AIC2023: Home Run

1 Lore

It was such an intense match. Thousands of people were gathered to watch the finals between the two best MLB (Major League Baseball) teams. It could have been anyone's game, but suddenly it started raining like it hadn't rained for years, and it didn't look like it was about to stop any time soon. The match was cancelled, determined as a draw, and everyone was sent home.

However, the baseball teams were not willing to give up their purpose because of some rain. Since the referees were not interested in the slightest on getting a single drop of water in their shirts, they decided to create some new rules to determine the winner.

2 The Game

In this game, each team can recruit new players, score points, gather reputation, buy new baseballs, and eliminate opponents by throwing them into the water or making them collide. If a team gets 2000 points, they win the game. If no team wins in 1500 rounds, the winner team is chosen according to the following criteria, in order:

1. The team with more points.
2. The team whose reputation plus the value of all of its units is the highest.
3. Randomly.

3 Economy

There are two types of resources in the game: **points** and **reputation**. Points are necessary to win the game and are the first tiebreaker whenever the game reaches the maximum number of rounds. Reputation is used to recruit new units and to buy new baseballs.

At the start of each round, all teams get 2 reputation. Moreover, teams can also get additional points or reputation whenever a pitcher starts its turn on top of a **Base** or **Stadium**, respectively. Bases give 1 point and stadiums give 1 reputation. However, a base cannot give points more than once on a single round for a given team. Each team starts with 0 points and 100 reputation.

Since the game gathers more attention the longer it goes, both points and reputation suffer from inflation. This makes that the points and reputation obtained from all sources is increased by 0.001 for every passed round. For instance, if a pitcher starts its turn on top of a base at round 1000, it gets 2 points instead of 1 (note that the increase is not exponential).

4 Map

The map consists of a grid of dimensions between 20×20 and 60×60. Each unit occupies exactly one tile and all the tiles of the map are accessible by every unit except if there is water, another unit, or a baseball. Each tile that is not water may also contain a Base or a Stadium.

Units have finite vision range. Moreover, the vision between units is not shared. This means that all objects on the map that are outside of a given unit's vision range cannot be queried during that unit's turn.

Each team starts with exactly one HQ and it is guaranteed that all maps are symmetric. This symmetry may be horizontal, vertical or rotational. All tiles of a given map will have a fixed offset between 0 and 1000 (more precisely, the tile that would be $(0, 0)$ is marked as $(\text{offset}_x, \text{offset}_y)$ instead). This way units cannot guess the map edges without exploring.

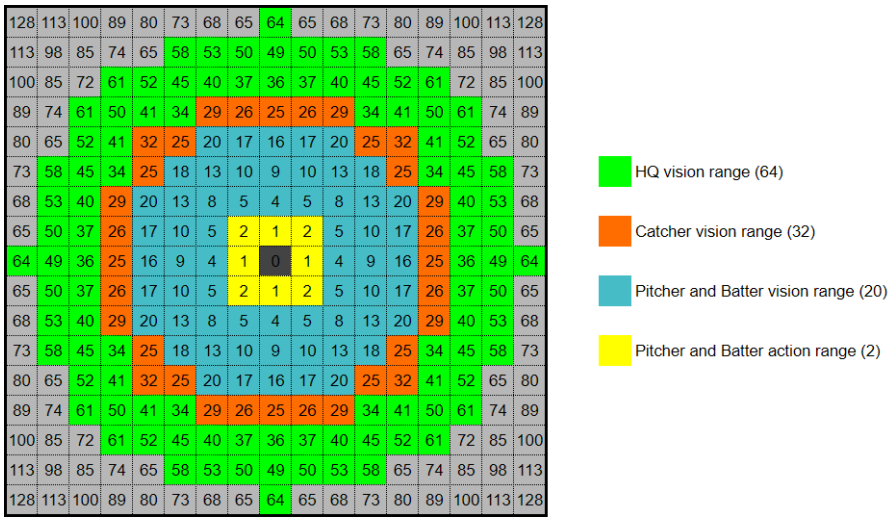
5 Units and Structures

Each unit has a unique identifying number (ID) chosen randomly between 1 and 10.000.

There are four types of units: HQ, Pitchers, Batters and Catchers. The parameters of each of these types are as indicated in the following table. All distances are shown in squared units. For instance, a unit with 12 vision range at $(0, 0)$ can query which object is at $(2, 1)$ since $2^2 + 1^2 \leq 12$, but it cannot query which object is at $(2, 3)$ since $2^2 + 3^2 > 12$. A $(*)$ indicates that this unit follows a special mechanic (see Section 7).

