Battlecode 2025: Chromatic Conflict

Background

The bread and food of yore have begun to run out, forcing robot society to adapt. Gone are the jovial ducks, replaced by steampunk robot bunnies who have converted their need for nutrients into a reliance on paint. These bunnies have become territorial, forming clans and defense formations to protect the resource that keeps them running.

For the past two centuries, these bunnies have stayed within their own territory, but clans have begun to degrade their environment and need to start branching out. Will these clans be able to expand their territory and generate enough paint to protect their families? Or will they stray too close to other clans and be wiped out in conflict?

Objective

In Battlecode: Chromatic Conflict, your goal is to paint the map. The first team to paint more than 70% of paintable squares on the map wins the game. Teams can also win by destroying all the enemy team's robots and towers.

Tie Breakers

If neither team has painted more than 70% of the **paintable squares** on the map after 2000 rounds, the game will end immediately. The following tiebreakers are applied, from top to bottom, to determine the winning team.

- Area painted on map
- Number of allied towers currently alive
- Total amount of money
- Sum of paint across all robots and towers
- Number of allied robots currently alive
- A uniformly random team will be selected.

Good luck!

Map Overview

Each Battlecode game will be played on a map. The map is a discrete 2-dimensional rectangular grid of size ranging between 20×20 and 60×60 inclusive. The bottom-left corner of the map will have coordinates (0, 0); coordinates increase East (right) and North (up). Coordinates on the map are represented as MapLocation objects holding the x and y coordinates of the location.

To prevent maps from favoring one player over another, it is guaranteed that the world is symmetric either by rotation or reflection.

Passability and Visibility

Robots can always sense map features and other robots up to sqrt(20) units away. Units cannot move over towers or other robots.

Maps may have walls, which robots cannot pass through or paint on. No more than 20% of a map will be walls. Walls cannot be removed or built by competitors. The same goes for ruinis.

Towers and Ruins

There are three types of towers—money-generating towers, paint-generating towers, and defense towers. Towers also serve as spawn zones for each team and can attack units from the opposing team.

To build a tower, a team must paint a specific shape onto a ruin (see towers section), a 5x5 zone pre-placed on the map. Each game starts with some number of ruins, and the ruins cannot be moved or destroyed. The centers of ruins will be at least 5 units apart, and there will never be any walls in the 5x5 zone around the center of a ruin. Each team starts with 2 towers, 1 paint and 1 money tower. Ruins cannot be seen by robots or towers unless they are within the vision radius.

Special Resource Pattern (5x5 shape)

Painting special patterns on the map can boost a team's resource production. These are known as special resource patterns (SRPs). For each active instance of this pattern that is painted on the map, all allied mining towers will mine 3 more resources per turn. Robots can mark the map with this pattern using the mark_resource_pattern() function. Robots must call the complete_resource_pattern() function once the pattern is painted on the map for it to start boosting resource production. Calling this method costs 200 chips. Once a pattern is created, it must exist without any of its paint being disrupted for 50 rounds before becoming active. These patterns are fixed across all maps and games.

Resources

Paint

Paint is the primary resource of the game. It is used to mark territory and count scores. Each robot and tower holds paint individually, up to a certain capacity determined by their

type. Robots face increased cooldowns or an inability to act at low paint, but they can refill their paint from moppers or by withdrawing from paint-generating towers. Each robot type (described below) has a distinct attack that will either consume paint to place it on a square or remove the opponent's paint from a tile. Robots can choose between using a team's primary and secondary color to paint a tile. These two colors are treated identically and are only distinguished when checking for completion of a paint pattern.

Chips

Chips can be used to build units & towers. Chips are generated each turn by money generating towers and can be used by any allied robot once generated. Each team starts the game with 2500 chips.

Units

The Battlecode world contains many kinds of robots. All robots can perform actions such as moving, sensing, and communicating with each other. In each battle, your robots will face one opposing enemy team.

The game is turn-based and divided into rounds. In each round, every robot gets a turn in which it gets a chance to run code and take actions. Code that a robot runs costs bytecodes, a measure of computational resources. A robot only has a predetermined number of bytecodes available per turn, after which the robot's turn is immediately ended, and computations are resumed on its next turn. If your robot has finished its turn, it should call yield_turn() to wait for the next turn to begin.

