Battlecode 2025 Postmortem

Michael Hahn (skipiano)
confused

1 Introduction

I'm an undergraduate student at University of Waterloo studying computer science. This is my third year competing in Battlecode, previous years being 2020 (HS 2nd) and 2021 (9-12th). I have competed as team **confused** in all three years, although this my first time competing as an individual. Despite a late start, joining after **Sprint Tournament 1**, I managed to secure the top spot in the scrimmage ladder by the **Qualifier Tournament** and entered the **Final Tournament** as the 1st seed.

My final submission can be found here.

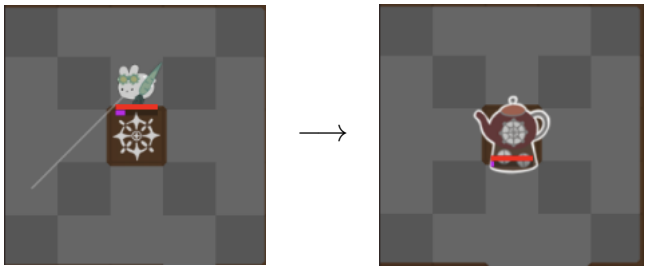
2 Game Overview

Battlecode 2025 centered around bunnies and paint. Teams could deploy both stationary towers and mobile units with special abilities:

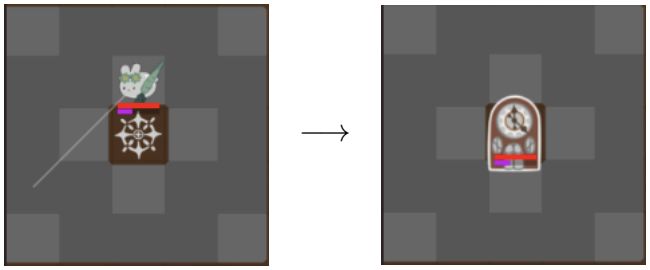
- **Paint Tower**: Generates paint, a primary resource.
- **Money Tower**: Produces chips, another crucial resource.
- **Defense Tower**: High damage output, generates chips upon attacking enemy units.
- **Soldier**: Versatile unit capable of painting cells and attacking enemy towers.
- **Splasher**: Performs AOE paint attacks, overwriting enemy paint in a 3x3 area.
- **Mopper**: Removes enemy paint and can perform a mop swing to decrease enemy paint in a 2x3 area.

Units had both HP and individual paint levels; chips were shared universally. Low paint levels (below 50%) penalized movement and action cooldown, while zero paint prevented actions and led to gradual HP loss.

Units could paint specific 5x5 patterns around ruins to build towers, costing chips. They lost paint when ending turns on neutral or enemy territory, with additional crowding penalties, encouraging strategic positioning.



(a) Paint tower pattern



(b) Money tower pattern

Figure 1: Different tower patterns

Special resource patterns (SRPs) could boost production when activated, requiring a specific 5x5 paint pattern and 50 rounds to take effect. Multiple SRPs could exist and overlap in a game.

Figure 2: Overlapping active and inactive SRP patterns

The overall win condition of the game was to paint over 70% of the entire map with your own paint.

3 Sprint Tournament 2

Because I missed Sprint Tournament 1 due to a late entry, I focused on rapidly developing my bot for Sprint Tournament 2. With only a few days before the deadline, I prioritized building a solid code infrastructure and implementing a rush strategy, which was considered strong in the current meta.

3.1 Infrastructure

Given the limited time, I decided to borrow both my own code from previous years and **camel_case**'s submission for 2024 with a slight modification to have individual classes for different units. This approach provided a robust object-oriented structure to build upon.

3.2 Communication

Despite communication being typically crucial in Battlecode, this year's limitations significantly reduced its importance. Towers and units could only communicate when close and connected by paint, or tower-to-tower within sqrt(80) units. The lack of direct robot-to-robot communication was particularly restrictive. By the end of the competition I barely used communication, which was the case for most top teams. Nevertheless, this initial focus helped me determine what information to store and how to process data within vision range each turn.