	HQ	Pitcher	Batter	Catcher
Reputation Cost	–	50	60	40
Movement Range	0	2	2	2
Vision Range	64	20	20	32
Movement Cooldown	–	1.5	1.5	1
Action Cooldown	$0^{(*)}$	$2^{(*)}$	$1^{(*)}$	$-^{(*)}$

This image displays all unit ranges present in the game:



6 Movement, Actions and Cooldowns

Each unit has a movement and an action cooldown. They can only move, or act whenever their respective cooldown is strictly less than 1. In particular, if all of its cooldowns are strictly less than 1, the unit can act and move on the same turn.

Every time that the unit acts or moves, it adds cooldown to its respective value. The amount added is the one given in the table in Section 5. If the unit moves in a diagonal direction, the movement cooldown added is multiplied by 1.4142, which is approximately $\sqrt{2}$. At the beginning of every turn, all cooldowns decrease by 1.

7 Special Mechanics

In this edition, all units have special mechanics:

HQ: Can recruit new units and construct baseballs on adjacent locations[1].

Pitcher: If it begins its turn on a base or a stadium, its team gets one point or one reputation, respectively. Moreover, pitchers can grab adjacent baseballs, carry them, and drop them to an empty adjacent tile.

Batter: Can bat baseballs or other non-HQ units one to three tiles away. Whenever an object A is batted, it moves following the direction that goes from the batter to A until it collides with another object B. The aftermath of the collision between A and B depends on both objects, and it is as follows.

- **Baseball collides with Pitcher or Batter:** The Pitcher or Batter is eliminated, and the ball continues its path.
- **Baseball collides with Catcher:** The ball is stopped by the catcher. The catcher is not eliminated.
- **Baseball collides with another Baseball:** Both baseballs are destroyed
- **Baseball collides with HQ:** The opponent of the affected HQ is awarded with 100 points and the baseball is destroyed.
- **Pitcher, Batter or Catcher collides with Baseball or HQ:** The Pitcher, Batter or Catcher is eliminated.
- **Pitcher, Batter or Catcher collides with Pitcher or Batter:** Both units are eliminated.
- **Pitcher, Batter or Catcher collides with Catcher:** The unit is stopped by the catcher. No unit is eliminated.

Keep in mind that the team each unit belongs doesn't matter when determining the outcome of a collision.

Additionally, if an object moves (or is moved, by getting batted) to a water tile or outside of the map, it is destroyed.

[1] Locations at distance at most 2.

8 Communication and Vision

Each unit can only sense the objects (units, balls, etc.) that are inside its vision radius. Vision is not shared. This means that objects detected by a given unit might not be detected by others.

Units run independently and they don't share memory. However, units can communicate by reading and writing on a shared array of 1000000 integers. The integer values of this array are initialized at 0. Both reading and writing on this array can be performed using the methods at *UnitController*.

9 Unit Order and Scheduling

Units take turns on each round. First, all units of the first team are scheduled, followed by the units of the second team. Whenever the HQ creates another unit x, x is inserted right after the HQ in the turn queue. However, units can modify the turn order by scheduling other units of the same team that haven't taken their turn yet in that round. Whenever a unit A schedules a unit B, B's turn is inserted right after A, and it stays that way until A or B are scheduled by another unit, or until A schedules another unit.

Note that the HQ is always the first unit to move in each team.

10 Energy

Energy is an approximate indicator of the number of basic instructions that each unit performs. More precisely, each bytecode instruction performed by a given unit consumes one unit of energy (except internal operations of the methods provided in the documentation, these consume a constant amount of bytecode which is given in the documentation). For users not familiarized with Java, it is not necessary to know how bytecode works, however it is good to have in mind that it is somewhat proportional to the number of code instructions. The amount of energy consumed up to a certain instruction can be accessed at any time using the methods in *UnitController*. Whenever a unit surpasses the amount of energy allowed (currently set to 15000), the unit pauses and continues running the remaining instructions during its next turn.

11 User Instructions

Players must fill the *run* method of the *UnitPlayer* class, which is run independently by all units of the player's team. This function has a *UnitController* as input, which is used to give orders to the given unit and to get information of the visible tiles or the common array, among others (check the documentation for more details).

Run does the following: Whenever a new unit is created, its *run* method is executed until either the unit finishes its turn by calling the *yield* method, or when it surpasses the amount of allowed energy. If a unit returns from its run method, it dies. Because of this it is suggested to keep its instructions inside a *while(true)* statement, and to finish each iteration with a call to *yield* (check *nullplayer* and *demoplayer*).

12 Implementation info

You may skip this section if you're not enough familiarized with Java.

Each unit is run in an independent thread. Each of these threads gets reactivated every time the unit is scheduled (once every round, following the same relative order every round). For safety reasons, it is forbidden to use any method or class outside *java/lang*, *java/math* and *java/util* (and some of the sub-classes of these). It is also forbidden to use static variables (which include switch statements, since they internally do so).

Even though codes that break these rules are automatically detected the instrumenter, we are going to check all the finalists' codes manually to be sure that everyone is playing a fair game. If a team breaks any of these rules without informing any of the devs, it will be disqualified.