Battlecode 2020 - Serpentine SnakeEyes

On the 7th of January Battlecode started. Battlecode is a programming contest organized MIT in which you have to create a bot that can battle automatically against other bots. This year's edition was all about climate change and the rising water level. Our team consisted of: Tristan Tomilin, Carlo Lepelaars and Pieter Voors. You can find the codebase for our submission on GitHub.

Game summary

In this year's game of MIT Battlecode you battle on a 2-dimensional map of size varying around 40 by 40 tiles. Both teams start with a headquarters which can produce miners. These miners can explore the map to find soup. When they bring soup back the the headquarters or to a refinery the soup is added to the team resources and can be used to build other building or robots.

The main idea of the game is the fact that there is water that is slowly rising. Whenever your headquarters flood you have lost the game. You can prevent the water from flooding your headquarters by using landscapers to dig dirt at one position and drop it at another position. The game also contains drones which can fly over water or high terrain and can pick up and drop other robots. As a defensive measure net guns can be build, which can shoot drones out of the air.

One challenge in the bot that you have to build is the fact that all robots run a separate instance of your code and all can only observe the world in a limited circle around them. This means that you do not have shared variables for inter-robot communication and do not have full vision of the map. To communicate between robots a block chain is available which functions similar to a queue of messages. This block chain can be read by all robots of both teams and is limited in the amount of messages that can be sent.

For a more elaborate explanation of the game and its specification consult the full game specification.

First approach

To start of the competition we first wanted to get a simple version of a bot running to get a sense of the code, get more ideas and have a basic version to expand upon. Our first approach consisted of having all miners walk in random directions mining soup if they come across it and then returning that to the headquarters. After a set round we would build a design school and start producing landscapers, which would build a small wall around the headquarters. While this

was an effective strategy to quickly get decent results it turned out that almost all teams had started with this idea.

Improved exploration

To improve the way the miners search for soup we implemented a direction-based approach. Whenever any robot is exploring to find soup it picks three consecutive directions of all horizontal, vertical and diagonal directions. Then it picks one of these three directions to walk in repeatedly. This results in the robot walking in one of the directions in general, so it does not get stuck as easily and they do not all walk in exactly the same path. If it reaches a point where it cannot continue in that set of directions then a new set of directions is chosen to continue exploration without walking into a wall, the end of the map, water or otherwise non-traversable terrain. We also added the functionality of sending a message on the blockchain to inform other miners once a miner had found a big portion of soup.

These additions improved our bot significantly as now the miners were a lot quicker at finding sections of soup and more resources were gathered before the water got too high to explore the map. The fact that soup was starting to be mined more quickly also helped to create more miners in the first phase of a game, which could then directly go to a reported section of soup resulting in even more soup being mined in the first phase of the game.

Another small problem with the mining of soup lies in the combination of building a wall and the fact that soup needs to be refined in either a refinery or the headquarters in order to be used. Once we started building a wall the miners would not be able to bring back soup to the headquarters anymore, stopping our income of soup as soon as a wall was starting to be built. To counter this we build a refinery next to each large portion of soup. Sometimes this resulted in multiple refineries being built as the miners could not see refineries on the other side of a portion of soup, but in general this was very useful to start building the wall earlier and keep mining soup for longer as well as miners not having to walk back to the headquarters every time.

