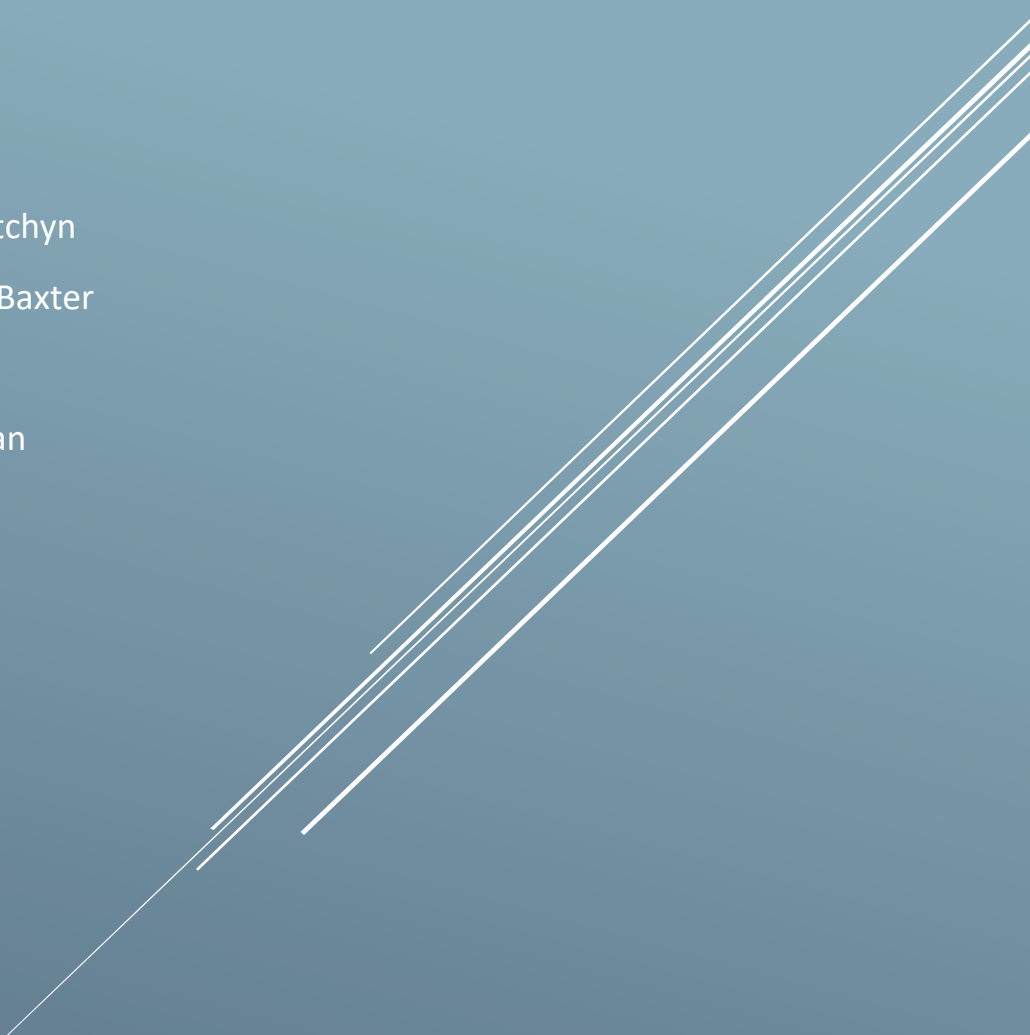


Professor Christopher Dutchyn
Tutorial Leader Jonathan Baxter
CMPT 370, Term 1
University of Saskatchewan



