Google AI Challenge

Organized by the University of Waterloo Computer Science Club and sponsored by Google

Sign In | Sign Up

Planet Wars Specification

Overview

Planet Wars is a game based on Galcon, but is designed to be a simpler target for bots. The contest version of the game is for two players.

A game of Planet Wars takes place on a map which contains several planets, each of which has some number of ships on it. Each planet may have a different number of ships. The planets may belong to one of three different owners: you, your opponent, or neutral. The game has a certain maximum number of turns. At the time of this writing, the maximum number of turns on the official server is 200, but it is not yet a part of the specification. Provided that neither player performs an invalid action, the player with the most ships at the end of the game wins. The game may also end earlier if one of the players loses all his ships, in which case the player that has ships remaining wins instantly. If both players have the same number of ships when the game ends, it's a draw.

On each turn, the player may choose to send fleets of ships from any planet he owns to any other planet on the map. He may send as many fleets as he wishes on a single turn as long as he has enough ships to supply them. After sending fleets, each planet owned by a player (not owned by neutral) will increase the forces there according to that planet's "growth" rate. Different planets have different growth rates. The fleets will then take some number of turns to reach their destination planets, where they will then fight any opposing

forces there and, if they win, take ownership of the planet. Fleets cannot be redirected during travel. Players may continue to send more fleets on later turns even while older fleets are in transit.

The Map

Maps have no particular dimensions and are defined completely in terms of the planets and fleets in them.

Planets

Planet *positions* are specified relative to a common origin in Euclidean space. The coordinates are given as floating point numbers. Planets never move and are never added or removed as the game progresses. Planets are not allowed to occupy the exact same position on the map.

The *owner* of a planet can be neutral, player 1, or player 2. Players always see themselves as player 1 and their opponents as player 2. The engine works out how to display the world differently to each player. The owner of a planet can change throughout the game. The owner is given as an integer with the following mapping:

- 0 means neutral
- 1 means player 1 (yourself, from your point of view)
- 2 means player 2 (your opponent, from your point of view)

The *number of ships* is given as an integer, and it may change throughout the game.

The *growth rate* of the planet is the number of ships added to the planet after each turn. If the planet is currently owned by neutral, the growth rate is not applied. Only players can get new ships through growth. The growth rate of a planet will never change. It is given as an integer.

Each planet is also implicitly assigned an *ID*. These are assigned as integers according to the order in which the planets are specified in the map, starting from 0. A planet's ID will never change throughout the game.

Fleets

The *owner* is represented in the same way as for planets, and the *number of ships* is again an integer.

The *source planet* and *destination planet* are specified according to the planets' IDs as specified above.

The *total trip length* is given as an integer, representing the total number of turns required to make the trip from source to destination. The *turns remaining* is also an integer, representing the number of turns left from the current turn to arrive at the destination. Trip lengths are determined at the time of departure by taking the Euclidean distance to the destination $\langle \mathrm{dx}, \mathrm{dy} \rangle$ from the source $\langle \mathrm{sx}, \mathrm{sy} \rangle$ and rounding up. That is, $\lceil \sqrt{(\mathrm{dx} - \mathrm{sx})^2 + (\mathrm{dy} - \mathrm{sy})^2} \rceil$.

File Format

The map file format is fairly simple. Each line may be blank, a planet, or a fleet, and each line is separated by Unix style line breaks (LF, not CR or CRLF). The `#` character and everything on the same line after it are treated as white space (comments), so a line beginning with a `#` character is considered blank, too.

Planet lines have the following format:

```
P <x:float> <y:float> <owner:int> <ships:int> <growth:int>
```

Fleet lines have the following format:

```
F <owner:int> <ships:int> <source:int> <destination:int> <total_turns:int> <remaining_turns:int>
```

Here is an example of a valid map:

```
# Example map

P 0    0    1 34 2  # Player one's home planet.
P 7    9    2 34 2  # Player two's home planet.
P 3.14 2.71 0 15 5  # A neutral planet with real-number coordinates.

F 1 15 0 1 12 2     # Player one has sent some ships to attack player two.
F 2 28 1 2  8 4     # Player two has sent some ships to take over the neutral planet.
```

In the above example, player 1's planet has ID 0, player 2's planet has ID 1, and the neutral planet has ID 2. In addition to the forces owned by each of the two players on the planets themselves, the two players each have fleets in transit. Player 1 has a fleet of 15 ships that is about to arrive at player 2's home planet. Player 2 has a fleet of 28 ships that is half way to the neutral planet.

