CMPT 370 Construction Experience Document for BattleBots Group A5

Mackenzie Power
Haotian Ma
Ryan Tetland
Will Revell
Mitchel Kovacs

Phase4_CONSTRUCTION

Table of Contents

Paired-Programming experiences

Overall, the paired programming experiences we have had with this project have been quite positive. In this section, each member has listed their first two designated paired programing experiences. Even though only two experiences were outlined for each member, we would like to stress that we have indeed done several more paired programming sessions with each other. Writing all of our experiences would be quite time consuming, so only each person's first two sessions are recorded. We can all say with confidence that we will continue to pair program with other courses and also for the remainder of this project.

Mackenzie and Mitch

Mackenzie's Experience:

Overall I found the programming experience with Mitch to be quite beneficial. We both worked on attempting to get the functions working for robots to shoot, deal damage and receive damage to all robots in a given space. We built a 'Mock' robot team to test their damage and range on each other. Mitch showed me a method for debugging that I had not previously been taught which I will no doubt employ in the future both for this project and others that I will do in java. One of the most beneficial aspects I felt was learned with this programming experience was that it excelled at speeding up implementing process'. More specifically, if I were stuck trying to figure out why something was not working, and the problem was something relatively minor, Mitch would point it out and the problem would be resolved quickly; and the reverse was true when I took on the role as the observer. We actually finished testing and implementing the functions much sooner than expected, and for the remainder of our time we decided to work on another members' implementation to help them along. We pair-programmed for about two and a half hours in total.

Mitch's Experience:

Mackenzie Power and I paired up for my first paired programming session. For our session, we attempted to get a lot of the robot actions working. The main goal was to get the shoot and receive damage completed and tested. Mackenzie started at the keyboard, I started with navigation. As the navigator, since I wasn't the one typing, it was easier to recognize things that would cause errors. We switched an hour and a bit into the session, to get a feel of the other role. As the one typing, I felt I was able to get a lot more done than I would alone. Anytime I did something that wasn't clear, I would clarify to my partner, which helped me think it through and allowed him to provide input. In the end we got more finished than what was expected. Overall, the experience was useful and is something I will consider using in future projects.

Mackenzie and Will

Mackenzie's Experience:

This was the second time I engaged in paired programming for this project. Working with Will, we implemented the scan and move functions of the AI robots. We came up with a method that would successfully scan all spaces within range of a robot relative to it's position and account for spaces that were both null and did not exist (if the robot was in a corner for example). Our time was spent mostly troubleshooting and testing the scan function. Like my experience with Mitch, we would each find minor issues that the other missed which helped to drive the process along. Paired-programming was beneficial when implementing this because the scan method could have been done multiple ways, and it was good to bounce different ideas off of each other to determine the most efficient way of implementing the function. We pair-programmed for about two hours.

Will's Experience:

For my first paired programming session I paired up with Mackenzie Power. We decided to work on the robot AI classes specifically focusing on the scan function. I had previously started on the scan function and was having problems getting it to work properly. It quickly became apparent that two eyes are better than one as the problem was quickly found and solved. I found that the code written in the paired programming time had far fewer errors than usual, and was far more efficient. While the code was better I do wonder if less actual code was written and work done in the allotted time. I believe that for more difficult problems paired programming proved to be very effective, but for the simpler tasks it seems like a waste of time as one person sits back with nothing to do.

Justin and Ryan

Justin's Experience:

I paired programming with Ryan for the stats panel. Working with Ryan was efficient and I did learn a different way to construct multiple panels together. He has started working on the panel before I joined him, but he was very patient explaining to me what things has he done and what their purposes' were. For me that providing a great understanding, so it was easier for me to help and follow his idea on building what he is doing. I took CMPT270 over the summer and our professor taught us a different method working on JPanels. I explained the way I was taught and compare it to his then we both agreed on using his method. From that I learned from him I think that will help me in the future if I need work with JPanels again. Altogether, I personally prefer pair programming over working alone, it is easy to find errors as someone keeps eyes on what you are coding and it is efficient as while one person is coding the other can watch and catch simple mistakes which sped us up.

Ryan's Experience:

During my pair programming session with Justin, I started off as the "Driver". We started off working on the GameController class trying to save information to use for creating the game board. After working on this for a while, we had a null pointer exception that we struggled with. We spent some time debugging by discussing what is going on and trying multiple things to fix it. Once we were at a deadlock, we decided to switch roles so I was the "Navigator" and Justin was the "Driver". We started working on the StatsPanel class which is a GUI. Justin was not as familiar as I was with the layout of JPanels so I guided him to create the Panel. During this time of a lighter subject I realized our previous problem had a simple solution and was able to quickly fix it once our pair programming session was finished.