DEPLOYMENT DOCUMENT

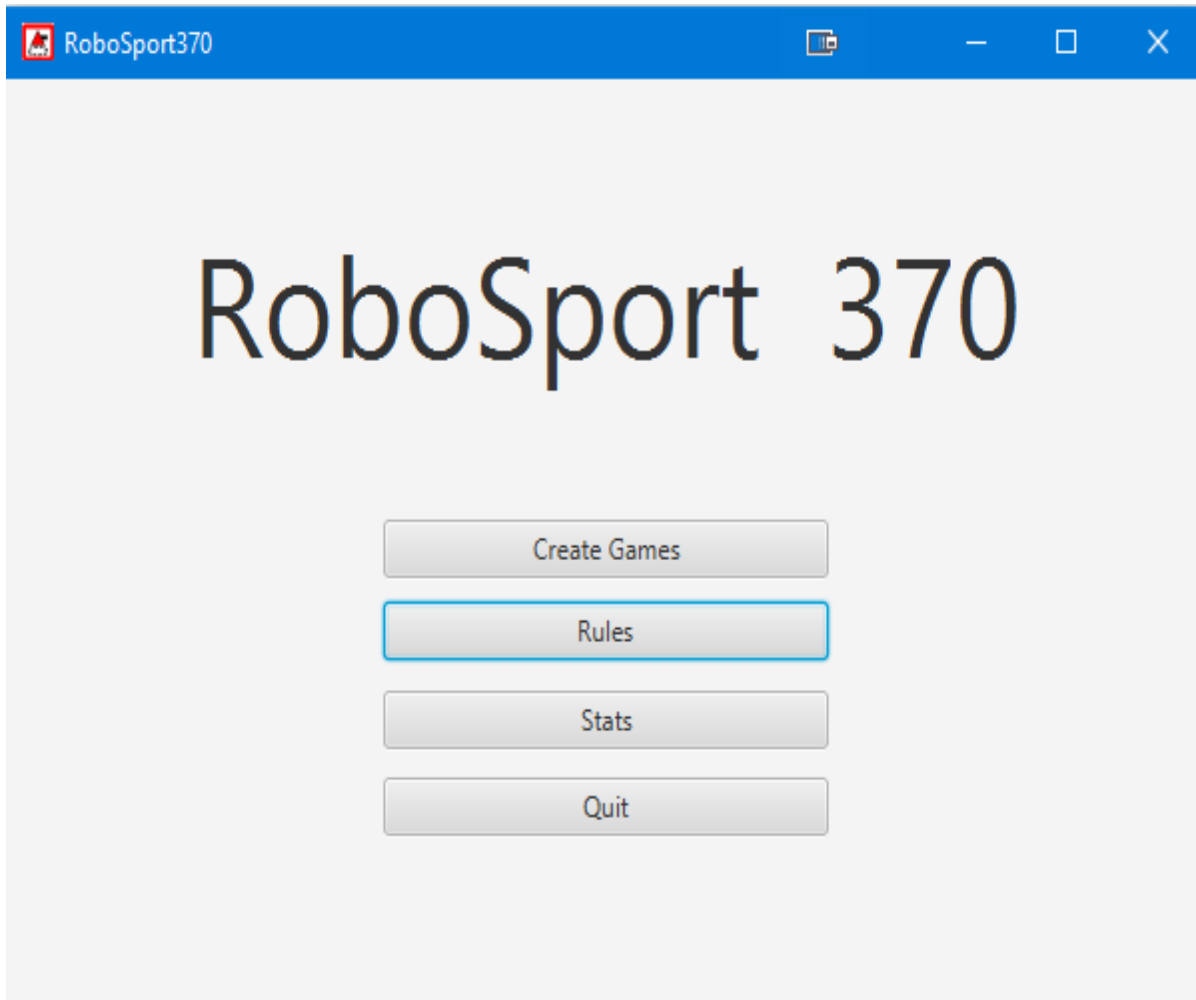
Group B1: Adam Ronellenfitsch, Dylan Prefontaine, Evan Snook,
Matthew Frisky, and Wynston Ramsay

Contents

Manual:	2
Main Menu:.....	2
Create Game:	3
In-Game:	4
Post-Game:.....	5
Rules:.....	6
Stats:	7
Quit:	7
Maintenance:	7
As-Built Architecture:.....	7
Details of intricate bits of the system	8
External libraries being relied on	8
How to compile and run:	8
Changes:.....	9
Changes since Requirements Document:	9
Changes since Design Document:	9
Bugs:.....	9