Drone exploration and enemy stealing

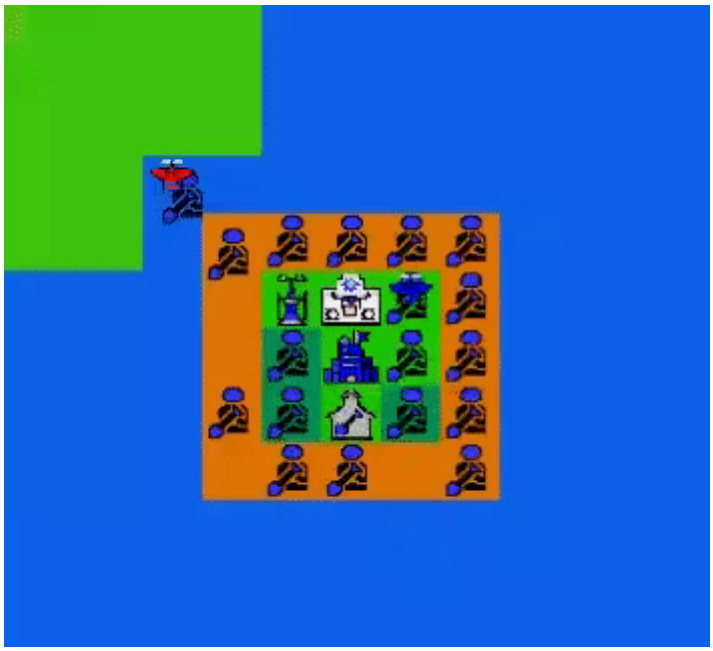
To protect our base from invading landscapers and to lower the unit count of the opponent we implemented a drone strategy. After a first phase of mining soup we build a fulfillment center to start producing drones. These drones then use the exploration strategy designed earlier to explore the map. The two main benefit of drones are that they can are not stopped by walls and water and that they can pick up units. The drones are therefore very effective at exploring the map. When a drone spots an enemy unit it will fly towards it and pick it up. Then the drone will continue exploring the map until it finds a spot of water, in which it will drop the enemy unit breaking the enemy robot. This way we could throw most of the miners of the enemy in the water, hurting their income.

An important factor to take into account while exploring the map with drones are net guns. Except for net guns, no robot in the game can hurt other robots directly. The net gun can shoot drones out of the sky when they come within its range. Luckily the range in which a drone can observe the world is slightly larger than the range in which net guns can shoot. Therefore we could make sure that a drone never went into the range of a net gun. With this implemented our drones could not be shut down anymore and would remain alive for the rest of the game.

Drone attack on enemy headquarters

One main part in which drones can be very useful is in the part of the game where most of the map is flooded with water and both teams are building up a wall around their headquarters. We would want to fly over to the enemy headquarters to then drop their landscapers in the water. Finding the headquarters was simple enough. While exploring the map if any robot spots the enemy headquarter, it sends a message in the block chain with the location such that all robots are aware of the enemy headquarters' position. Stealing the landscapers of the wall is hard since each head quarter contains a net gun, making that we cannot get too close to the enemy headquarters.

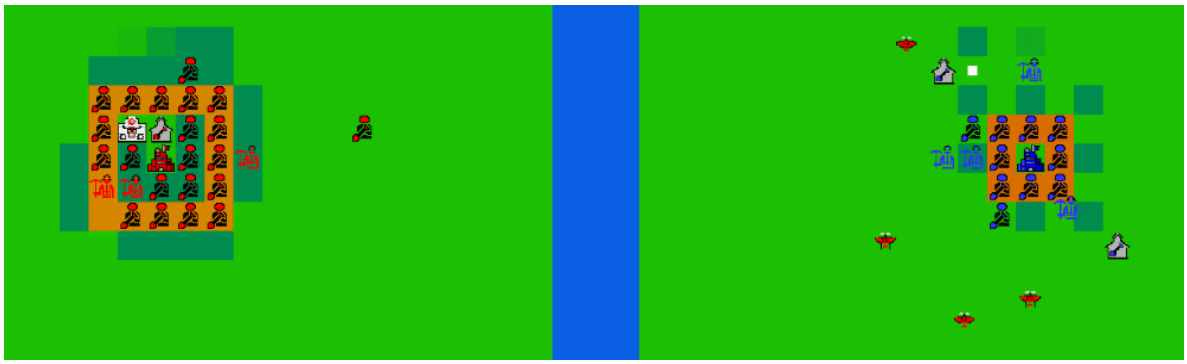
As previously mentioned we had already implemented a safety in our drone's movement system such that they will not move into the range of any enemy net gun which can shoot them down. As such we could not and should not get close to the enemy headquarters. We found that whenever the enemy would build a slightly bigger wall of 5 by 5 instead of 3 by 3, we could pick up enemy landscapers that were standing on the corners of their wall without being shot. For most teams this resulted in them noticing an empty spot in the wall and sending another landscaper to that spot periodically such that we could then pick up that landscaper and safely drop it in the water, slowly emptying their wall of landscapers.



