Construction Document
CMPT 370

Group C4
Jack Huang
Brandon Jamieson
Ixabat Lahiji
Daniel Morris
Kevin Noonan

Changes from our design:

- **Model:**

For every class we decided to make every field variable private since there are some fields that will not be changed after the creation of each class. We do not want any of these to be accidentally changed. Since each field value is private we implemented the appropriate getter and setter methods.

Board:

Robot:

Since the range statistic is an integer we changed the datatype of range from Hex to an integer. We also changed the absDirection() method to being a field. It was not possible to calculate the absDirection so it will have to be stored in the robot instead. We decided to make every field private since there are some fields that will not be changed after the creation of the robot and so I do not want them to be accidentally changed.

RobotTeam:

We decided to implement a team number attribute. This would allow the creation of robots inside of this class to know what team number they are a part of.

Spectator:

Nothing further was changed.

Hex:

Nothing further was changed.

Interpreter:

- **View:**

- **Controller:**

Pair programming reports:

Daniel Morris pair programming session reports:

The pair programming session I had with Jack went very well. Jack and I had both been assigned to work on the model classes within our model view architecture. During the session Jack and I decided to work on the more difficult methods within the model such as searching for robots within a specified range of a location and moving robots between locations. At the start of the session it seemed as if we were both very involved in the problem at hand and at least in my mind we were both trying to be the pilot, instead of having one of one of us being the navigator. After a while I started to realize that when I am not the one at the computer I did not have to be totally involved in the task at hand. After realizing this I felt that when I was not the one sitting at the computer that I could think more lightly on the problem at hand and focus on the bigger picture. I definitely think this helped us become more efficient and ensure that the person at the computer was not too fatigued and could work very effectively.

At first I did not think the pair programming session with Kevin was going to work out very well but it turned out to be very successful. During the session Kevin and I decided to work on joining the model, view, and controller together. This turned out to be difficult for us since that there were some classes that either of us hadn't worked with yet since they were made by other team members. Although with that being said we were able to decipher what needed to be done to join the classes together and we started working on that. The main problem that we came across was that the panels in the view had initially been constructed with separate listeners for every button or combobox they had displayed. Due to the architectural choice made in the design document we were not able to make changes to the model within these listeners. The solution to this problem was to make the controller an action listener and whenever a view panel received input from the user the panel would trigger the listener within the controller with a unique actionCommand. The controller would then use this command to make the necessary changes to the model and update the view accordingly. Once we had decided on this solution it got the ball rolling for us and it became a very efficient pair programming session. We were able to take turns as the pilot / navigator to maintain our focus and went on to accomplish the task at hand.

Brandon Jamieson pair programming session reports:

November 23rd

My first pair programming session with Kevin was a great experience. We decided we would implement our PlayMenu, in which players will select teams and board size. I acted as the driver initially while Kevin served as the navigator. It took some getting used to but once we got in the groove of things, pair programming proved to be quite efficient. Implementing the GUI elements was simple with someone telling me exactly what listeners and elements needed to be made. On top of this, I found I was much less distracted when working with someone else and a lot more focused on the problems in front of us. After about an hour, Kevin and I switched roles and we continued to build the menu functionality, implementing board size and team selections.

During this, being the navigator also turned out to be a positive experience. Saying what I was thinking out loud really made me consider the reasoning behind my decisions in-depth. I also found it a lot easier to focus on the overall state of the menu, as well as what needed to be completed next without worrying about the actual coding of it. After a couple hours and several role switches, we finished the menu's functionality, as well as general formatting. We had been having troubles cloning the repository to commit, and also ended up solving the issue before ending our session (a directory name contained an invalid character and prevented any Windows users from cloning). Overall, I would say the session went smoothly with no real snags, and that it was a positive and fun experience.

November 26th

My second pair programming session with Ix went just as well as my first session had. We met up on the weekend and promptly decided we would work on finishing up the InGameMenu and connecting it to the model through the controller. Again, I served as driver first while Ix navigated for me. We decided it would be best to change our design and create an InGameMenuPanel class for holding all gameplay visuals, and have the InGameMenu class serve as the frame for it. Coding this transition turned out to be quite easy with Ix's help, having no real issues that a couple minutes of thinking didn't solve. It was very helpful having a different viewpoint on the tasks, and made me closer consider alternative ways of doing things.

We switched after this, and decided we would implement a way to get robot images by loading them from our resource directory. After some quick research, we learned how to accomplish this and created a method for determining the correct image based on the robot type and team. We connected it where needed in the panel, as well as cleaned up a large amount of errors and loose code. After this we switched and began hooking up the InGameMenuPanel to the controller. During this, we needed to decide how each private listener class would get a reference to the controller, and concluded that modifying the Controller to use a singleton pattern would be an elegant solution. Implementing this concept was simple, and made the rest of the panel easy to make, taking about a half hour, taking us to the end of our session. This session was both productive and informative; I learned new things and caught up on other parts of the development from Ix.

Code review reports: