The Blades Cambridge Battlecode 2026 Postmortem

By bluecrab
May 22, 2026

I. Introduction

My team is The Blades. My team placed top 32 in the International Qualifiers. Previously, I participated in MIT Battlecode 2026, placing in the top 60 on ladder. This postmortem will go over how I approached this Battlecode and what strategies I used. My gitHub repository can be seen here.

II. Game Overview

Background

The year is 2076. A crystalline ore called **axionite** — a room-temperature superconductor — has been discovered on Titan, Saturn's largest moon. At least six corporations have deployed autonomous extraction fleets to Titan's surface. Titan is lethal to humans: −179°C, a nitrogen-methane atmosphere, and a 76-minute communication delay to Earth. All operations are carried out by robots. You write the software that controls your fleet: mining ore, refining axionite, and outcompeting the enemy.

Objective

Collect resources and **destroy the enemy's core**. To do this, you must find ore deposits, build harvesters, deliver resources back to the core using conveyors, and expand your territory.

(from https://docs.battlecode.cam/spec/overview)

Gameplay

There are two teams placed on a map. Both teams must fight over titanium and axionite to win. Each team starts with a core. If your core is destroyed the other team wins. The core can create builder bots. These builder bots can move and place several different buildings to defend, attack, or collect resources. These buildings fall into four (unofficial) categories: transportation, resource collection, turrets, and defense. Transportation buildings primarily are used to transport things (shockingly), such as builders or resources. Some transportation buildings: roads, conveyors, and splitters. Resource collection buildings collect/refine resources (I'm very creative), these buildings being harvesters and foundries. Turrets attack enemy buildings and builders and need ammo to attack. The turrets are the sentinel and gunner. Breaches are too powerful to be considered turrets, or be categorized at all. In fact, they were so powerful, teams collectively decided to avoid using breaches to ensure fairness. Defense buildings, launchers and barriers, deny space from enemy builders and buildings. Using these four building types various interesting strategies can be developed to optimize titanium/axionite collection or rip through enemy defenses to destroy supply lines and the core.

III. Intelligence Free*

For this competition I did not use artificial intelligence (AI) for most of my programming. The only time I did use it, was for translating files from Java to Python. While using AI makes it easier to implement ideas, I prefer writing code by hand and it allows me to better understand the code my bot is using. However, if you are a beginner in competitions like, AI can be a good tool to quickly get you up to speed.

IV. Set Up

Since I previously participated in MIT Battlecode, I decided to reuse the code I had written for it. Both competitions have robots moving through a tile based map, collecting resources, and communicating, so my code for pathing, encoding/decoding comms, and general map utilities would still work. However, MIT Battlecode is in Java, while Cambridge Battlecode is in Python. I did not feel like rewriting my code, so I threw it into an AI model and told it to convert the Java code to Python. While AI got me most of the way there, I still had to switch from using the MIT Battlecode library to the Cambridge Battlecode library. This process took about two days. For beginners, it is a good idea to use pre-existing code for things that work in all Battlecodes, as it allows you to quickly get to implementing strategy (often referred to as macro), which is my favorite part of the competition.

V. Tools

During the competition, I created a few tools to help test my bot. The main tool I created was a script that ran matches between two local bots on every map twice, once for each side. Running two matches on every map took a long time (sometimes over an hour), especially as more maps were added. To speed up this process I sometimes only ran games on some maps; however, taking advantage of multithreading led to much faster speeds, allowing me to run more matches. Since running the matches in mass gave too much information to easily read manually, I made a script that summarized the data, giving winrate and win types (e.g. core destroy or resource victory). Although testing against your own bots is not considered optimal, I did not use any scripts to run unrated matches against other teams, such as Blue Dragon's script. Using this script would probably have improved my bot, but I found watching ladder and occasional unrated matches sufficient for testing (and I didn't want to set the servers on fire).

VI. Pathfinding

I reused “my” pathfinding algorithms from MIT Battlecode. My bot used two pathfinding algorithms: BugNav and A*.

BugNav

The BugNav algorithm was taken from XSquare’s 2024 MIT Battlecode bot and converted to Python (XSquare’s repository). I still had to make some changes specific to this Battlecode. Some of these changes were updating what made a tile passable and building roads before moving onto a tile. This pathfinding algorithm started as my main way for bots to move, however, the high compute time limit allowed A* to become the primary pathfinding algorithm.

A*

The A* algorithm was copied from my 2026 MIT Battlecode bot and converted to Python. This A* could persist between turns, so the bot timing out was not a problem. I initially only used A* for pathing conveyors. Since conveyors cost a significant amount of titanium to build, I determined that finding the most efficient path for them was most important, even if pathing took multiple turns of compute time. To further improve conveyor efficiency, I implemented bridges. Bridges can transport resources to any tile within 3 tiles. To use bridges, I allowed my A* to visit tiles up to a distance of 3. This solution massively increased the compute time required, but I could not find a much better solution that still found the most efficient path. Also, since bridges cost more to build, I punished using bridges in the heuristic function to avoid bridge spam. The heuristic also used Manhattan distance for a base value and punished overlapping conveyor paths and building near enemy turrets. Later, once I realized that the time limit was very high, I used A* for bot pathing, I referred to my bot versions using A* for pathing as the Bat’leth series.