Map files should not contain fleet lines, but this format is also used for game state updates, as will be described below, so the fleet lines are included in the specification here.

About the Current Maps

Most maps on the server and in the starter packs were generated programmatically by a Python script. You may also create your own to use for testing.

Turns

The game engine performs the following steps repeatedly:

1. Send the game state to the players.
2. Receive orders from both players.
3. Update the game state.
4. Check for endgame conditions.

There is an unspecified maximum turn limit. At the time of this writing, the maximum is 200 turns, but this may change. The intent is to have this number nailed down later in the contest period.

A *turn* is defined as the above four steps. They are performed up to 200 times and then the game stops. This means that the players receive the game state up to 200 times and send sets of orders up to 200 times.

Bot I/O

The engine on the official server launches your bot in a sandbox environment and communicates with it via stdin and stdout. It silently absorbs your stderr stream, and your bot is prohibited from writing to files. A single game is a single instance of your bot process. That is, the same bot process is used from turn to turn in a single game, but an entirely different process is used for a new game. The unofficial engines may have their own ways of handling these things, so if you are testing with a different engine you are responsible for being aware of these details on a per case basis.

At the beginning of each turn, the engine sends the map state to both bots. The format of this map is the same as the map file format specified under "The Map" section above. The end of the game state is denoted by a single line containing the text "`go`". For example, the same map example from earlier might look like this:

```
# Example map

P 0    0    1 34 2  # Player one's home planet.
P 7    9    2 34 2  # Player two's home planet.
P 3.14 2.71 0 15 5  # A neutral planet with real-number coordinates.

F 1 15 0 1 12 2     # Player one has sent some ships to attack player two.
```

```
F 2 28 1 2  8 4     # Player two has sent some ships to take over the neutral planet.
go
```

It's unlikely that the engine will include comments in this output, but I wouldn't rely on it not to. Note also that because planet IDs never change, the planets in the bot input will be in the same order from turn to turn.

Once the engine starts sending the game state to the bots, the bots each have 1 second of wall-clock time to receive the game state, process it, and send their orders. (~~The first turn is an exception, in which case the bots have 3 seconds of wall-clock time.~~ An additional two seconds is given after launching the bots but before sending them the game state. This was previously reported as having a longer first turn, but since the map is not given, no processing is possible for these two seconds. This is just to help bots written in languages whose VMs need some "warm-up" time.) Both bots perform these operations concurrently, and they are each unaware of what the other bot is doing. The choice of using wall-clock time rather than CPU time is primarily due to technical difficulties in measuring child processes' CPU times in real time. If a simple way to monitor CPU time instead is discovered, this part of the spec may change.

Bot Orders

An "order" is a line of text that the bot sends to the engine to make a fleet depart from a planet on the next game state update. The format of this line is as follows:

```
<source:int> <destination:int> <ships:int>
```

If the source and destination planets are the same, the bot instantly loses the game. If the number of ships is greater than is available at the source planet, the bot instantly loses the game. If the bot doesn't own the source planet, the bot instantly loses the game. The bot may issue as many orders in a single turn as it likes so long as the sum of all ships in fleets leaving a planet is not greater than the ships residing on the planet.

When the bot is done issuing commands to the engine, it sends a single "`go`" line. Here is an example of the output from a bot for a single turn:

```
1 17 50
4 17 50
go
```

That means the bot wants to send 50 ships from planet 1 to planet 17 and 50 ships from planet 4 to planet 17.

Game State Update

After receiving complete lists of commands from the players, the engine then updates the game state, advancing the game to the next turn. This happens in three phases: departure, advancement, and arrival.