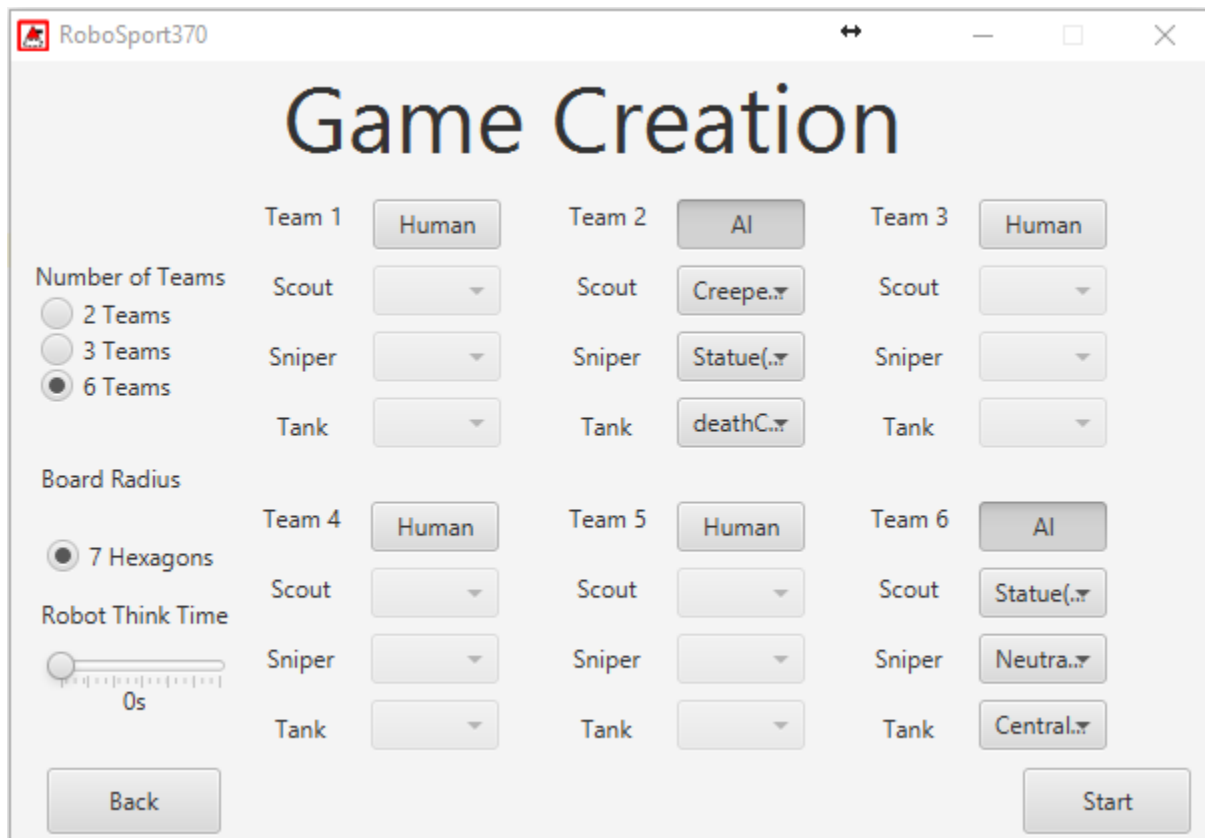
Manual:

Main Menu:



The Main Menu is the first screen that appears when you start RoboSport370. There are four choices which you can make from this screen: Creating a new game, reading the rules, viewing statistics of robots stored online, or quitting the program. There is always a way to get back to the Main Menu screen from any other game screen since it is the root for all program flow.

Create Game:

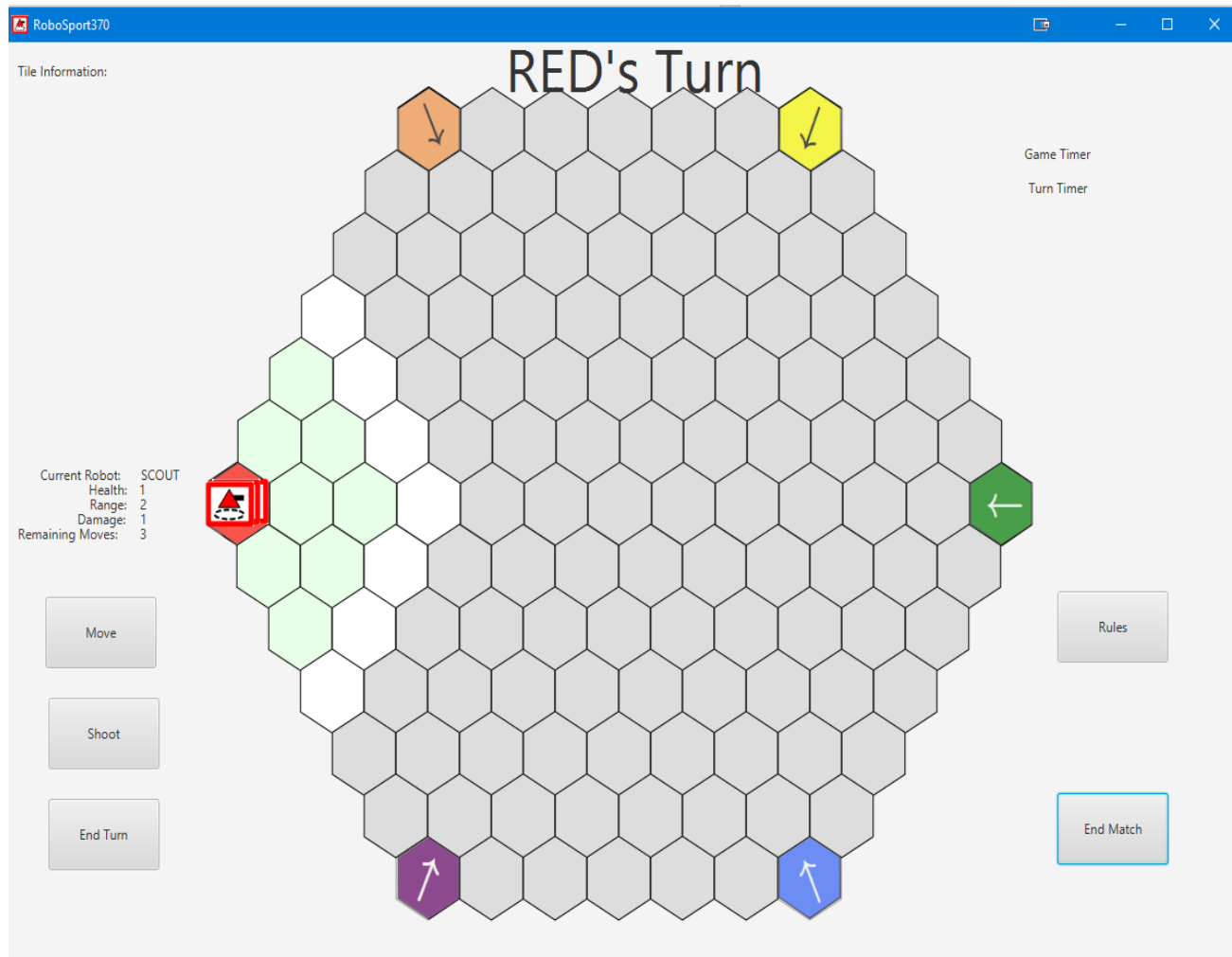


The image shows a screenshot of the 'RoboSport370' Game Creation window. The window has a title bar with the application name and standard window controls. The main title 'Game Creation' is centered at the top. On the left side, there are three sections: 'Number of Teams' with radio buttons for 2 Teams, 3 Teams, and 6 Teams (selected); 'Board Radius' with radio buttons for 7 Hexagons (selected) and 5 Hexagons; and 'Robot Think Time' with a slider set to 0s. The main area is divided into six columns for Team 1 through Team 6. Each team has a button to select its type (Human or AI) and three drop-down menus for Scout, Sniper, and Tank units. Team 1 is Human, Team 2 is AI with units Creeper, Statue, and deathC, Team 3 is Human, Team 4 is Human, Team 5 is Human, and Team 6 is AI with units Statue, Neutra, and Central. At the bottom left is a 'Back' button and at the bottom right is a 'Start' button.

Team	Type	Scout	Sniper	Tank
Team 1	Human			
Team 2	AI	Creeper	Statue	deathC
Team 3	Human			
Team 4	Human			
Team 5	Human			
Team 6	AI	Statue	Neutra	Central