3.3 Pathfinding

Developing robust and performant pathfinding is challenging, and I experienced difficulties with it in previous years. This year, I adapted **camel_case**'s 2024 pathfinding solution. This algorithm combined unrolled Bellman-Ford with bug navigation, switching to the latter if Bellman-Ford failed to make progress for three consecutive turns.

I used IntelliJ's search and replace feature to adapt the unrolled Bellman-Ford without code generation. While I encourage teams to look at pathfinding solutions of previous top teams, I also wouldn't recommend blindly copying their implementation, as it can hinder future modifications. In my case, I needed to adjust weights to prioritize friendly paint and avoid enemy paint, optimizing the pathfinding for this year's unique terrain dynamics.

3.4 Soldiers

As the foundation of my rush strategy, soldiers emerged as the most crucial units due to their versatility in building towers, attacking enemy structures, constructing SRPs, and overall adaptability.

3.4.1 Exploring & Symmetry Checking

In Battlecode, maps are guaranteed one of three symmetries for fairness: horizontal, vertical, and rotational. Identifying the correct symmetry was vital for a rush strategy to locate enemy towers efficiently.

Symmetry could be deduced from wall and ruin locations, which remained constant throughout the game. I implemented a set of already checked locations to minimize bytecode usage, as this was a computationally intensive process. For horizontal and vertical symmetries, I only checked cells on the opposite side of the symmetry line and their corresponding pairs, reducing redundant checks. For rotational symmetry, I applied a similar concept, checking only cells in the opposite quadrant.

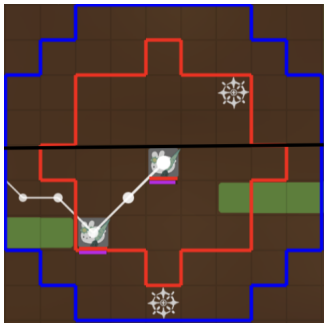


Figure 3: Sense nearby ruins and walls to check that vertical symmetry is broken

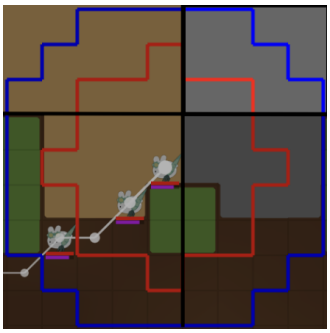


Figure 4: Only check the highlighted quadrant for rotational symmetry

Soldiers defaulted to assuming rotational symmetry unless proven otherwise, directing them towards the map center initially. This offered a strategic advantage, serving as the optimal location for determining the map's symmetry while also minimizing the distance to potential enemy tower locations.

3.4.2 Tainting Ruins

Inspired by **Asteroid**, one of the top rush teams, I adopted a strategy of partially painting nearby ruins without completing them. This prevented enemy soldiers from completing a

ruin into a tower unless they had a mopper or splasher to remove the paint. This cost-effective tactic prevented enemy soldiers from easily converting ruins into towers, effectively hindering their expansion.

Figure 5: With a single paint, the silver soldier is unable to build a tower.

3.4.3 Building Towers & SRPs

For tower and SRP construction, soldiers maintained a distance of 1 from the center, rotating to complete the pattern. They prioritized painting under themselves to conserve paint before filling the rest of the pattern. If there were two soldiers, they would both remain stationary and complete their respective sides of the pattern. Any additional soldiers would see the two soldiers and just move on to other tasks. This also happened if the soldier saw enemy paint and saw no moppers nearby, since they wouldn't be able to complete the ruin.

For building SRPs, since the SRPs were tilable, I simply used the modulo 4 trick that most teams were using at the time, where you assign all $(2 \bmod 4, 2 \bmod 4)$ cells as the center of the SRPs and check whether it is far away from non-tower ruins or walls, and fill it in. This allows a very simple heuristic to deciding the SRP centers that allow tilability.

