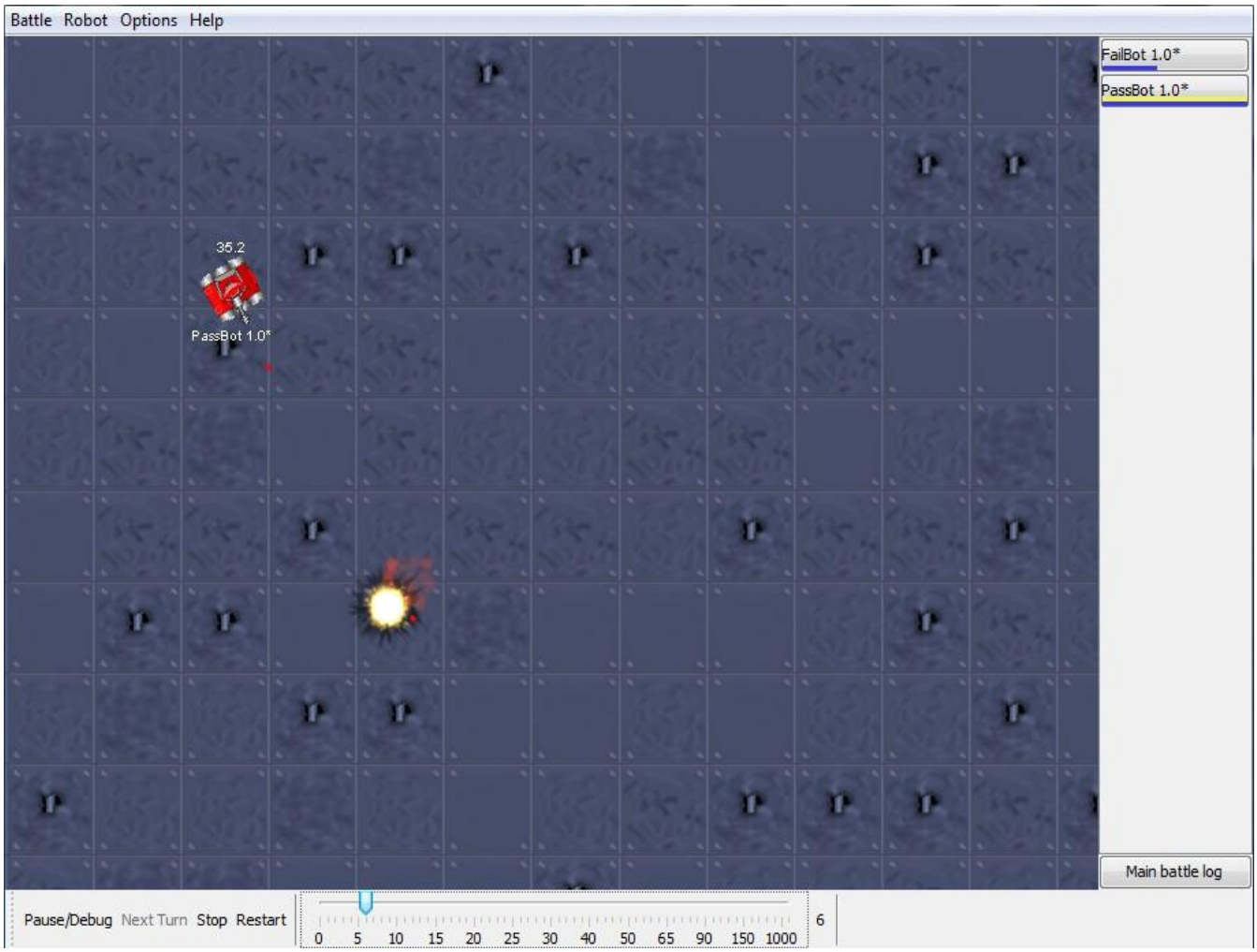
Robocode Project

Table of Contents

Introduction

The Robocode Tournament at Full Sail is a way for students to develop AI strategy for a real agent in a competitive environment. The tournament will be scored according to survival, inflicted damage, and elimination. Students should take this opportunity to apply some concepts learned in AI in the process of shaping and designing their robot for the tournament battles.



For this project, you will implement a single software robot from the ground up that functions in the Robocode environment. Robocode robots are tanks with radar and a single turret. Your goal will be to match or exceed the performance (based on score as computed by the environment) of the robot provided to you with the lab project materials, affectionately known as **FailBot**.

Getting Started

Building a Robocode robot can be a fun, but challenging, experience. A wealth of information is available online at http://robocode.sourceforge.net/. In addition, a tutorial can be found on the Robowiki at http://robowiki.net/wiki/Robocode_Basics.

Assignment Specification

Please follow these guidelines when building your robot and preparing your team. Submissions that do not follow these guidelines may be disqualified and/or graded zero.

Robot Code

All robot code submitted must meet these technical requirements:

-Robot must be in the "fullsail" package (i.e., contain the line "package fullsail;")
-All code, including that for supporting classes, must be in a single class file (use private classes)
-Robot radar and bullets **must be visible** (no transparent radar / bullets)
-Robots must compile and run to receive credit

Citation and Submission

In addition to meeting technical requirements, students building robots should be careful to follow these rules for the assignment itself:

-Student submissions should represent original work and not simply a reuse of existing ideas
-Students will outline their robot strategies in a README file of 1-2 pages (TXT or PDF)
-Students are explicitly forbidden from engaging in plagarism and decompilation of robot code
-**Students will properly cite any strategy inspired by outside resources**

Students must only use those resources defined in this document. Under no circumstances may students work together on code – Robocode is no exception to the academic honesty rules in place. If and when a student uses one of the sanctioned outside resources, proper citation of strategies developed by others is ***crucial***. Failure to properly cite the use of ideas developed and/or published by others will be considered academic dishonesty.