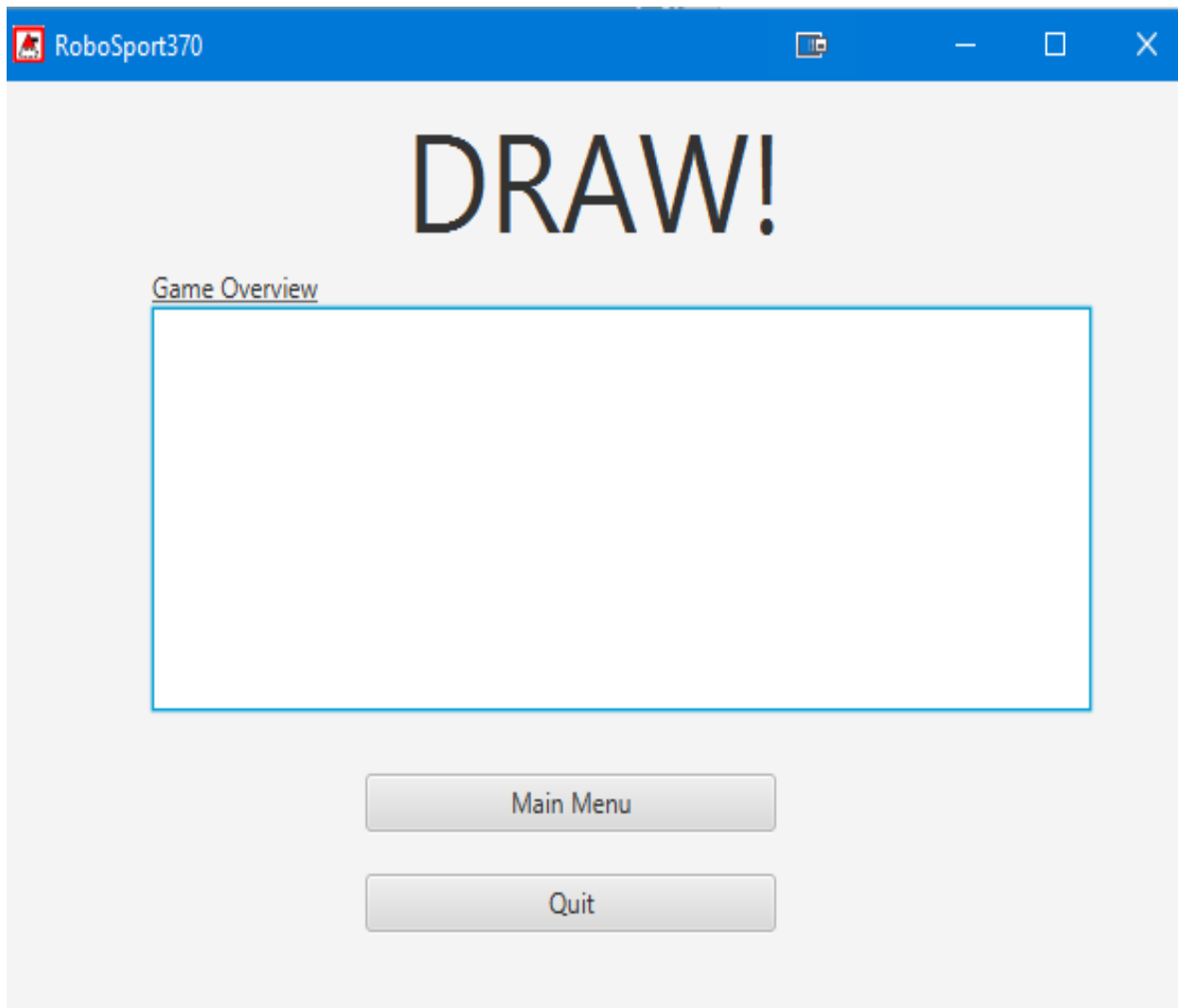
When the Create Game screen is brought up from the Main Menu, it allows you to manage settings before playing a new game. One of the first options you will see in the top left corner is labeled "Number of Teams". This allows you to select how many teams will be in the game, either 2,3, or 6 depending on which radio button you select (6 is default). Below that is the "Board Radius", that determines the size of the board that will be played. The program will auto select a size of 5 hexagons if there are two teams, or 7 hexagons if there are 6 teams. You have the choice of size 5 or 7 only when the number of teams selected is 3. Next is the "Robot Think Time" slider which appears below the "Board Radius". This effects how long the computer controlled teams will take to make their turns (in seconds up to 30). The rest of the screen is covered by "Teams" which can be changed from Human to AI controlled by clicking the button beside the team number. For AI teams, each robot needs to be linked to a specific robot stored on the Moodle server. This is done by using the drop-down menus for each robot type on each AI team. There is a "Back" button to return to the main menu, and a "Start" button to create the game and enter the in-game screen.