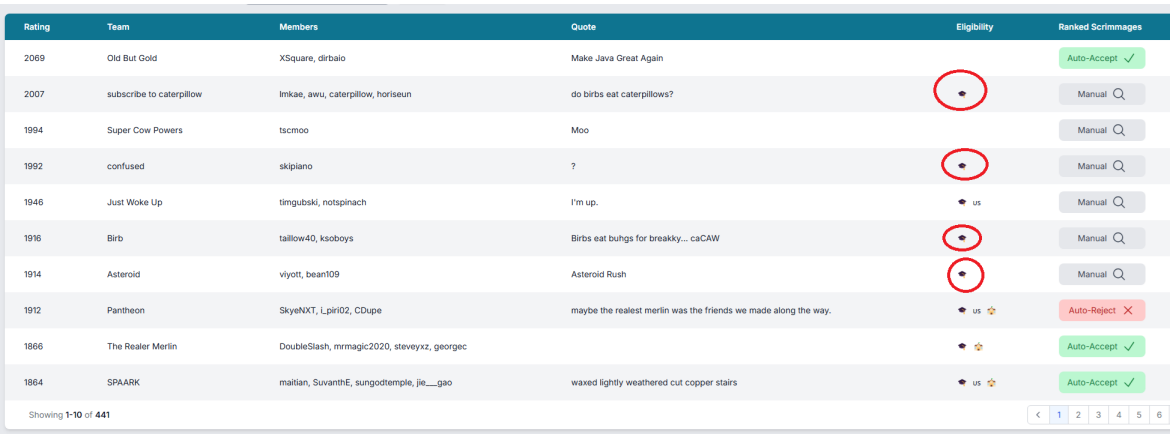
3.5 Results

With these soldier strategies implemented, I had only a day to develop basic mopper and splasher functionality. This rush-focused approach was able to secure a top 16 place in the tournament, primarily by relying on my rush strategy to destroy enemy towers.

4 Qualifier Tournament

I was in the international bracket, which only takes top 4 competitors and was very competitive this year.

A significant balance change by **Teh Devs** also drastically altered the meta: starting towers now began at level 2, granting them 500 additional health. This effectively nullified the rush



Rating	Team	Members	Quote	Eligibility	Ranked Scrimmages
2069	Old But Gold	XSquare, dirbaio	Make Java Great Again		Auto-Accept
2007	subscribe to caterpillow	lmkae, awu, caterpillow, horiseun	do birbs eat caterpillows?		Manual
1994	Super Cow Powers	tscmoo	Moo		Manual
1992	confused	skipiano	?		Manual
1946	Just Woke Up	timgubski, notspinach	I'm up.	us	Manual
1916	Birb	taillow40, ksoboys	Birbs eat buhgs for breakky... caCAW		Manual
1914	Asteroid	viyott, bean109	Asteroid Rush		Manual
1912	Pantheon	SkyeNXT, i_piri02, CDupe	maybe the realest merlin was the friends we made along the way.	us	Auto-Reject
1866	The Realer Merlin	DoubleSlash, mrmagic2020, steveyxz, georgec			Auto-Accept
1864	SPAARK	maitian, SuvanthE, sungodtemple, jie.__gao	waxed lightly weathered cut copper stairs	us	Auto-Accept

Showing 1-10 of 441

Figure 6: 4 out of top 5 eligible teams were international before qualifiers

strategy and forced me to quickly adapt my approach to avoid further rating losses.

Faced with this paradigm shift, I revisited my strategy by analyzing replays and identifying the game's core objectives.

4.1 Game Philosophy

After extensive replay analysis, I formulated two key principles:

4.1.1 Towers as top priority

Early tower establishment proved crucial for accelerated economy and map control. To optimize this:

- I prevented soldiers from painting under themselves before round 150, redirecting that paint towards tower construction and attacks. Painting under yourself costs 5 paint and only saves 1 paint per turn, so it wastes 4 paint per turn.
- Maintained aggression, as it either destroyed enemy towers or forced expensive mopper spawns, disrupting their economy.
- Delayed SRP construction until after establishing 6+ towers, as SRPs scale with tower count and require 50 rounds to activate.
- Focused on money towers initially, transitioning to a balance between paint and money towers after reaching 7+ towers.