Departure

In this phase, the players' commands are carried out. This consists of creating new fleets and removing the appropriate numbers of ships from each planet. Fleet trip lengths are determined by taking the Euclidean distance to the destination from the source and rounding up.

Advancement

This phase advances fleets and grows populations. Fleets are advanced by simply decrementing their "turns remaining" values.

For each planet, if it is owned by a player, its growth rate is added to its number of ships. For example, if the planet is owned by player 1, has 4 ships, and has a growth rate of 2, the planet will next have 6 ships, still having its growth rate of 2 and still being owned by player 1. However, had the planet been owned by neutral instead, its population would not have grown.

Arrival

This phase handles fleets whose "turns remaining" became zero during the advancement phase. It does so by considering each destination planet at a time.

For each planet, consider its owner and ship count along with the owners and ship counts of each fleet. We will call each of these groups a "force." Combine the forces according to their owners. For example, if the planet is owner by player 1 with a population of 5 ships, two of player 1's fleets are arriving with 3 ships each, and two of player 2's fleets are arriving with 5 ships each, the result of this operation gives us a force owned by player 1 consisting of 11 ships and a force owned by player 2 consisting of 10 ships.

If there is only one force, that is the new occupation of the planet.

If there are two forces, the new owner of the planet is the owner of the largest force, and the losing player's ship count is subtracted from the winning player's ship count as the new population. However, if both forces are the same size, then the winner is the original owner of the planet, and the planet's new ship count is zero.

If the original owner of the planet was neutral, then it is possible for there to be three forces fighting for one planet. In this case, the owner of the largest force is the new owner of the planet, and his ship count is reduced by the number of ships in the second largest force. If the top two forces are the same size, the original owner retains ownership of the planet but the forces are set to zero.

There is an intuitive way to understand these rules. Let's look at the rules with an example. Say player 1 has a force of 5 ships, player 2 has a force of 4 ships, and neutral has a force of 3 ships. We create groups of units that will battle, each group consisting of two ships from different forces or of three ships all from different forces. In this case, the division is as follows:

- P1 vs. P2 vs. N
- P1 vs. P2 vs. N
- P1 vs. P2 vs. N
- P1 vs. P2
- P1

Each group of two or three fighting ships cancels out. In this battle, player 1 wins with a single ship left.

If we apply the original procedure to the above battle, we use the three-way rules. The two largest forces are player 1's and player 2's. The winner must be player 1 because he has the most ships, and we subtract player 2's ships from player 1's to obtain the new ship count, $5 - 4 = 1$.

In actuality, all three cases for one, two and three forces follow the same procedure. In all cases, the largest force wins, and the second largest force is subtracted from the first, and in the case of a tie the original owner keeps the planet with zero ships remaining. A critical detail is that if forces completely cancel out then the original owner retains the planet.

Endgame Conditions

The following conditions will cause the game to end:

- The turn limit is reached. The winner is the player with the most ships, both on planets and in fleets. If both players have the same number of ships, it's a draw.
- One player runs out of ships and planets entirely. The winner is the other player.
- Both players run out of ships and planets at the same time. The game is a draw.
- A bot sends invalid data and forfeits the game.
- A bot crashes and forfeits the game.
- A bot exceeds the time limit without completing its orders (it never sends a line that says "`go`") it forfeits the game.
- A bot attempts to do something that the tournament manager deems a security issue and is disqualified.

More Information

If you have a question about the game mechanics, you may ask in the forums, ask in the #aichallenge IRC channel on Freenode, or go straight to the contest's source code.

Overview

- Home
- Current Rankings
- Problem Description
- Contest Rules

Getting Started

- Create Your Account
- Five Minute Quickstart Guide
- Starter Packages
- Tutorials & Strategy Guides
- Planet Wars Specification

Help

- Frequently Asked Questions
- Forums

Galcon

- Galcon Home
- Galcon for Desktop
- Galcon for iPhone
- Play Galcon Online for Free

Search for user: Go