All robots have a certain amount of HP (also known as hit points, health, life, or such). When a robot's HP reaches zero, the robot is immediately removed from the game.

Robots are assigned unique random IDs no smaller than 10,000.

Robots interact with only their nearby surroundings through sensing, moving, and special abilities. Each robot runs an independent copy of your code. Robots will be unable to share static variables (they will each have their own copy), because they are run in separate JVMs.

Two or more robots may not be on the same square. When their movement cooldown goes below 10, robots can move onto any of the 8 neighboring squares.

All robots also have a stash of paint, allowing them to store paint up to their capacity. Robots face increased movement and action cooldowns the lower their stash of paint is. Robots with a stash of paint that is X% full, where X is less than 50, face (100-2X)%

increased cooldowns. Once a robot's stash of paint is entirely depleted it is unable to move or perform actions other than self-destructing until it receives paint, and it will lose 20 health/turn. Robots spawn with their paint stash 100% full and can have it refilled at a tower or by a mopper (described below). Robots lose 1 paint when ending their turn on neutral territory, 2 paint when ending their turn on enemy territory, and no paint to end their turn on allied territory. Robots also face an additional paint penalty each turn equal to the number of adjacent allied bots (bots in the 8 squares around the robot). This penalty is doubled while in enemy territory.

Soldier

Soldiers have 250 health and have a maximum paint stash of 200. They cost 200 paint and 250 chips to produce. They can attack a tile, painting it if it is empty or has allied paint on it as well as dealing 50 damage to enemy towers (not robots). This attack costs 5 paint, has an action cooldown of 10, and has a radius of 3.

Splashers

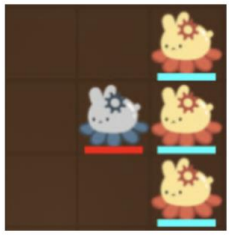
Splashers have 150 health and a maximum paint stash of 300, costing 300 paint and 400 chips to produce. Their attack costs 50 paint and has an action cooldown of 50, but it affects a circular area within a radius of 2 from its center point. Within a radius of sqrt(2) of the center location, it can also paint over enemy paint. It targets a center location up to 2 units away. This attack deals 100 damage to enemy towers (not robots) and paints all empty/allied tiles.

Mopper

Moppers have 50 health and a maximum paint stash of 100, costing 100 paint and 300 chips to produce. They can choose to mop a tile, removing enemy paint on the tile. If an enemy robot is on the tile, they lose 10 paint which is converted to 5 paint for the mopper. This action incurs a 30-action cooldown.

Moppers can also transfer a specified amount of paint from their own paint stash to allied robots or towers at a 10-action cooldown. These actions have a radius of sqrt (2) units. However, moppers face double paint penalties from moving and staying on enemy terrain. Moppers can also swing their mop in one of the 4 cardinal directions, removing 5 paint each from adjacent 3 enemy robots in a line (see diagram). Mop swinging has an action cooldown of 20. This attack applies twice in the direction of attack, which means it applies once one step in the direction and again two steps in the same direction. This way, there can be a total of 6 potential targets to be mopped.

Examples of Mop Sweeps



Silver swings east (right). All three gold bots lose paint.

Silver swings north (up). All three gold bots lose paint.

Silver swings south (down). The bottom gold bot loses paint.

Silver swings west (left). No gold bots lose paint.

Robot Statistics Table

Most of the robot statistics are summarized in this table.

Name	Health	Paint Capacity	Paint Cost	Money Cost	Attack Cost	Attack Radius	Attack Damage	Attack Cooldown
Soldier	250	200	200	250	5	3	50 health to towers	10

Splasher	150	300	300	400	50	2	100 health to towers	50
Mopper	50	100	100	300	0	Sqrt(2)	-10 paint to robots	30

Towers

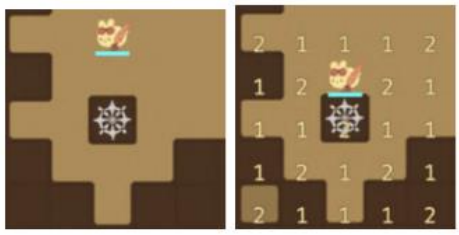
There are three types of towers: money-generating, paint-generating, and defense. By default, every team will start with a money-generating tower and a paint-generating tower, both upgraded to level 2.