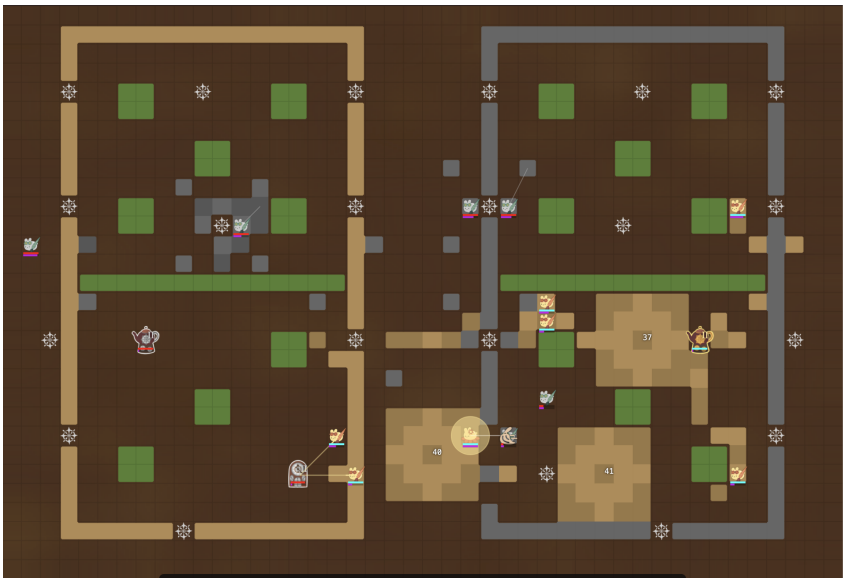


Figure 7: confused (silver): Only paint spent is building towers, tainting ruins, and attacking.

4.1.2 Minimizing unit idle time

Because towers were so important, efficient ruin exploration often determined the victor. To reduce unproductive wandering:

- Units defaulted to moving towards potential enemy tower locations based on symmetry predictions.
- Introduced randomness to ensure comprehensive map coverage.
- Implemented an idle timer, redirecting long-inactive units to the opposite side of the map center.

This ensured that most units would try to go towards the enemy towers, guaranteeing their value. Combined with the randomness, the entire map would eventually be explored.

4.2 Splasher Micro

If you simply take the paint value of splashing on to an empty field, you can paint 13 cells, which corresponds to 65 paint if we use 5 paint/attack for soldiers. This might seem worth it, but the true power of splashers comes from the ability to overwrite enemy paint, and splashers should hold paint until they have a really good opportunity to do so. Soldiers can already paint empty tiles - the problem is painting over enemy paint. Moppers are quite slow, so splashers are the fastest way to overwrite paint.

For my splasher micro, I looked at the 7x7 area around the splasher, and assigned each cell a value based on its paint value, and also whether it was close to a ruin. I would only decide

to splash if the sum of the values of the 3x3 around a possible attack location was over 50, the paint value of the splasher attack.

```
public void preprocess() throws GameActionException {  1 usage  Michael Hahn
    if (!rc.isActionReady() && !rc.isMovementReady()) return;
    computeSplashHeuristic = new int[7][7];
    MapInfo[] mapInfos = rc.senseNearbyMapInfos( radiusSquared: 18);
    for (MapInfo mapInfo : mapInfos) {
        int x = mapInfo.getMapLocation().x - rc.getLocation().x + 3;
        int y = mapInfo.getMapLocation().y - rc.getLocation().y + 3;
        if (x < 0 || x >= 7 || y < 0 || y >= 7) continue;
        PaintType p = mapInfo.getPaint();
        if (p.isAlly() || mapInfo.isWall()) computeSplashHeuristic[x][y] = -2;
        else if (p == PaintType.EMPTY) computeSplashHeuristic[x][y] = 1;
        else computeSplashHeuristic[x][y] = 10;
    }
```

Figure 8: Note the aggressiveness of the weighting of enemy paint vs. empty paint.

