Battlecode 2020 Postmortem

The High Ground (Eli Lifland, Aaron Ho, Alex Hoganson)

1 Introduction to the game

This year, the goal of the game was to have the last HQ remaining. An HQ was destroyed either by being flooded with water or by being buried with dirt. The map consisted of varying levels of elevation, with lower elevation tiles flooding first as the water level rose. Most units could only move or build onto tiles that had a similar elevation to the one they were currently on and robots that ended up on a flooded tile were destroyed. The resource used to build robots this year was called soup and was scattered throughout the map. There were 3 types of units belonging to a team:

- Miners: Can only be built by the HQ. Can mine soup from the map and build buildings.
- Landscapers: Can change the elevation of tiles by digging and depositing dirt.
- Drones: Can move on any elevation, even over water, and pick up and drop other units. Slower than miners and landscapers since if they weren't they'd be insanely overpowered.

Finally, there was 1 neutral unit type: cows. Cows move randomly around the map and cause pollution in the area around them, making units near them move slower. Miners can build 5 types of buildings:

- Refinery: A place for miners to deposit soup they have collected from the map.
- Design School: Produces landscapers.
- Fulfillment Center: Produces drones.
- Net Gun: Can one-shot kill drones in a relatively small radius.
- Vaporator: Produces 2 soup per turn for a cost of 500 soup.

The HQ had the capabilities of a Refinery and Net Gun built in. All buildings could be buried if enough dirt was placed on them by enemy landscapers without being removed by friendly landscapers.

Finally communication was done through a "blockchain," in which units could bid soup to send messages, and the top 7 per round were published anonymously to both teams. This meant that in principle it was possible to mess up your opponent by either spending a lot of soup to spam messages and prevent them from communicating, or mess up their communication by sending messages they would interpret as their own.

2 Week 1

For the first week, we were some combination of busy and lazy. We cloned the lecture player to a git repo and that was about it. We decided not to submit a player for the sprint tournament and eagerly awaited to see the meta from the results instead.

3 Sprint Tournament

In the sprint tournament we ended up seeing three main strategies:

- Rush/turtle: Run a miner to the opponent's HQ, build a Design School and perhaps Net Gun, then bury the opponent's HQ with landscapers (See Figure 1.). In case this doesn't work, turtle as described below.
- Turtle: Build a circle (or something resembling one) of landscapers around your base, and build a huge wall to survive the flood as long as possible. See Figure 2.
- Terraform/attack: Utilize landscapers to "terraform" the whole map, making it walkable for all your units. Teams would build up their "lattice" to a height such that it wouldn't flood for a while, then place buildings on this lattice. Particularly, vaporators were a powerful building for this strategy, allowing for a huge economy to be built up and producing a snowball effect. Soon before the lattice would flood, terraform/attack teams would "crunch" with their drones, removing enemy landscapers from their turtle wall and replacing them with their own, then burying the enemy HQ. Another component of this strategy was a drone harass, in which teams would send drones toward the opponent HQ to find enemy units and flood them. This proved to be effective at limiting the opponent's economic capabilities. See Figure 3 for an example win.

The first place team was **Bruteforcer**, who utilized the terraform/attack strategy very well. Second place was **Battlegaode**, the best rush/turtle team.

4 Week 2

After the sprint tournament, the devs nerfed vaporators from cost/production of 1000/7 to 500/2. However, we and others on Discord thought that ter-

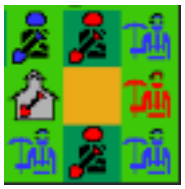

Figure 1: **Battlegaode**'s rush burying the enemy HQ.

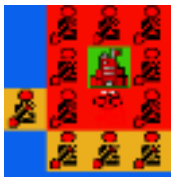

Figure 2: **Battlegaode** turtling after a failed rush.