In-Game:



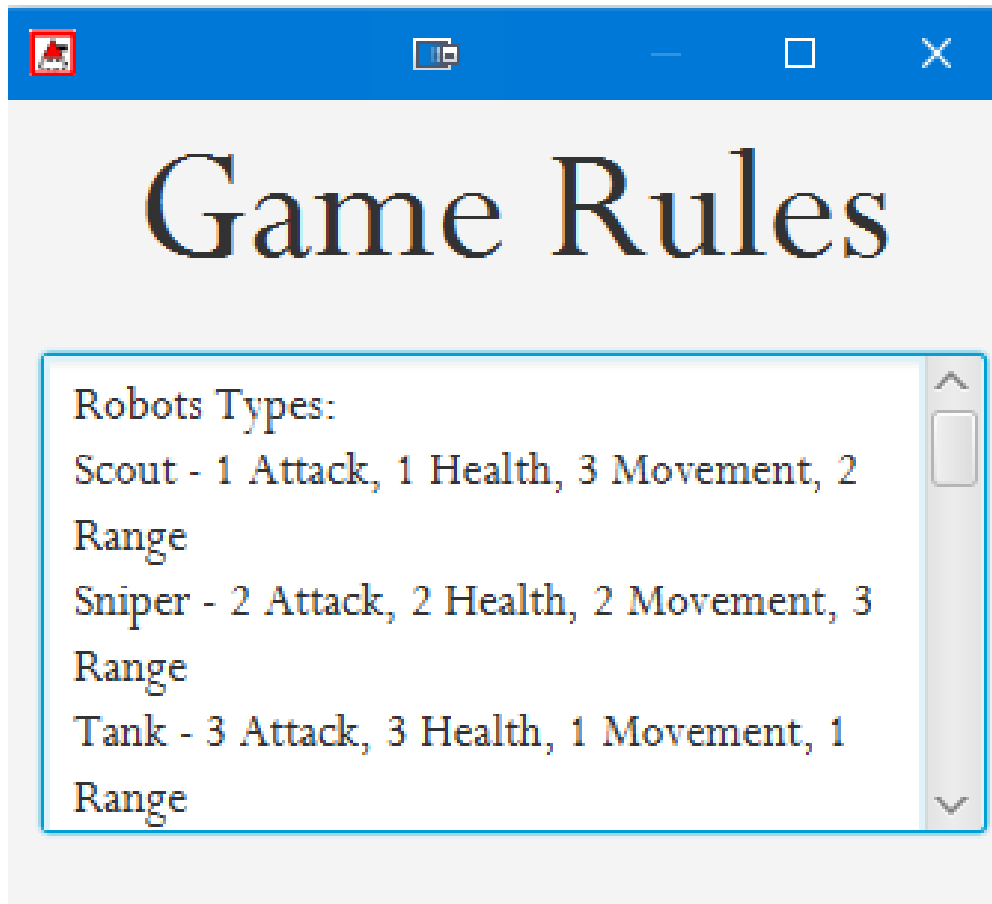
It is recommended to read the "Rules" before playing a game, however there is enough in-game directions given on who's turn it is and what robot is making its move. The top of the screen shows text on what team's turn it is, there is info on the left side of the screen on which robot is making a play, and its current statistics such as: health, range, damage, and remaining moves for the turn. When it is your turn, you can move a robot by selecting a tile adjacent to the robot and pressing the "Move" button. It is worth noting that whenever the "Move" button is pressed, the current robot will move in the direction it is FACING; the direction its facing is changed by selecting tiles around it. To shoot with the current robot, simply select a tile and press the "Shoot" button. Once you are done with a robot's turn, press the "End Turn" button. If the "End Turn" button is not pressed after 30 seconds of starting a turn, the game will move on to the next team's turn. There is a "Rules" button that displayed the rules window in case you forget something. The game is played until one team wins (see Rules), or until the "End Match" button is selected. Either situation will direct you to the post-game menu.