Splashers are also able to attack non-defense towers without getting hit themselves. It is still questionable whether it is worth spending 50 paint on a 100 damage attack, but with my philosophy of towers being important, I prioritized splashers attacking towers, and that could lead to a possible tower conversion.

Figure 9: Splashers can hit the tower indirectly by sitting just outside their range.

4.3 Tower Marking System

To prevent conflicting tower construction, I implemented a marking system for soldiers to agree on which tower to build.

Upon completing a tower, a soldier would mark it as a defense tower, regardless of its actual type. Then if this tower was destroyed, a soldier would come to it and see the mark and build a defense tower, and this implementation was able to build defense towers in previously attacked areas. Eventually I changed it so that this would only happen if there were still

Figure 10: Tower marks

enemy units around, as defense towers were being overbuilt. This change helped a lot when the center area was very contested.

4.4 Results

Having a good read on the game allowed me to enter the qualifiers with 1st seed. However, the qualifier did not go smoothly as planned. I almost lost to **geese geese go**, the 8th seed team (fellow UWaterloo team as well!) and barely won 3-2. I did get top 4 and qualified for the final tournament after beating **Birb** 3-0, but afterwards I lost to **Asteroid** 2-3 and **subscribe to caterpillow** 1-3, despite being 1st seed.

5 Final Tournament

I also watched the **US Qualifier Tournament** as well, and I noticed a very interesting strategy by **Gone Whalin**. They only built money towers, but they used the fact that all towers spawned with 500 paint to continuously destroy and rebuild their tower to convert 1000 chips to 500 paint. This 'tower flickering' allowed them to expand very quickly onto other ruins, and continuously used all of their resources very efficiently.

5.1 Tower Flickering

This strategy aligned perfectly with my established game philosophy, addressing additional issues I had previously:

- Focusing on money towers only allow very fast expansion onto new ruins, aligning with prioritization of towers.
- Converting excess chips into paint enabled efficient allocation of both resources.
- Enemy soldier aggression becomes less relevant as you can simply build a new tower on the destroyed tower, replenishing your paint supply.

- Tower flickering provides a constant source of paint through new money towers, becoming robust against losing your paint tower.

For tower flickering to be efficient, it's important to note that ideally, the paint pattern around the money tower needs to remain consistent. This means that we need to prevent SRPs from being built near money towers, which will decrease the amount of SRPs being built drastically. To combat this, I made my soldiers not build SRPs around 1/3 of the money towers.

Introducing tower flickering dramatically improved my bot's performance, resulting in a 70%+ win rate against my submission for qualifiers. This also meant that defense towers were no longer necessary to my strategy since I wanted to destroy my own towers, so I disabled them. Unlike **Gone Whalin**, I still spawned paint towers when chips exceeded 3000, but tower flickering (activated above 2500 chips) took precedence.

5.2 Results

Despite entering the final tournament as the top seed, scrimmages against top teams **Just Woke Up**, **Om Nom**, and **subscribe to caterpillow** were consistently close, barely maintaining a 50% win rate.

5 - 5	T	±0	Just Woke Up (±0)	Unranked
6 - 4	W	±0	immutable (±0)	Unranked
6 - 4	W	±0	Birb (±0)	Unranked
5 - 5	T	±0	Just Woke Up (±0)	Unranked
7 - 3	W	±0	lobster roll (±0)	Unranked
6 - 4	W	±0	Om Nom (±0)	Unranked
4 - 6	L	±0	Just Woke Up (±0)	Unranked
8 - 2	W	±0	Just Woke Up (±0)	Unranked
6 - 4	W	±0	Just Woke Up (±0)	Unranked
4 - 6	L	±0	Just Woke Up (±0)	Unranked
4 - 6	L	±0	Just Woke Up (±0)	Unranked
3 - 7	L	±0	Just Woke Up (±0)	Unranked
6 - 4	W	±0	Just Woke Up (±0)	Unranked

Figure 11: Very even scrimmage results against most finalist teams

I progressed through the winner's bracket with a clean 3-0 record against all opponents, including a surprising 3-0 victory over **Just Woke Up**. This led to a final showdown against **Just Woke Up**, who had emerged as the winner of the loser's bracket.