Money-generating tower

Paint-generating tower

Defense tower

All towers are built with the same procedure. Towers can only be built on ruins, after a certain 5x5 pattern of paint is placed around the ruin. Call mark_tower_pattern() to automatically place markers on the 5x5 area surrounding a ruin, designating where to place paint of the primary color. This method costs 25 paint.



Effect of calling mark_tower_pattern()

Once the shape is painted, a robot can call complete_tower_pattern() to spawn the designated tower at the ruin location. This tower will remain alive until its health reaches 0, even if the paint pattern around the ruin is later removed. Note that a tower pattern can be completed without needing to first call mark_tower_pattern(), if the pattern of primary and secondary paint is matched.

Generated towers will occupy a 1x1 block in the center of the spawn shape. Different shapes correspond to different types of towers with different ranges, attack rates, mining rates, and health (as detailed in the table below). Towers will always be built at level 1 and can be upgraded to higher levels using the upgrade_tower() method within a radius of sqrt(2) units. Neither team may have more than 25 towers at any given time. Each tower can store 1000 paint. Each defense tower that is alive increases the single-block attack strength of all allied towers by +5/+7/+9 based on its level respectively. This buff does not affect the AoE attack of allied towers.

Name	Build / Upgrade Cost	Attack Range	Single-Block Attack Strength	AoE Attack Strength	Base Mining Rate	Health
Money Lv 1	1000 chips	3	20	10	20 chips/turn	1000
Lv 2	2500 chips	3	20	10	30 chips/turn	1500
Lv 3	5000	3	20	10	40 chips/turn	2000
Paint Lv 1	1000 chips	3	20	10	5 paint/turn	1000
Lv 2	2500 chips	3	20	10	10 paint/turn	1500
Lv 3	5000 chips	3	20	10	15 paint/turn	2000
Defense Lv 1	1000 chips	4	40	20	20 chips/attack *	2000
Lv 2	2500 chips	4	50	25	30 chips/attack *	2500
Lv 3	5000 chips	4	60	35	40 chips/attack *	3000

* Must successfully hit at least one robot with either AoE or single attack. Multiple robots hit with an attack do not stack the chip gain, but AoE and single attack will both provide a bonus. This stat is not affected by SRPs.

Attack

To defend themselves, towers are able to attack units. Towers have two types of attacks—a single-block attack that attacks an enemy unit on one block and a ranged area of effect (AoE) attack that does damage to all enemy units within its range. Every turn, a tower can perform one single-block attack and one AoE attack.

Mining

Every turn, money and paint towers will passively mine resources. Paint is stored in the tower and can be removed by allied robots. Money is available to all allied units.

Every tower starts with 500 paint to spawn different types of units. It can spawn a bot within 2 blocks of the tower. If a tower runs out of paint it will not be able to spawn units until it is refilled or regenerates paint on its own. A tower can be refilled by the paint collected by moppers.

Communication

Robots can only see their immediate surroundings and are independently controlled by copies of your code, making coordination very challenging. You will be unable to share any variables between them; note that even static variables will not be shared, as each robot will receive its own copy.

Communication is done through two methods:

1. Messages, which can contain any information, but are restricted to 4 bytes.
2. Markers, which will state which color a space should be repainted to and if it is visible to all allied robots. Markers do not have any other effect on the game.

Messages

The robots and towers who are connected by paint can talk, but the robot who is not on paint cannot talk to the tower.

A robot is “in range” of a tower if it is within sqrt(20) of the tower and connected to the tower by paint (i.e., it can move to the tower by only walking on blocks with its own paint). Any robot in range of a tower can send a message to the tower and vice versa.

A robot or tower will store all messages it has received in the last 5 turns in its buffer, after which, they will be deleted. Robots can send 1 message per turn; towers can send 20 messages per turn. A message consists of one 32-bit integer, as well as the round number that the message was sent in and the id of the robot that sent the message.

Towers have the additional ability to broadcast messages to other nearby towers within a sqrt(80) range. The towers can utilize this ability to extend their long-range communication. Towers do not need to be connected to each other by paint to perform this ability, and performing this ability once will count one toward the message limit.

