Chexers battleground server

Last updated April 28, 2019

To play a game of Chexers against another team's program (without sharing code, as would entail a breach of academic integrity), we provide a 'battleground server', and a client program (in a Python package called `battleground`) for connecting to this server to play a game.

This document describes the structure and usage of the batleground client and some notes about how this resource relates to your Project Part B assessment.

Overview of structure

In case you are interested to know how the battleground client works, the main program essentially has the following structure:

1. Attempt to connect to the online battleground server.
2. Wait for the server to find some suitable opponents and send back the list of players in the game (colours are randomly assigned to players).
3. Coordinate a game of Chexers between your player package and the server:
    - When the server tells you what colour your player will play during this game, call your player's `__init__()` method with the appropriate string.
    - When the server says that it's your player's turn, call your player's `action()` method and send the chosen action back to the server.
    - When the server sends through the latest action played by your player or by an opponent, call your player's `update()` method to inform your player.
    - When the server says the game has ended (either normally by someone winning or by reaching a draw condition, or abnormally by someone playing a non-allowed action or disconnecting), print a message to let you know the result.

The server itself is similar to the referee program, except it waits for three battleground clients to connect and request a game (on the same channel) before starting a game loop, and instead of calling the `__init__()`, `action()` and `update()` methods on local player classes, it communicates with the battleground clients instead.

You can find out more about the client by inspecting the source code. You can find out more about the server by asking Matt by email or on the LMS Discussion Board.

Usage

To play a game using the battleground client, invoke it as follows, ensuring that the battleground package (the directory named `battleground`), referee package (the directory named `referee`), and your player package (the directory named with your team name) are both within your current directory, and the referee package is updated to at least version 1.1:

```
python -m battleground <player package> <online name>
```

where `python` is the name of a Python 3.6 interpreter, `<player package>` is the name of a package containing the class Player to be used for your player, and `<online name>` is a string that will be used to identify your player to your opponents.

The battleground client offers many additional command-line options, including for limiting the opponents matched using a 'channel' or 'passphrase', controlling output verbosity; creating an action log; and using other player classes (not named `Player`) from a package. You can find information about these additional options by running:

```
python -m battleground --help
```

Rules

While you are using the battleground client and server, please keep in mind the following two important rules:

1. Please **do not** use offensive or malicious strings as your online name.
2. Please **try not to** overload the server, intentionally or accedentally, e.g. by sending thousands of connection requests within a short period.

Relation to Project Part B

The use of the battleground client and server is *completely optional* for Project Part B. However, note that you may benefit from the chance to test your player program against other students' players, as you are interested in creating a player that can beat the benchmark players we will test your player against while marking.

Please note the following additional points about the battleground:

- In the case of unexpected problems we will try our best to keep the server online and functioning correctly. Nevertheless, we cannot provide a guarantee that the server will remain online between now and the project deadline. You should avoid relying on having the server available as your only means of testing your program's performance. You can help keep the server online by reporting any bugs to the LMS Discussion Board
- Games played through the server do not allow the enforcement of resource limits. This means you can throw as much computation time and space at the games as you have patience and RAM. Please coordinate this with your opponent separately. On a related note, you probably won't be able to connect to the server from `dimefox`, since `dimefox` does not allow network access. This means you will need to run the battleground client on your own computer (or server) or on the lab computers to play.