When a student uses one of the sanctioned resources in the development of his/her robot, the student will clearly site the source of the strategy in source code and in the README file.

UNDER NO CIRCUMSTANCES ARE STUDENTS PERMITTED TO COPY CODE FROM ANOTHER SOURCE. All implementations of code must be done "clean room" style – students may read about techniques but are prohibited from examining resources which clearly display code while writing the source code for their robots.

In addition, robots may only use data generated by the student or derived from references. While students may use training data from outside sources to train their robots, they may not use any data or script which defines the fundamental architecture of the agent's decision making process.

Tournament Rules

The tournament will adhere to the following rules:

-Robots submitted will be provided (compiled) to future classes.
-Students may submit at most one robot as an entrant into the tournament.
-All robots submitted for a grade must be submitted as entrants to the tournament.

Robots could be disqualified if they:

·Do not start due to bugs in their codes, or cause the Robocode environment to crash
·Use neural network weights / scripts / code generated or provided by a third party

However, an acceptable robot might still:

·Use training sets generated by a third party to train a neural network
·Dictate behavior based on a script rather than hard-coded instructions

Team Requirements

Each student must include exactly one robot on a team with other students. The follow rules apply:

-All teams must be 2-4 robots in size
-All teams must choose a team name and team color
-All robots in a team must use the same radar color and bullet color

Students will work together to prepare their team strategy. ***Students may not share any code to implement such a strategy***, though they may develop a communication strategy together which must be implemented individually.

Building a Team Tournament Package

The following are steps to be used in building a team package for testing teams.

1. Place all robot source (java) files in the "robots/fullsail" folder within the Robocode project.
2. Make sure all robots have "package fullsail;" as the first line in the source.
3. Clean and rebuild the project using Eclipse. DO NOT SKIP THIS STEP.
4. Check that each robot functions in the Robocode environment.
5. Create a team with the robots (Robot → Create a robot team.)
6. Create a team package (Robot → Package robot for upload.)

NOTE: Be sure the checkbox marked "Include source" is checked!

Scoring & Grading

This assignment consists of two parts: **robot performance** and **the tournament**.

Robot Performance

Each robot's performance will be measured against the robot provided as part of the assignment (FailBot and PassBot) and will be graded as follows:

-Student robots will be posed one-on-one again FailBot for 100 rounds (600x600 battlefield)
-Robots must score as well or better than FailBot in order to be graded 100%
-Robots failing to score as well as FailBot will be graded less than 100% (proportionally)
-Extra credit is awarded according to the following ranges for student final scores:
    -Student robot score is 55%-64% = 1% extra credit
    -Student robot score is 65%-74% = 2% extra credit
    -Student robot score is 75%-84% = 3% extra credit
    -Student robot score is 85%-94% = 4% extra credit
    -Student robot score is 95%+ = 5% extra credit

Tournament

The tournament battles consist of multiple brackets and a specific ranking system.

Brackets

All robot entrants will participate in all tournament brackets. The brackets are as follows:

One-On-One

Number of Competitors:	2
Advancing Robots:	1
Rounds of Battle:	3
Battle Field Size:	600x600

Team Rumble

Number of Teams:	Varies
Advancing Teams:	Varies
Rounds of Battle:	5
Battle Field Size:	1200x1200

Ranking

Robots will be ranked according to score based on the competition type and battle type.

Battle Ranking

Each round, robots will be awarded points according to survival, damage, and elimination of opponents. Robots / teams scoring 1st in the One-on-One competition and 1st or 2nd in the Rumble

or Team Rumble competitions will advance to the next round.

Championship Ranking

After the championship battles are complete, robots will be ranked and score points according to place. Championship scores will be summed and the robot with the highest overall score will be the winner. The three robots ranked next will be the runners up. ***For Team Rumble, team members will split the winning point pool evenly.***

Scoring:

One-On-One		Team Rumble	
1st Place:	1000 points	1st Place:	1440 points
2nd Place:	750 points	2nd Place:	1080 points
3rd Place:	500 points	3rd Place:	720 points
4th Place:	250 points	4th Place:	360 points

Ties

In the event of a tie for a single battle, sudden death rounds will be added until clear winners emerge. In the event of a tie for the win, the tied competitors will participate in three rounds of battle on a 1200x1200 battlefield, with additional sudden death rounds as necessary to determine clear ranking.

Hall of Fame

The tournament winner and three runners up will be placed in a battle with current Hall of Fame robots. All robots will compete together in a Free-For-All battle on a 1200x1200 battlefield for 1000 rounds. The top ten robots will be placed in the Hall of Fame according to rank. Ties will be broken using sudden death rounds (on 1200x1200) with all competitors until a clear order emerges.

Code Conventions

The grader of the lab will enforce some or all of these conventions by penalizing any violations.

- Do not change the `public`, `protected`, or `friend` interfaces of an existing or derived class.
- Helper methods or added variables must be in `private` scope (not `protected`).

Submission Procedure

You will upload a compressed (Zipped) file named <*lastName*>.<*firstName*>.*RobocodeLab.zip* which should include:

<robot name>.java

Also, please submit any source/header files you have created yourself.

Place this file in the ROOT of your zip file, not in a folder or subfolder. DO NOT SUBMIT EXECUTABLES, LIBRARIES, OR OBJECT FILES.