It must be elevated 6–10″off the ground and there must not be any objects obstructing its visibility from any direction other than down.
- **IF YOUR ROBOT DOES NOT STOP FOR A SECOND AFTER SWITCHING TO DEFENSE, YOUR OPPONENTS WILL BE GIVEN AN AUTOMATIC VICTORY.**

3.2 IR signals

`GameStart` This signal will be issued once per match, when the game starts. Your robot must wait for this signal after finishing its calibration phase. Once it is issued, your robot can immediately start moving.

`GameOver` This signal will be issued once per match, when the game ends. Upon receiving this signal, your robot must cease movement and end its program.

`Tag` This signal indicates that the defensive robot has tagged the offensive robot, and that the two robots should switch roles. The robot that assumes the defensive role must stop and wait for one second before continuing operation. This signal will be accompanied by a field indicating which robot is on defense. You must use this field as a sanity check.

`Score` This signal indicates that the offensive robot has scored.

`ManualSwitch` This signal indicates that either the offensive robot has strayed out of bounds, or that the defensive robot has failed to tag the offensive robot. In either case, the robots are to switch roles. This signal will accompanied by a field indicating which robot is on defense. You must use this field as a sanity check.

`Pause` This signal indicates that the game has been paused. Your robot should immediately stop moving.

`Restart` This signal indicates that the game has been unpaused. Your robot may immediately start moving *unless* a `ManualSwitch` signal was issued while the game was paused. In this case, the robot should wait for one second before moving. This signal will be accompanied by a field indicating which robot is on defense. You must use this field as a sanity check.

3.3 Support Code

Stencil files are available at `/course/cs148/asgn/robotag` you will want to look through the supplied files to get an understanding of how the support code works. We wrote the IR handling routines for you, but make sure you understand the flow of control.

3.4 Referee tool

We will be using a small program to referee the game. You will want to use this tool during your testing routines. On linux, the program is available at `/course/cs148/bin/midterm_menu`; On windows/cygwin, the program is available at `l:/winbin/midterm_menu`. You can pass in the Defense Timeout value in seconds as a command ine paraemeter.

The source code for the referee tools is available at `/course/cs148/src/midterm_menu`.

3.5 Practicing and Preparing

There will be a board in the Lego Lab that you can come and test your robot on. We may also end up putting a board in the Sun Lab (on the stage) or in the MS Lab.

3.6 Grading

In order to receive a satisfactory grade on the midterm, you are responsible for completing a number of **required features**.

- Your robot must demonstrate that it recognizes all zones and zone boundaries.
- Your robot must be able to detect and seek a moving light source with a reasonable degree of success and efficiency.

- Your robot must demonstrate that it has both a defensive and an offensive strategy that are not the same. Both of these strategies must be non-trivial (i.e., a robot that runs around randomly regardless of what's going on does not have a satisfactory strategy). The effectiveness of your strategy will count to a certain extent.

- Your robot must react to the infrared signals quickly and in the prescribed manner (i.e., start playing when `GameStart` is sent, transition to an offensive/defensive algorithm on the appropriate signals, pause and restart correctly, utilize sanity check data, wait for one second when switching to defense, etc.). You will be penalized for any deviation from the specifications in this document.

- Your robot must be able to indicate in some way that it has scored a goal. It must not erroneously indicate that it has scored a goal. Although this indication will have no bearing on the gameplay during demo day, it will be part of your grade. Similarly, your robot must also indicate when it has tagged the other robot. A robot on offense should never tag anything, so make sure that doesn't happen.

- Your robot should never get stuck out of bounds. If it wanders out of bounds, it must immediately find its way back in bounds in a timely manner.

- Your program should never crash. After receiving the `GameOver` signal, it should cleanup and exit nicely.

You must include a paper handin. The paper handin will account for 50% of your grade

Your paper handin should include a description and drawings/figures of key design elements of your robot. You should explain both your offensive and defensive algorithms, and your calibration phase. You should explain your design process and the design decisions you made. You should explain any problems and difficulties you encountered and how you overcame or worked around them. This handin should be slightly more detailed than previous project handins. A reader should be able to implement your algorithm based on the information in this writeup. (This doesn't mean it has to be longer, just efficient and detailed. Please don't leave the writeup until the morning it's due.)

Please electronically hand in your code as well **/course/cs148/bin/cs148_handin robotag**. One copy of your code will be sufficient for the group. (Handin script only works from a Linux machine.)

Note: You must handin a writeup or else you will not get credit for your robotag robot.