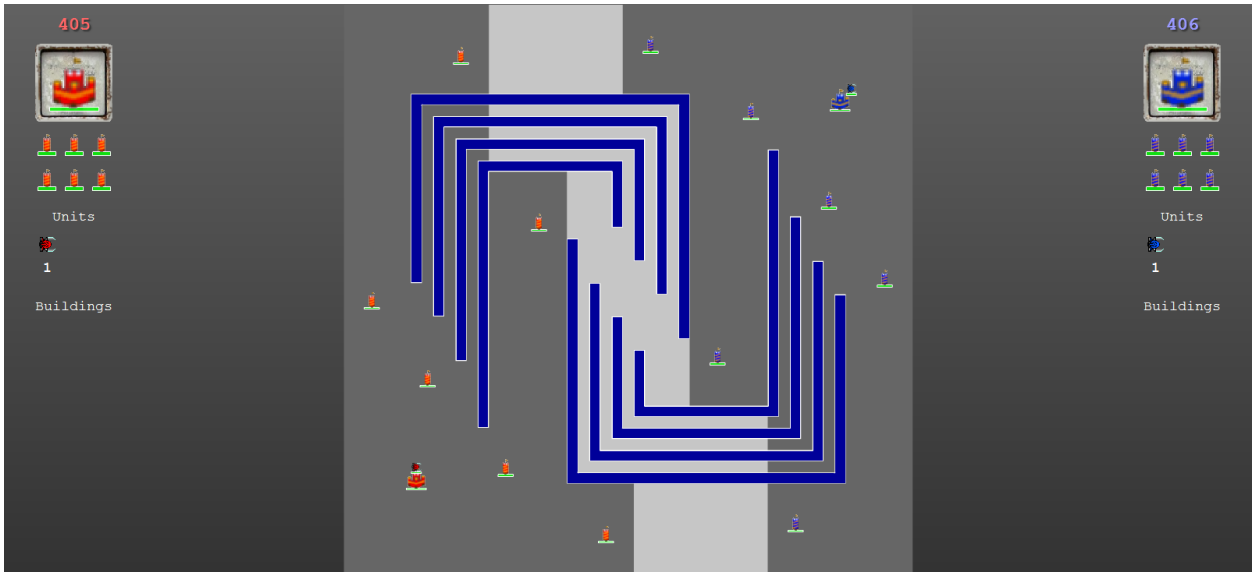
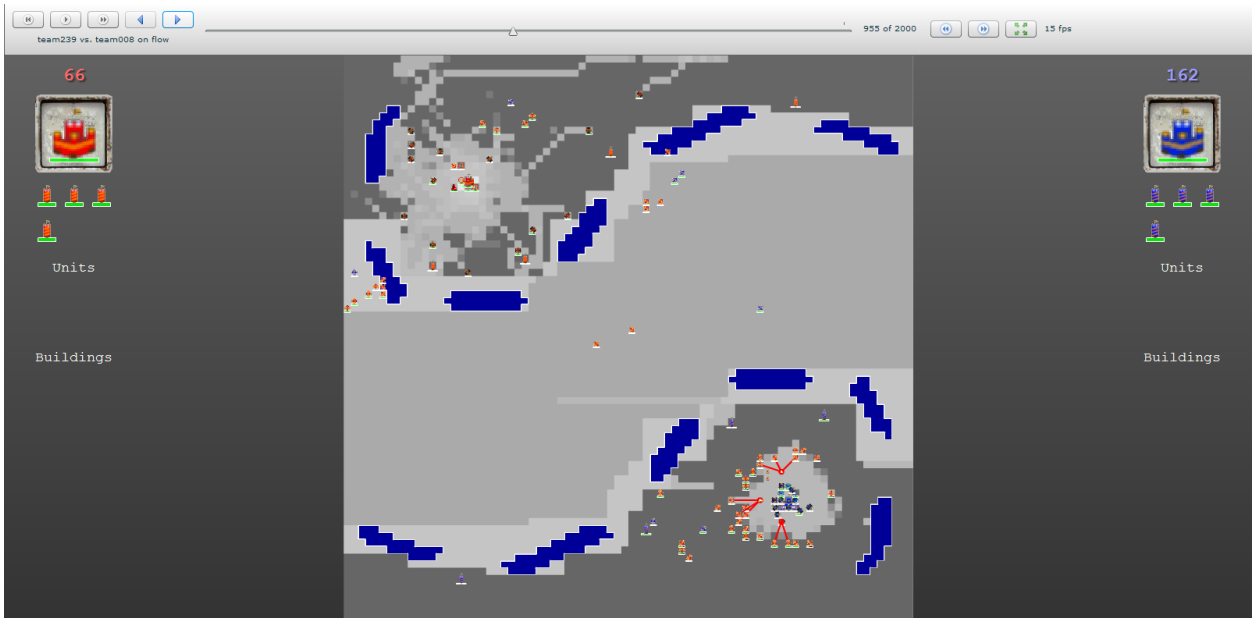
In this years Battlecode competition two AI players fight to destroy each other's HQ, which sounds simple enough right? The catch is that we had to script our own bot to fight the enemy bot effectively, which turned out to be a bit of a challenge. Each player starts each game with a headquarters, a set amount of ore and a predetermined number of towers (of mass destruction I might add). To win the game you either need to destroy the enemy HQ or have more combined tower health than your opponent at the end of the match. The first objective in every game was to build a beaver which can then mine ore, (pretty much the most important thing in the game) build structures, or occasionally fight. Once the first beaver is built many more options are opened up as the beaver is the only unit which can construct buildings. It was kind of the norm to build a miner factory to produce miners after the first beaver was built, which were more effective at mining than beavers!