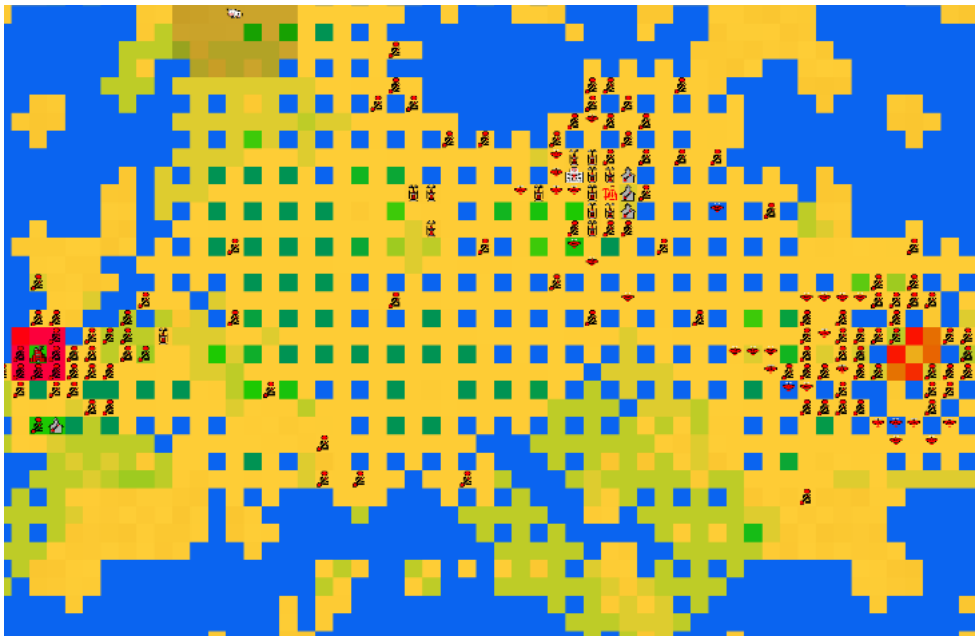

Figure 3: **Bruteforcer** burying the opponent after terraforming the map.

raform/attack was still viable as it still paid for itself within 250 rounds. While we still thought that rush/turtle and terraform/attack were the two best strategies we had seen, we decided to first improve the lecture players basic turtle to get our bot up for some scrims. We added some basic infrastructure such as bug navigation and basic communication, and improved the lecture player in a few ways such as making miners run toward nearby soup.

As we were making these basic improvements to our bot, we noticed that we kept losing to rush/turtle teams, and that lots of rush teams were rising up the leaderboard with former top terraforming teams **Bruteforcer** and **Super Cow Powers** ineligible for future tournaments. Rush bots did very well in this period in part because rushes are quicker and easier to code than an equally powerful rush defense or terraforming bot. Additionally, the early map pool

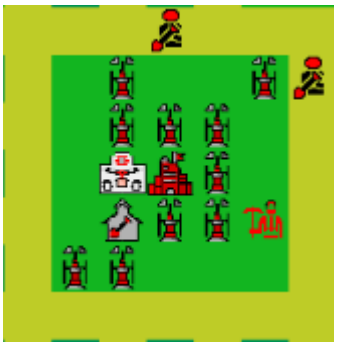

Figure 4: **Java**'s wall and the open space inside it.

was weighted toward small and easily pathable "rush-friendly" maps.

The night before the seeding submission, we realized we didn't have enough time to code a good enough terraforming bot and we would lose to both good pure turtles and rush/turtles. We strongly valued a high seed and reasoned that we would eventually want a good rush bot to test against anyway. We quickly coded up a rush bot, submitted it and woke up to being 2nd place on the scrimmage server! This confirmed our suspicion that rush bots were relatively easy to code, and we spent the rest of the day of the deadline ironing out some bugs. One thing that separated us from some other rush bots is that our rush miner would first run to the middle of the map in an attempt to determine which of 3 possible symmetries the map was, and then rush toward the opponent. This made our rush worse in cases when we would have guessed the right symmetry anyway, but more consistent overall.