Post-Game:



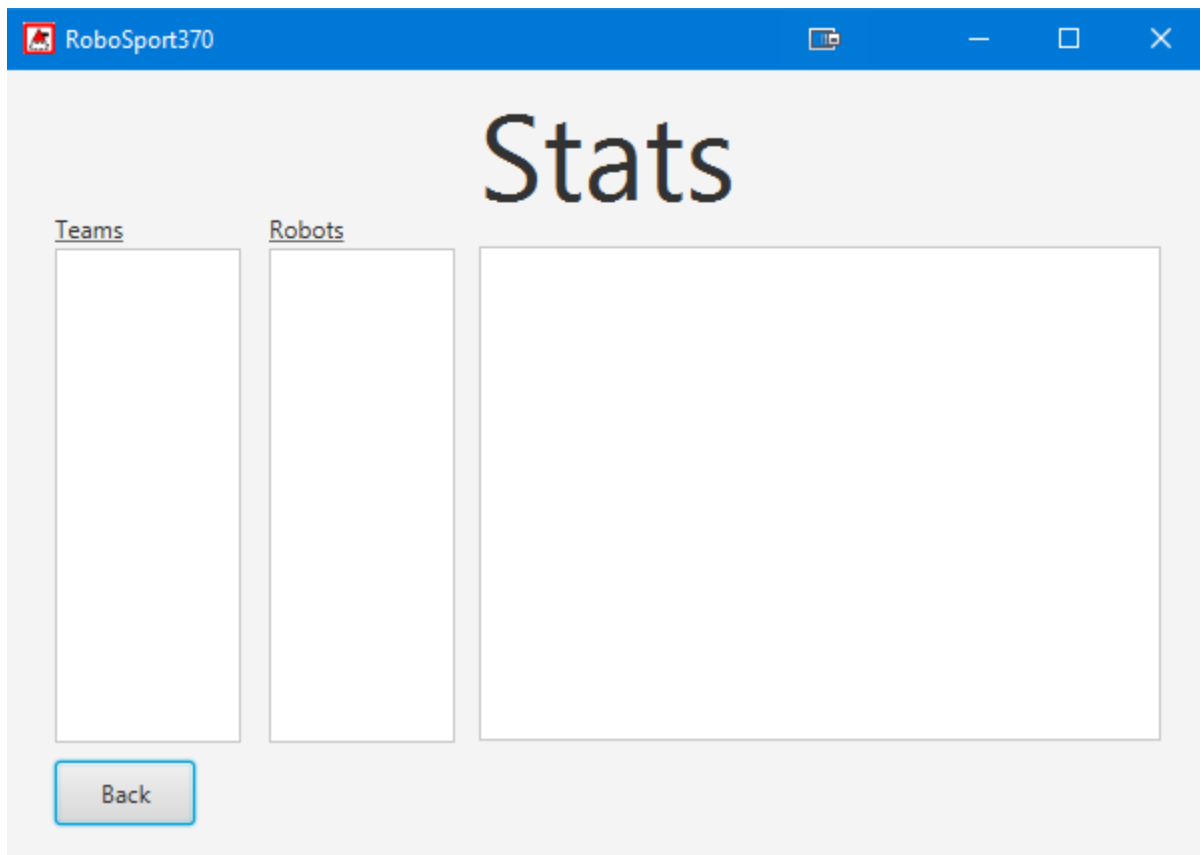
The Post-Game screen is brought up when a game is ended from a team winning, or the user selects the "End Match" button. The top of the screen will show text of the winning team if there is one, or "draw" if there isn't. The game overview section shows statistics of the game and automatically uploads these stats to the Moodle server. There are two buttons: "Main Menu" and "Quit", which you must choose from. Selecting "Main Menu" returns you to the Main Menu screen, and "Quit" will end the current RoboSport370 session by closing all windows associated with it.

Rules:



When the Rules screen is opened from the Main Menu or In-Game Screen, a separate window is brought up to read the game rules. This is useful because one can be in a current because you can read the rules while keeping the game screen open. There are no actions to be performed other than scrolling through the text to read all the rules, and closing the window. There is no "Back" button since it is a separate window from the rest of the program and clicking the rules button again will just flash this window at the user if it's already been instantiated.

Stats:



When the statistics screen is activated via the Main Menu, it will display all the robots stored on the Moodle Server. There are querying options to find specific robots which can be done by selecting specific teams from the "Teams" column, or by selecting specific robots from the "Robots" column. To select a robot or team, simply left-click the robot or team to query stats for that specific instance. Left-Clicking the "Back" button will return to the Main Menu.

Quit:

When the 'Quit' button is selected from: The Main Menu, In-Game screen, or Post-Game screen, RobotSport370 will close and end the current session.