While the corner stealing works well for enemies with a large wall, this of course does not work for enemies which build a small wall around their headquarters. For this we added another strategy, which we call the drone rush. In the second part of a game where most of the map is flooded, we keep creating more drones in our base. These drones then all fly over to the enemy base circling around it. Once a set amount of drones have gathered around the enemy base all drones at the same time move in to the wall to steal enemy landscapers off of the wall. Because they all move in at the same time, the net gun on the enemy headquarters cannot shoot down all drones as quickly.

One important difficulty here is the fact that all drones run their own version of the code and can only communicate via the shared blockchain. Therefore it was hard to ensure that all drones would start attacking at exactly the same time, which is crucial to avoid all drones being shot by the net gun too quickly. In the end we solved this by putting a message in the blockchain once a drone detected that there are enough drones present to attack. In the round that this message actually arrives in the block chain and all drones will read this message all drones attack. Care was then taken to ensure that attack messages from previous attacks were not confused with attack messages of a second or later attack, as otherwise any drone created after the first attack would fly directly into the enemy headquarters thinking it was time to attack.

While this approach required a lot of tweaking to get right, in the end, this resulted in a strategy that a lot of teams were not prepared for. Some teams could counter this strategy by repopulating their wall with new landscapers when we stole landscapers or building multiple net guns in their base such that our drones would be shot down too quickly and from too far away but in general if we could get enough drones to the enemy base, we could significantly reduce the number of landscapers on their wall with this strategy.



Base construction

For our own base design, we decided to build a wall of 5 by 5. The main reason to do this is because we make use of a fulfillment center inside the base that survives until after the largest portion of the map is flooded such that we can still create new drones. Since we use a bigger base, we can also put a design school inside our base to build new landscapers. This was useful in case the opponent was stealing landscapers from our wall or if we could not get enough landscapers on the wall initially. We now can put new landscapers on the wall while it is already being built. Another related problem is friendly miners or enemy landscapers or other enemies taking up space on our wall where we want to put landscapers and them potentially attempting to bury our headquarters. Since we now have drones around our base, we can use these drones to pick up enemy units on the wall. For friendly miners that were accidentally on the wall taking up a landscaper's spot we decided they should just suicide and thereby remove themselves from the wall.

This resulted in being able to generate more drones for our other strategies and keeping a fresh supply of landscapers on as many spots of the wall as possible. If the wall is full of landscapers, building up the wall is easy: each landscaper simply keeps raising the wall underneath itself. When the wall is not entirely populated, however, you need to balance which part of the wall you are fortifying. To do this we made sure to always put dirt on the lowest part of the wall, unless it is only slightly lower than the part we are next to and thus requires walking, which takes time to do. In that case first we continue raising the wall close to the landscaper before it walks over to a lower spot to minimize the time spent on walking. Because the wall is square it is not possible to walk directly from one side of the square wall to the other, so we had to implement a separate piece of logic to find an efficient path between any two parts of the wall. Together this gave a more robust wall building system that raises the wall as quickly as it can with the resources available.

Results

During the development process we repeatedly tested our bot against other teams by means of scrimmaging. The Battlecode website also continuously ran games with the latest version of our bot against other teams to determine a general ranking. In the end we ranked 76th out of the 285 participating teams, with a score of 1545.

Besides this ranking, there is also a tournament where you are matched against random opponents and using a double elimination brackets battle for the finals. With our final bot, we were able to win the first match of the qualifying tournament. In the second match we were defeated, sending us to the losers bracket. In this bracket we were immediately faced against the other team of Serpentine, team Noodles, who won from us. The entire bracket and results of this tournament can be seen on the corresponding Challonge bracket page.