After losing the first set 1-3, the final set was tied 2-2, with the championship hinging on a single deciding game.

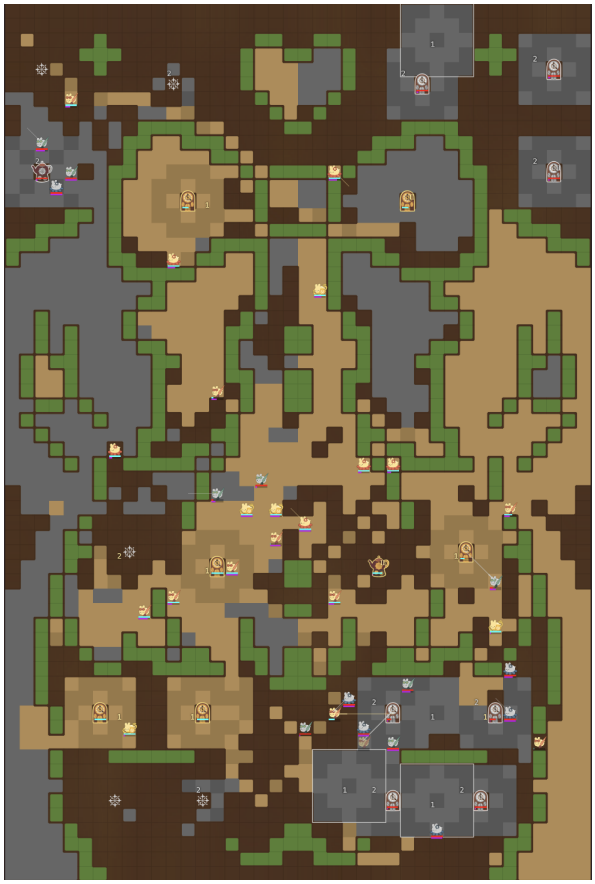


Figure 12: Just Woke Up (silver) vs. confused (gold) final game: corner vs center control

The moment I saw the map, I was not optimistic about my chances of winning. It was a large map with low ruin density, and also had a lot of ruins at the corner, which my bots were bad at exploring because they mostly defaulted towards the center. Indeed, while at first I killed some of **Just Woke Up**'s towers and was in the lead, I wasn't able find the corner ruins and translate the lead into a win, resulting in a win for **Just Woke Up** and 2nd place for me. Congratulations to them! Their unique defense tower spawning criteria came in clutch for securing crucial rounds.

6 Aftermath

I talked to a lot of the other finalists during the final tournament about their strategy. These conversations highlighted the relative simplicity of my bot compared to some of the more complex implementations of the other teams.

One of the main messages I wanted to convey through this postmortem was that a deep understanding of the game's fundamentals and thorough analysis of gameplay can lead to

strong results, even without relying heavily on advanced algorithms or extensive bytecode optimization. I didn't really do any bytecode optimization and I wouldn't say any of the algorithms I had were very hard to implement. My approach involved extensive replay analysis, borrowing strategies that looked good from other teams, and a focus on aligning all development with a well-defined core game philosophy, obtained from watching all those replays. Special shoutout to **Old But Gold**, **subscribe to caterpillow**, **Asteroid**, and **Gone Whalin** for being my primary source of effective strategies this year.

Obviously doing this in practice is harder than it sounds, but I believe that replay analysis and prioritization of high-impact features are essential skills for success in Battlecode. Especially as an individual competitor with limited development time, it was crucial for me to ensure that the majority of my code directly contributed to improving my bot's performance.

While this was my second time as a finalist (after 2021), it was my first time competing in person at MIT, as the 2021 competition was held remotely due to the pandemic. Thank you to **Teh Devs** and the sponsors for organizing the competition and the career fair, and it was amazing talking to all the other finalists as well. I hope this postmortem inspires newcomers to participate in Battlecode and provides valuable insights for returning competitors. I'm looking forward to (hopefully) seeing all the new and old faces at next year's Battlecode at MIT!