5 Seeding Tournament

Although a lot of the top teams were rush/turtle, the devs indirectly nerfed rush through large and hard-to-path maps in the seeding tournament. Some very strong rush teams were knocked out pretty early, and we barely scraped by in several sets before getting eliminated in 4th. We were fairly happy with this seed. The team which got 1st was **smite**, who combined a decent rush with a post-rush turtle particularly robust to the new maps. 2nd place was **Java Best Waifu**, who had taken the mantle as the top terraform/attack team. We will refer to them as **Java** from now on. **Java** had a unique strategy of leaving area near their HQ open as they built a 7x7 square wall around their HQ, allowing them to build buildings in the open space inside the wall before they really had terraforming going. This wall and open space is shown in Figure 4. Once they got the wall and some vaporators up, they did very well, but in a few games they failed to get their wall up giving **smite** the victory.

6 Week 3

We thought that although **smite** got 1st place in the seeding tournament, Java's terraform/attack variant was the future. This was due to **Java**'s first place rank on the scrimmage ladder, and the potential room for improvement in the terraform/attack as opposed to rush/turtle. So, we decided to shamelessly copy **Java**'s strategy. As Steve Jobs said, "Good artists copy. Great artists steal." We pulled up some replays of **Java**'s games and worked on a new bot for several days until we imitated it as closely as possible, except we made a few different decisions:

- We built in a nicer pattern than them. This meant that we were less likely to block ourselves in, but would struggle more with maps where a good chunk of the terrain wasn't terraformable.
- We had a cleaner build order than them, perhaps due to the build order prioritization system we set up, where we had a desired unit composition and robots would have a priority representing the minimum soup required to build. A higher ratio of actual units of a type to desired units of a type meant a higher soup priority.

Similar to **Java**, we set up a scoring system for landscapers such that they would dig from and deposit to the highest-scoring tiles. However, our system was a bit less robust than theirs, perhaps due to the fact that they had a clever way of checking all the locations within sensor radius with a small amount of bytecode. We struggled to check all the locations in our sensor radius before running out of computation.

Our bot was also similar to them in a funny way: both of us completely ignored cows and pollution. Due to the small amount of cows on most maps and lots of other teams drowning cows for us, it wasn't too big a deal in most games. We always had higher priority items to work on.

During Week 3, turtle and rush/turtle teams began to more reliably attempt to get enough economy to build up drone walls. If a full drone wall was formed around your turtle as in Figure 5, it was impossible to crunch on as drones could not move through other drones. Even a partial drone wall could impede the progress of a crunch, and even just a few drones could be used to drown enemy landscapers dropped on the wall. Drone walls could only be defeated by terraforming to the other team and building net guns to kill the drones.

Some teams took defending their turtle to a bigger extreme. A 4th archetype began to become pretty popular in Week 3: mini-terraform/turtle. Teams such as **smite** and **Bowl of Chowder** would terraform just enough to build up an economy and defenses of drones, net guns and landscapers around their turtle. They called their terraforming formation a cookie, and we refer to this strategy as such from now on. When the cookie was fully set up as in Figure 6, it was almost impossible to break through, even if you terraformed to the other team. The only way to stop these teams was to harass them enough that they couldn't get their cookie up. We think that if the tournament had run a few more weeks,

Figure 5: **Steam Locomotive**'s turtle with a drone wall.

Figure 6: **smite**'s cookie, from the qualifying tournament.

cookies would be the dominant strategy as defense would prevail over offense. Additionally, the economic advantage from land control was only temporary as the map would flood at an exponential rate.