I know you're probably thinking Battlecode sounds easy up to this point, but all these actions require code to run in the background, and thats where it starts to get tricky. Before actions such as mining, building, or attacking can be executed there are some checks that need to be run before the action is set in motion. All these checks were run by each and every unit

on the battlefield during every round of the game. All these checks are written in Java code which is written by the human, and trust me it gets more complicated! After getting a suitable number of miners working to help you build the ultimate fighting force the resources and potential for barracks, supply depots, helipads, factories, etc were all at your fingertips. So now that income is being generated, what's next you ask? Well, the object of the game is to destroy the enemy, so naturally we want to build offensive units such as soldiers, tanks and flying drones. In the early games of the tournaments many teams resorted to a drone heavy rush attack, which surrounded the enemy HQ and prevented enemy units from leaving their death grip(as can be seen below implemented by the red team). After a certain condition was met(set by the human) the drones would then go tower to tower leaving destruction in their wake and eventually turn their sights to the enemy headquarters. Another important aspect to focus on while writing our bot was supply. During the course of a game units ran off of supply which was produced by supply depots and maintained by relaying it through units. It was essential to keep your units supplied if you wanted them to work at maximum efficiency. Everything was doubled at full supply of a unit; movement speed, attack speed, mining speed you name it.

A major part of Battlecode is the pathing and unit micro. Each map is symmetrical so it comes down to trying to figure out whether it is on the diagonal or the horizontal. Along with finding out which kind of map symmetry comes finding an effective pathing method. In Battlecode there are certain methods that you can use to find out various things about the map and its orientations, such as finding which direction the enemy HQ is in. You can then use these in combination with querying the map terrain to find a path to the enemy HQ. The one hard thing is sometimes the terrain is laid out in a way that proves to be difficult for certain movement algorithms so you have to come up with ways to circumvent this.



Tactics changed quickly as rebalancing weakened some units and strengthened others and teams had to adapt, but it was hard to judge what the final tactic would be for the finals. Although we didn't make it very far we did learn a lot, and actually won a game! Battlecode was constantly evolving with each tournament, and for my teammate and I it was hard to get a functioning bot that could retaliate to the enemy strategy properly. Especially for me having no prior experience in Java before this competition, concepts came easily it was just the code implementation that was difficult for me. Although I would say it's easy to catch on to if you apply yourself and watch the lectures put up by MIT.

At the beginning of this years competition things got started off a little slow. We weren't exactly sure which direction to head other than following the basic code given to us through the online lectures on the Battlecode website. It was difficult starting from nearly scratch, but not impossible. Our goal coming into this year of Battlecode was to beat the reference player put out by MIT, and attempt to be competitive in the tournaments. We learned quickly that it takes a lot more time than what we put in, and the results reflected that to us. But we took the losses as a learning experience to help improve our code, taking ideas from opponents and watching the victors strategy from the sprint cup. We learned quick and adapted to the change in what the majority was trying and attempted to implement it effectively.

I'd say the only major problem we ran into was GitHub... and I mean major. We couldn't figure out how to let a second person push his code up to GitHub so it just all together slowed down the whole process as we both had to be together to program our bot. When it came down to the final tournament and beating the reference player, we had finally come up with a working bot that won us a game in the final tournament! After losing out of the tournament we focused all of our attention to beating the MIT reference bot. It wasn't hard to beat it after all since its strategy didn't change over time. All we had to do was edit our current bot and ultimately switch up our unit composition. To beat it we relied a little more on tanks rather than drones, which is the opposite of what won us the tournament match. The thing I felt helped the most in winning games was the upkeep of supply. All of the best teams usually had plenty of supply depots and it sped up the process of everything! Supply might have been overlooked by us early on, but we realized that it was a key element to having a good bot. Although we didn't necessarily do very well I feel we learned a lot from the competition and it really helped to hone coding skills! I would highly recommend Battlecode to the avid programmer, and beginners as well, but it is just going to take a little extra work. It was a good thing one of us had prior experience in Battlecode, because overall it made it a better experience.

Our BattleCode 2015 repo:
https://github.com/MattJohnerson/BattleCode2015