Justin and Will

Justin's Experience:

Will and I were paired programming for the section of word translator, I initially could not understand how it fully works, but working with Will I have gained a much better understanding from the way he implemented it. Although I was not particularly helpful for building that class, I did help him catch small errors that more or less saved us sometime and I did learn something I did not know about before. Overall, pair programming with Will was basically a learning process to me, it would have taken me a much longer time to finish that section if I was working by myself.

Will's Experience:

For my second paired programming session I worked with Justin. We decided to work on the interpreter and the word translator, classes which I had been working on before and was having difficulty with. I yet again found that paired programming was very good at catching and preventing syntax errors and small mistakes in the code. This helped improve the efficiency and quality of the code making it more effective and compact. I yet again found paired programming to be simply better than programming alone significantly cutting down on wasted time.

Ryan and Mitch

Ryan's Experience:

For my second pair programming session, I paired up with Mitchel. During the session we worked on mapping the mouse position to properly detect when the mouse is the bounds of a hexagon. After some time of trying to debug we pulled out a pen and paper to figure out and visualize what was happening. We found the task to be quite frustrating but by working with someone else I was able to stay focused and on task much longer than if I were by myself. It was also helpful to has someone else's understandings of what is happening.

Mitch's Experience:

Ryan and I met for my second paired programming session. The second time around, the experience was completely different. The goal we had for the session was to make each hexagon clickable. Ryan started on the keyboard, while I showed him what I had and what was wrong. While explaining it, I thought of a few things to try that I hadn't considered before, unfortunately none of those things helped. Throughout the session we switched a few times, between typing and observing, every time one of us got stuck. My partner was able to come up with ideas that I had not thought of and in the end it worked slightly better than when we started.

Code Review Summary

For our code critique, we decided to evaluate a piece of code involving how our program draws the hexagon shapes and how we could fix it and make it more efficient. As it stood, it did not work correctly, which is one of the primary reasons why we chose to do our code critique on it. The code itself involved measuring values, taking the on screen position of a mouse click, and then using mathematics to determine if the position clicked was inside of a specific hexagon. The problem was that the lower and upper portions of each hexagon button would not work correctly by either producing no action or selecting and adjacent hexagon.

Our review of the code lasted approximately two hours. Initially it began with the two group members who wrote explaining in detail how the code worked, what it did, and where the problem was. To demonstrate the math, we wrote all the calculations on a white board with triangle diagrams to visually represent the code and problem at hand. After reviewing the current calculations and variables, we each then began to rewrite the mathematics the way we understood it to determine if perhaps there was an arithmetic error involved with the code. Initially no one discovered any errors when redoing the math themselves. We then wrote how all the variables involved in the code changed on the board and went through the code line by line to determine if there were any changes we were overlooking. We did discover that there was some calculations done that were redundant so we were able to delete them. Unfortunately, we did not find the bug that was causing the problem. Since we could not find the bug, we decided to look for an out-of-the-box solution to the problem that would perhaps help. Together, we came up with a solution that simply generated a square within each hexagon that represented each space on the game board. This resulted in spaces that would correctly detect if they were the mouse were clicked within each space. Unfortunately, it did not account for situations where a player would click within a triangle. Doing so would simply yield no result or errors. After this, we decided to correct the errors and commenting style of the group member's code. Informing them of the problems that currently did not match the style that the rest of the group agreed to and also that they should add in more comments as to what certain variables and functions actually represented. Once this was done, we decided to shelve the fixed code and come back to it once other more integral aspects of the program were implemented and working correctly.

While the experience of this code review was beneficial to the point that each member of our team now understands the math and calculations involved in generating a clickable space in the shape of a hexagon; it took time away from implementing and focusing on more important aspects of our program. We had chosen this code to review for two reasons: one because it was one of the more complicated pieces of our program (due to the mathematics involved) and second because of the presence of a bug which we had hoped to be able to find together. In hindsight, we should have chosen to do our code review on the interpreter because it was an integral aspect of our programming that uses the most code and has required the most amount of testing and troubleshooting. Reviewing the interpreter would have allowed all members to understand it better and be able to switch off working on it. As it stands, the costs of reviewing the code that we did resulted mostly in a loss of time from working on other functions and classes. Had we been successful in determining the initial bug in the code, then our opinion of the experience would have been much greater. However, and increased efficiency, understanding, and reinforcement of coding guidelines was achieved through the experience. If we had more time on this project, we would definitely perform another code review on something more important to the system, like the interpreter.

Design Document Changes:

Overall we have followed our design document quite closely while constructing our game. No dramatic changes have been made, but we have had to make quite a few additions to the various classes to achieve the desired outcome. These changes will be outlined according to their place in our architecture: Model-View-Controller.

Model

For our robot class, we have added a few more attributes which we felt were necessary to make it interact with other classes easier. One of the primary attributes we have added to each robot was to have a reference to the actual game board class so that each robot can actually interact with it more efficiently while moving and shooting. In regards to moving and shooting, we also wrote in the functions for both shooting and moving in the robot class. Each robot will now have a reference to three different stacks: the first being the mailbox stack of values, the second being a stack of forthValues which will serve as a stack to pop values from for robots being controlled by artificial intelligence, and the third being the variableStack which contained the variables created in the Forth code. We also implemented a hashtable in the robot class which contained the values of each individual forth variable. One last attribute was added to the Robot class, which is a 'isHuman' attribute to a Boolean value of true or false on whether that particular robot is controlled by a human player of not. We have also decided that for the robot classes designed to be controlled by artificial intelligence that they be able to record and keep track of their own statistics which will then be used to upload to the JSON file. The robot class also receives an attribute called 'codeInstructions', which references a JSON object containing its Forth code. This attribute is set to null for robots controlled by humans (since humans don't use code). To make things easier for the interpreter, we also gave each robot class three mailbox Stacks for each class of robot it can receive messages from. These are set to null for human robots, and robots of the same class will have their mail box set to null.

Within the three separate AI classes we added three different functions into each to correspond more appropriately with the given Forth code. These were the check function, which returned information about the square directly in front of the robot, the getDirectionOfEnemy function which returned an integer value representing the relative location of the enemy compared to the AI, and finally the getRangeOfEnemy function which returned a value of one, two or three according to how far away the enemy was.

For the GameBoard class we have added an attribute called aliveList, which is there for the system to keep on track of which robots are still alive so that only living robots can take actions on their turn. We also ended up removing the gameObserver class as it did not add any important aspects to the game.

In the model package Hex class, we started only having one attribute called robotList and a function called isEmpty. Now we have the Hex creation function which determines all-important integers required to draw the hex on the gamepanel as well as the draw Hex function which paints the hex shapes onto the gamepanel. Also we created three functions to draw the robots on hexes which we called drawHexWithScout drawHexWithsniper and drawHexWithTank, with each taking in the robots location and drawing it onto the gamepanel. Finally the last change in the hex class was the PointAtHex function, which creates the clickable location on each of the hexes.

Controller

In general, the controller class has remained the same, with the exception that we have added the interpreter class to it. One of the largest additions we have made to our construction which was detailed very little in our design was the interpreter. We created an Interpreter class to retrieve values from JSON and store them and also execute an array of Strings of robot Forth code. We also created a WordTranslator class which was essentially a HashMap/Dictionary with keys indexing anonymous functions to be executed for each Forth word. To execute functions, an Execute interface was created. The interpreter class will take in a robot class and a JSON object and with them will execute the forth code through the translator corresponding to the robot. Both of these classes were not talked about in our design document so we spent quite a bit of time deciding how we should go about parsing both JSON and forth code. Although currently we have a very good grasp and understanding of how the interpreter works with Forth code, it would have been much more beneficial if we put time into understanding it for our design document. This likely would have saved us a tremendous amount of time during construction.

View

In general, there were no significant changes to our view classes. We have updated the team selection panel which in our requirements document did not show the correct attributes each robot should have according to examples given in class. That has now been rectified. Provided we have the time, we will still add visual health attributes to the game panel (which currently are not implemented).

Summary

Overall, this project has been a good experience for all of us. It has given us the experience to work in groups following the procedure of software engineering. Being able to peer-program has shown us several benefits of doing so and proved to help aspects of construction be more efficient and faster. In the future, many of the projects we will work on will involve us utilizing both paired programing and taking the time to actually review complicated aspects of code to better understand them as a group and also increase their efficiency. While we were able to stick closely to our design document, we now know what must be focused on in the future for other projects to prevent larger changes from happening.