One more theme in Week 3 was that every top team, no matter what their strategy, now had a drone harass. A drone harass was essential to attempt to sabotage your opponents' strategy, no matter what it was. Thus, we decided to direct our miners to try to run toward the enemy HQ location along our lattice, effectively making them be near the edge of the lattice and ready to build net guns at almost all times. We built net guns whenever we saw drones, but had a few restricting conditions: we kept at least as many vaporators as we had net guns until we stopped building vaporators, and we would usually build net guns at least 8 distance squared away from each other. Additionally, we would only build net guns in the "corners" of the space inside our wall to help landscapers get our initial wall up, or on top of our lattice such that they wouldn't get flooded until round 1600. All conditions except the "inside corner" and vaporator ones were relaxed whenever we were close to the enemy HQ (we love offensive net guns) or could see lots of enemy drones and buildings. Figure 7 shows an example of our net gun formation.

We spent a lot of time coding during Week 3, and it paid dividends. We were able to rise up to the front page on the scrimmage rankings, and were beating lots of teams with our **Java** clone. However, we still struggled with rush teams. Unlike many other teams, we didn't keep a drone back to defend rushes, which made us weaker vs rush teams but stronger vs other non-rush teams, as the harassment from the early drone sometimes snowballed into a decisive economic advantage. We especially struggled with the rush team **Kryptonite**, who would often save up a ton of soup as they rushed to our base, then as soon as they arrived spit out a ton of units, while we had been busy building up our economy. Additionally, **Kryptonite** prioritized surrounding their own net gun over destroying our HQ, which often led to extended standoffs as we were unable to build drones and unable to bury their net gun. This messed up our build order as we had our HQ signal when rushed, and our miners stopped building vaporators when we were being rushed.

Figure 7: Our net gun formation: notice the 4 net guns on the inside corners. Each net gun is at least 8 distance squared from the nearest net gun, except for on the left edge where we were coming into contact with the enemy's lattice.

Despite our approximately 50-50 win rates against **Kryptonite**, we were beating all other teams the majority of the time in scrimmages, so we were fairly confident going into the qualifying tournament with our 4th seed. We were able to attain rank 1 on the scrimmage ladder which was another confidence booster.

7 Qualifying Tournament

But of course, it can never be easy. Despite having the 4th seed, we ended up being matched up against **Kryptonite** in the round of 16, who had the 20th seed due to not building their rush bot until after the seeding tournament. However, we luckily managed to defeat them in a nerve-wracking 3-2 best of 5, in which we lost the first 2 games. We won the games where they either failed to path to us or we happened to pick up and drown their rushing miner right as it was reaching our base. Deservingly, **Kryptonite** ended up qualifying from the losers bracket. However, **Chicken**, who had a seed in the 50s, was not so lucky. Despite being on the top page in the scrimmage rankings, they failed to qualify due to being matched up with **Kryptonite** even earlier than us and facing a tough matchup in the losers bracket. We feel their pain as we failed to qualify in a similar fashion in 2017.

8 Final Adjustments

After the qualifying tournament, we had just a day to make some final improvements to our bot. We wanted to make some improvements and bug fixes with our drones, but our drone class was a bit of a mess code-wise. We made the perhaps unwise choice to rewrite our whole drone class that night, staying up

Figure 8: Us continuing our normal build order despite **Kryptonite**'s rush resulting in a stalemate through round 900.

until 5 AM debugging the new version. While the new version worked fine and both removed some bugs and implemented some new features such as dropping miners off in the crunch to build net guns and harassing enemy soup locations, it was a lot of time spent on something that could have been very little time spent if we had written the drone class in a more extensible fashion earlier. This may ended up costing us a lot.

The other main focus of our last day was implementing a strategy to counter Java. Since neither of us built any sort of turtle wall, both of our lattices and HQ would flood at round 1640, making games between us a coin flip. To counter them, we decided that if our crunch failed we should run back with our remaining landscapers and raise our 7x7 wall until we flooded. We ended up getting this to work but it was very buggy as it was implemented in the very last few hours. Additionally, we did not want to test it against **Java** or **smite**, who we viewed as our main competitors for 1st place in finals and thus didn't want to give them any information about our last-minute changes. This meant that bugs which appeared against them but not ourselves would go undiscovered.