Markers

Markers are information that can be placed/erased on the map by robots. They can either indicate “Secondary color” or “primary color.” They are only visible to allied robots and remain there until either erased or another marker is placed on the spot by an ally. Markers can be placed manually on a square within sqrt(2) units of the square the robot currently occupies, or with the mark_tower_pattern()/mark_resource_pattern() function, which marks the 5x5 area around a square within sqrt(2) of the robot with the appropriate pattern. Markers can only be placed on tiles that are paintable.

Actions and Cooldowns

Robots perform actions to interact with the game world, and some cannot be performed multiple times in a single turn or in a short period of time. These actions are:

Attacking/Painting

Attacking and painting are performed with the same method. Attacking can only be done if the robot's action cooldown is less than 10 (can check with the is_action_ready() method). Attacking can be performed by calling the attack() method. This targets a particular square, dealing damage or painting according to the robot's type as described in the Robots section. This increases the robot's action cooldown as determined by the robot type.

Moving

Moving can only be done if the robot's movement cooldown is less than 10 (can check with the is_movement_ready() method) and if the ending square is unoccupied and passable. Moving can be performed by calling the move() method. This moves the robot in the specified direction. This increases the robot's movement cooldown by 10. If you wish to check whether a move is valid, you can use the can_move() method.

Marking

All robots can mark their current tile or remove an allied mark from their current tile at the cost of 1 paint, incurring no action cooldown. This can be done by calling the mark() or remove_mark() methods. These marks do not affect territory ownership (i.e. marking a tile does not set it as allied territory) but can be sensed by ally robots as part of the sense_nearby_map_infos() method. The mark_tower_pattern()/mark_resource_pattern() function uses 25 paint and marks a 5 by 5 space with the tower pattern. Marks can only be placed on squares that are paintable (i.e. have no wall or ruins on it).

Transferring

Moppers can transfer paint if the robot's action cooldown is less than 10 (can check with the is_action_ready() method) and can be performed by calling the transfer_paint() method. Moppers must be within sqrt(2) units of the location to transfer resources. This transfers the specified amount of paint to the target and removes it from the mopper's stash. If the given quantity is negative, the robot instead removes the specified paint. This increases the robot's action cooldown as specified in the robot details.

Withdrawing

All robots can withdraw paint from allied towers to replenish their paint stash. This requires an action cooldown of less than 10 and will increase their action cooldown by 10. It has an action radius of sqrt(2) units and can be called with the transfer_paint() method, using a negative value to denote the amount to withdraw.

Spawning

Towers can spawn robots, but only when their action cooldown is less than 10 (can check with the is_action_ready() method) and can be performed by calling the build_robot() method. This deducts the cost of the robot from the tower's stockpile, then creates one robot of the specified type in the specified square. It also increases the tower's action cooldown by 10.

Communicating

Robots and Towers can use send_message() to send a message to an allied tower or robot within their range. A bot can use read_messages() to read any new messages sent within the past 5 rounds from their message buffer. These methods do not increase action cooldown. Two units must be on squares connected by ally paint to communicate with each other.

Disintegrating

Any robot can call the disintegrate() method. This immediately destroys the robot that calls it.

After every turn, the movement and action cooldowns of all robots are decreased.

Bytecode Limits

Robots are also very limited in the amount of computation they are allowed to perform per turn. Bytecodes are a convenient measure of computation in languages like Java, where one Java bytecode corresponds roughly to one basic operation such as "subtract" or "get field", and a single line of code generally contains several bytecodes. Because bytecodes are a feature of the compiled code itself, the same program will always compile to the same bytecodes and thus take the same amount of computation on the same inputs. This is great, because it allows us to avoid using time as a measure of computation, which

leads to problems such as nondeterminism. With bytecode cutoffs, re-running the same match between the same bots produces the same results - a feature you will find very useful for debugging.

Every round each robot sequentially takes its turn. If a robot attempts to exceed its bytecode limit (usually unexpectedly, if you have too big of a loop or something), its computation will be paused and then resumed at exactly that point next turn. The code will resume running just fine, but this can cause problems if, for example, you check if a tile is empty, then the robot is cut off, and the others take their turns, and then you attempt to move into a now-occupied tile. Instead, use the yield_turn() function to end a robot's turn. This will pause computation where you choose and resume on the next line next turn.

The bytecode limit for all robots is **17500**, and the bytecode limit for all towers is **20000**.

Some standard functions such as the math library and sensing functions have fixed bytecode costs, available here. More details can be found in the docs/BYTECODE.md file in your team repo.