Professor: Christopher Dutchyn

Tutorial Leader: Jonathan Baxter

Class: CMPT 370 Term 1

Project: RoboSport 370

CONSTRUCTION DOCUMENT

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

Contents

Paired Programming Sessions

Adam's Sessions:

My first paired programming session went quite well. I've worked with Evan in the past for other classes so I knew how to work effectively with him. We got a lot done, it's an effective way to code. Our session lasted about 2 hours. We decided to use JavaFX for the GUIs. In doing so we got acquainted with the fxml file format. We are happy with the decisions we made and it is an improvement to our original design.

My second session was with Wynston. We implemented the HexNodeClicked function in the GameMaster class. We thought about the function and its purposed. It quickly came to mind that we should use the HexNodeIterator to iterate through the game board data structure to solve the solution. It was really handy to have someone there to talk out loud to about the problem and to express our thoughts about how the function should be implemented. We found the in our opinion the best solution. We also found a point of failure in the function and added a case to reinforce code. Over all the session went well I found Wynston and I work really well together and will work together in the future.

Evan's Sessions:

My first pair programming session was with Adam, whom I have pair programmed with in the past. We worked on GUI and our view controller using JavaFX and fxml files. Our implementation is slightly different than our design document which made it a bit more challenging, though we decided as a group that this would be a better way to set up our interfaces. We were happy with the way that everything turned out as we got our interfaces started and got some of the initial grunt work out of the way as well as we are both more familiar with fxml files.

My second pair programming session was with Dylan working on an implementation of HexNodes and traversing them. We chose this section because we decided to try and hard code the HexNodes instead of procedurally generating them as we had planned in our design document. In the end, we got the board setup and figured out our algorithm to iterate through the board for functions like scan. We used Team view for the session as we were both at home when the issue arose and decided to figure it out earlier rather than later. The medium we used was effective as we were able to talk to each other and use our mouse to point out errors on the other member's screen as well as give control of the mouse and keyboard to the other member, so we do not feel as not doing it in person was a limiting factor.

Dylan's Sessions:

My first pair programming session was with Evan and it rotated around creating an efficient way to traverse through the tiles of the board. We found an effective way to generate the model for the board but decided that making a visual representation based off the model was very troublesome. Since we had a WYSIWYG for creating scenes we ended up hardcoding the two sizes of boards as generating it from the model would not have any benefits.

My second pair programming session was with Matthew and our goal was to get the robot's images loaded onto the game board. We ended up changing our Robot class to better suit references the images to keep the design as simple as possible. Once we had the robots linked, we noticed that a method was becoming convoluted so we spent some time splitting it into methods while also improving what they were doing.

Matthew's Sessions:

My first paired programming session was with Wynston and we implemented the code for the initTeams method in the game class. It was a seamless process where we decided what needed to be done beforehand which was to create three conditional if statements to determine the number of teams to be initialized. Once we began programming is was easy for us to lay down the code we had discussed and to discuss and solve any issues we came across through discussion.

My second pair programming session was with Dylan working on adding the images of the robots to the game board and initializing the images. We started off by deciding what we were going to do and all the changes that needed to be made in order to implement it. Along the way, we decided to write the code in an easier more logical way that became clear to us once we started the actual coding. We also made some changes to GameMaster to make it more organized, as well as added a few variables and getters and setters to other classes that we needed. It was very helpful to be able to talk to each other and realize that there was a better way to implement what we wanted.

Wynston's Sessions:

My first paired programming session was with Adam where we implemented HexNodeClicked function in the GameMaster class. Before we wrote any code, we brainstormed briefly on how the method should flow. We quickly concluded that we would need to use the HexNodeIterator to search through our Board. After that it was very easy transition into implementing the method; we both got confused with our loop condition but being able to talk it through with someone helped us understand it. After we got the method to work the way we wanted to, we added an error trap just to solidify the method.

My second paired programming session was with Matthew where we decided to implement the initTeams method in the Game class. Like my first session, Matt and I brainstormed and set out a plan for how the method should function. We both immediately agreed that we would need three conditions for the three possibilities of team combinations. Once that was set up, all that was left to do add each team to the list of teams, and enable the proper teams for each condition. This session went very smooth because the instance one of us froze, the other could clarify the other's error in understanding.

Code Review

In the planned code review meeting, we decided to evaluate one of the tougher classes to implement, which is the Forth Interpreter. Dylan wrote the class and so he was the one to walk the group through his implementation. One immediate issue was that one of the methods was several hundred lines long. We decided that this could be resolved by creating a 'factory'. Besides that, the implementation followed the group's coding style guideline (Google) very well; variables followed the naming convention, sufficient use of Javadoc comments was evident, and so on. The extremely long initWords method was costly since it does not do one simple task which is what a method should be. It makes it difficult for programmers first seeing this code to understand what the function does. Luckily, this can be easily fixed by creating a method for each task done in the method. The only functional issue found is that when two nested loops are put side-by-side, improper output is given. Other than that small issue, the Forth Interpreter class functions as we wanted it to.

Changes

In our Design, we had planned on manually coding each GUI and having them all inherit from one AbstractView class. This has been changed so that we are now using JavaFX SceneBuilder 2.0 to construct our scenes as it was much quicker and very simple to link up fxml files to the project. This change also affected how we are drawing our board as originally, we had decided to make a Draw function for our board that drew each polygon individually and then linked them to HexNodes. However, since we started using fxml it makes a lot more sense to just hardcode the boards as they don't need to be that robust because we only have 2 types of boards.

Our other major change affects how we are Iterating through the HexNodes. Originally, we had tried to devise an algorithm for scanning that would take in the robots range and scan the board. however, the boundary cases for such an algorithm proved to be more difficult that we felt necessary. Thus, to fix this issue, we made the boards each 3 polygons larger in the data structure than in the view so that our robots can only move to the edge of the view, but since there are extra nodes, we do not have to worry about the boundary cases.