We also made a last-minute change to somewhat counter **Kryptonite**'s super annoying extended rushes: return to normal build order at round 400 whether we are being rushed or not. While hacky, this solution allowed us to win some games against them even when they had a net gun surrounded near our base, as illustrated in Figure 8.

Finally, we introduced the notion of pollution to our bot in the very last hour before the deadline. We made it so that if we were going to move to a square that had so much pollution that our sensor radius would be less than 2, we wouldn't. That was all we had time to do, and still didn't drown cows or drop cows off near the enemy team like some others.

9 Final Tournament

We went into the final tournament feeling cautiously optimistic about our chances as the 1 seed with a counter implemented against Java. However, we knew we were still vulnerable to rush teams.

The first surprise of the final tournament came when **Prasici** successfully tricked **Steam Locomotive**'s communications by broadcasting the first message their seeding bot had sent. This hilariously caused **Steam Locomotive**'s landscapers to run towards the edge of the map, trying to get to where they thought their HQ was. We were initially very concerned by **Prasici**'s communication sabotaged, but it turns out that it wouldn't have affected us as we made sure to change our communication encryption constants right before we submitted.

One more shoutout goes to **Bowl of Chowder**, who was the only team to successfully implement self-destruction of landscapers in order to create islands to build net guns to break drone walls. This was something that we never got around to doing and it was very exciting to see it in action. It was the only way to defeat a drone wall without terraforming all the way to the opponent.

A final shoutout goes to **Bagger288**, who implemented a strategy where their landscapers would self-destruct and be replaced with net guns if they detected they were soon going to get crunched on. This would have been a fantastic counter to us if we hadn't implemented our last-minute turtling.

We were able to take our first two matches against **horsepaste** and **Bowl of Chowder** 3-0 and 3-1. However, we were matched up against **Battlegaode** in the semi-finals on winners, who was still a top rush/turtle team. One of the games we lost to their rush. The other three games, we were able to defend their rush, however they had a new feature where they would build defensive net guns on the edges of their turtle to defend against crunches. This worked very well against us in the games where we were not able to harass them well enough or terraform to their HQ. Thus, they were able to defeat us 3-1.

Once we fell to the losers bracket, we were able to defeat **the-levee-builders**, matching us up against **smite**. We weren't sure what to expect since we hadn't scrimmaged them in a little while. We knew that if they got their cookie up, we were probably screwed as the cookie formation was very difficult to break. They ended up defeating us 3-2, with 2 of their victories being on maps with very many cows. It turned out that not dealing with cows came back to haunt us, as in one of those matches there were so many cows near our HQ we failed to get our initial wall up, and in the other it slowed our progress considerably, making our drone harass less effective and allowing **smite** to get up the vaunted cookie. While scrimmages later revealed our bot's win rate against **smite** was fairly high, it was fair for us to get eliminated based on not dealing with an important aspect of the game.

Our loss against **smite** meant we got 4th place. **Java** ended up winning, defeating both **Battlegaode** and **smite**. It turned out that **Java** also implemented a last minute fix to counter us, and through scrimmages we found out that they had a good win-rate against us. When we hadn't happened to build

net guns in all 4 corners of our initial setup, their drone harass was able to pick landscapers up off our last-minute turtle, allowing their last-minute turtle to survive longer than ours. It was pretty hilarious to watch with them and discuss how both of our last-minute strategies could have been easily improved with more time.

10 Final Thoughts for 2020

Thanks to teh devs for creating a super interesting and fun game and for running things very smoothly this year! The return to the tried and true Java engine removed many hiccups from the process that were present in 2018 and 2019.

Congrats to **Java Best Waifu** for getting 1st, and thanks to all the competitors for a fun and competitive year. Battlecode is so fun because of the community.