Maintenance:

As-Built Architecture:

Our system implements a model-view-controller architecture as described in our design document, as we wanted the interface to get data from model, and we wanted the controller to take input from view and to use the input to modify the model and the view.

The view of our system is contained in resources/view and are all in fxml format. Assets are also held within the resources folder in resources/images and holds all the images for our robots, health, and start tiles to be used in the views. All others images are generated with JavaFX and the default JDK library. Views can get information from the model and are only accessed and modified by controllers.

The controllers in our system are split into view controllers in the controllers/view folder and other controllers are just in the controller folder. View controllers are all linked to the fxml files in resources/view and they do event handling for their respective interfaces. Other interfaces include the ForthInterpreter which handles reading Forth code and creating a stack out of it as well as GameMaster to handle events on the game views and enforce the rules of RoboSport370. Lastly, we have ViewController which instantiates scenes and manages showing and hiding them. Some of the controllers have a public model in the model folder that is used to access them via the model because they are driven by events that happen within model classes, and not just user-input like the view controllers.

The model is all within the model folder of the project. The Model classes, enumerators, and interfaces all hold information and functions to manipulate and present themselves. The models get their accessors and mutator's called by controllers as well as from JSON files fetched from the university's servers

Details of intricate bits of the system

ForthInterpreter initializes words defined by the given Forth language using regular expressions that will be used to recognize strings in the JSON files for the robot AI's. ViewController is used to create the stage and scenes and to change the scenes. The publicViewController in the model folder is used by events to send a message to the view controller to change the scene. Board consists of HexNodes that are all linked to each other by their sides. The important part to note is that the radius of the board is 3 nodes more than the actual graphical representation to make scanning and iterating around a robot simpler because it won't have to scan a null node if the robot is on the board and so it is less efficient system wise, but made sense given our time restraints.

External libraries being relied on

JSON-simple is being used to read JSON files from the University's servers and to send the robot back in JSON form when we are done with it. Junit is being used mostly for assert in unit testing

How to compile and run:

The Start class contains main and loads the MainMenuView and sets the view information such as icon and title. From there, the controllers take over and allow the user to navigate the program. To run the tests for a class, navigate to the tests folder and run the main for the test class that you wish to run.

Changes:

Changes since Requirements Document:

Here we will highlight some of the main features that we added/removed in the current build of RoboSport370 that differs from our Requirements Document. First, we decided to not include our RoboManager feature. RoboManager was a way for the user to manage robots stored locally but as a group we decided it was a feature that was not as important as some of the others. Statistics from local games are still loaded to the server but nothing is stored locally anymore. The user can query for statistics based on robots and teams but they can no longer reset or upload local stats since they only get published, and not saved locally. We planned for health bars to be set above each robot but concluded that it would not mesh with our interface and we would rather have a simple design with full functionality. We had plans for an "Observer" view when there is only AI playing the game which would have given the user some interesting observing features; in the end, we decided to cut this to focus on functionality as it was a "luxury" feature. The user will still be able to watch AI vs AI but it doesn't have features such as fog of war filters and changing turn timers in game. Lastly, a small design that we changed is that the user can no longer create a game from the post-game screen. They must go back to the main menu then go to the create game screen to achieve this, we believe this benefits simplicity in our interface and reduces clutter (while sacrificing very little productivity).

Changes since Design Document:

Like the Requirements Document, from our Design Document we cut RoboManager and locally stored statistics in our current build of RoboSport370. Originally, we designed our Board to be drawn when a game is created, but since we only needed two separate boards it made much more sense to manually create the two boards. It is more efficient this way and required less man hours to code. GameMaster is the main function for controlling the game. Our design did not include enough detail in this method from what we initially thought. As we tried to implement it we decided it was best to revamp GameMaster and it currently includes: several setters and getter functions, HexNodeClicked, ,updateRobotBox, outputTile, selectTile, robotClicked, , initRobots, initStartTiles, linkPolygonsToHexNodes, showRules, robotMove, robotShoot, endTurn, betweenTurn, startPlay, draw, endMatch, clearAreaFog, makeFoggyOut, colourRange. After this change, our GameMaster class is a lot easier to understand and easier to modify as well.

Bugs:

Currently the only bug is an issue with our Game Timer. It does not display and update during a game and we hope to find a solution soon. Other than that, there are no other issues that we are currently aware of. Any bug that we encountered was resolved as quickly as possible.