VII. Communication

This Battlecode had very difficult communication. My bots mostly relied on what they could see to determine what needed to be done. However, markers did get some use by my bots. The most important use of markers in my bot was the “library.” The library was a marker that stored information deemed necessary for the bots to function. This

marker was maintained by the core and contained the role of the previously spawned robot (for role assignment on spawn), the map symmetry type, and (only in older versions) the side of the core the foundry would be on. At one point the library was a set of three markers, each containing a value. To condense this information, I created an information encode/decoder for the library. The marker would just have an identifier at the start and a sequence of numbers representing the data. Another use for markers was using them to denote a spot where a building needed to be placed, but could not be on that turn. The marker would just store an identifier, the round number, and the placer's ID. This system allowed the builder to reserve that spot, but if they died or ran off for some reason, other builders could use the spot.

VIII. Strategic Philosophy

The main idea behind my strategy was to get a strong economy and repel enemy attacks. I did not prioritize offense. I thought that if I could mine enough resources, my enemies would not be able to overcome my defense. Then, the enemy would be crippled if they attacked me. Once my bot got a strong enough economy, it would send a small attack to sabotage enemy supply lines or destroy the enemy core. Each bot would be created with a goal in mind, or role. These roles were not strict, as some bots would multitask and often the bot's role would switch once the goal was complete. This strategy worked well, but I underestimated my opponents' abilities to destroy my defense.

IX. Economic Strategy

The economy is the backbone of any strategy, so I spent most of my time working on it, especially since it was the focus of my strategy. There are three parts of my economic strategy: exploration, titanium and axionite collection, and axionite refining.

Exploration

Exploration was needed to find where ore deposits were and what tiles could be used to place conveyors. My exploration started out simple. The bot would get a random tile, and move towards it. Once the bot could see the tile it would get a new random one. Along the way, the bot would store the tiles information it saw to a 2D array, called the

map. The information stored was the tile's environment, building, and the team of the building if there was one. Using this information, the bots would have near perfect information when pathfinding. The bot would also store the mines it saw in one of two arrays, depending on whether it was titanium or axionite. Also, the bot would use this information to determine how the map was symmetric. Eventually, I made the exploration algorithm only choose unexplored tiles.

Titanium and Axionite Harvesting

If the bot was out to collect titanium, it would get the closest unharvested titanium mine to the core and harvest it. If the bot was out to collect axionite, it would get the closest unharvested axionite mine to the core and harvest it. For the conveyors, I found that using bridges to input resources into the core was more effective. Bridges were more effective, since they create more entrance points into the core. Using conveyors, there are only 12, but with bridges, every tile within three tiles of a core tile could act as an entrance point. To incentivise using bridges to input resources, I had the cost of using bridges in the A* heuristic linearly scale based on distance from the core.

Axionite Refining

To refine Axionite, I used a simple method, putting splitters around the core, funneling into foundries. This method allowed the axionite harvesters to syphon some of the titanium to refine the axionite and stop the foundry from clogging.

X. Defensive Strategy

Although defense was an important part of my strategy, the implementation was very simple. The bots that collected titanium would place splitters near the core to feed sentinels on the other tiles. These splitters were the same as the ones used for axionite. This defense was the only way I could repel early attacks. Later into the game, a defense bot would be created. This bot would place turrets and launchers around mines. This defense bot prevented the enemy from destroying supply lines. Another part of the defense was healing. Every bot would heal opportunistically. One role, maintainers, would path to heal the most important building in its vision. These maintainers would be built when under attack. The maintainers would also repair destroyed conveyors and

redirect conveyors from enemy turrets (since I implemented this functionality the same day as international qualifiers, I do not know if it works that well).

XI. Offensive Strategy

Offense was not important to my strategy, but I still included it in my bot. Once I knew the enemy core's location, using the map symmetry, I would create a few attacking bots. These bots would look for a mine close enough to the enemy core for a sentinel to shoot it. If the bots found a mine that fit this criteria, they would place sentinels around it. If not, they would hijack an enemy conveyor and place either a gunner or sentinel. Gunners were much more powerful than sentinels in terms of damage output, but had a drastically shorter range. The bot would place a gunner if the enemy core was within its limited range. Although this role was the only intentionally offensive role, the defense role would often act as a stronger offense, slowly creeping closer to the enemy with each mine defended. As a result, the enemy would lose their supply lines.

XII. Conclusion

Overall, Cambridge Battlecode was incredibly fun and I would recommend it to anyone. It is not too hard to get into as a beginner and has lots of strategic depth that I was unable to get into fully as I wrote this in three hours. I look forward to participating in and improving in the next Battlecode.