Lastly, we compiled a large amount of hard coded constants in our bot in the MagicConstants.java file. We wanted to tune these constants, but it was tough to tune them robustly given we could only play against bots we had, and we also wanted to tune against both cookies and strong rushes like **Kryptonite**'s. Additionally, we were constantly making big changes to our bot, so it felt like any tuning would soon be possibly irrelevant. Thus we ended up not being able to tune any of our constants robustly, but encourage future Battlecode teams to tune better!

11 Lessons from 6 years of Battlecode

This is the end of 6 years straight of Battlecode for Eli and probably Aaron, though we might be back if we go to grad school at some point. Our performance has ranged from 3 top 4 finishes to missing top 16 twice. Here are some tips for doing well, roughly in order of importance:

- **Don't be afraid to copy**: I think lots of teams want to come up with a cool new original strategy and demolish everyone with it. While that feeling must be amazing, in all our years we have done this exactly 0 times. The most tried and true strategy in Battlecode is to copy whatever better teams are doing. There is 0 shame in this, in our opinion. We were able to temporarily have a significantly better bot than Java by copying exactly what they were doing, then making some beneficial tweaks.

- **Prioritization, prioritization, prioritization**: Every bot in the final tournament has so many areas for improvement, even the top ones. **Java** (1st), **smite** (2nd), and us (4th) were laughing about some of the bugs and missing features in our final bots after the tournament. I really think the main difference between the better and worse teams is not coding skill or speed but prioritization. You only have 3 weeks to write your bot, so you

have to decide which features, and later bug fixes, are most important. This means that it is not a good strategy to go through a replay and note every single area of improvement, then go fix them all immediately. Rather, use a task management system (we used Asana but there are lots of options) to keep track of things to do sorted by priority. Always do the highest priority things first. If you are having trouble determining which things are the highest priority, think about how much the change will increase your win rate. It helps to have a lot of experience with strategy games. There is one exception to this as described in the next bullet.

- **Do general things first**: The first week and a half-ish of Battlecode should be mostly focused on writing code which won't change if your strategy changes. For example, navigation, resource-gathering, and communication. While this advice has been given many times in many postmortems, it is worth repeating because it is true.

- **Start early, but don't worry about placement in the Sprint Tournament**: This year we made the mistake of not doing much in the first 1.5 weeks, which put us at a disadvantage in general code as described above. I think we underestimated how much this mattered until **Java** told us about their trick they had come up with for going through sensed locations in a bytecode-efficient manner. As described above, spend this time doing general things! Placement in the sprint tournament is meaningless, and the meta will often shift due to balance changes and/or natural meta evolution.

- **Write good (readable and extensible) code**: Even though Battlecode is only 3 weeks, it's worth taking the time to write good code the first time. Re-writing and debugging badly written code is very costly, as illustrated by our late drone rewrite when we could have been spending time on more important things.

- **Debugging**: Use indicator lines when debugging location-based bugs, they are super helpful and save so much time compared to print statements. We were dumb enough to never use them until this year!

- **On the time between qualifying for finals and the final submission deadline**: In 2015 and 2016 when we qualified for finals, our qualifying bot was the one submitted for finals. In 2019 and 2020, we were presented with a new dilemma: we had about a day after we found out we qualified to improve our final submission. In 2019, we were far too cautious. We made very few changes as we didn't want to introduce any bugs, and this led to us falling from the top team to a deserved 4th place finish. This year, we tried to make too many changes, and didn't make any individual one of them super well. For example, our last-minute turtle worked well except that it was dead to any semblance of drone harassment. Our advice to teams which qualify for finals in future years is to choose one or two impactful improvements to your bot, and make them very well.

Scattershooting a lot of small, buggy improvements did not work out too well for us.

We hope these tips are useful to future competitors and that Battlecode continues to thrive for many years!