Sean Miles
Jonathan Acoltzi
Pratit Vithalani
CSS432

Documentation
Multiplayer Battleship

Overview

Team Cookie Monsters' term project is a multiplayer version of the Battleship turn-based board game. Both players have 5 battleships with sizes from 2-6 and place them throughout their grid, which the other player cannot see. The players then take turns guessing where the other players' battleships are, and if guessed correctly, the battleship receives a hit. If all slots on the battleship are hit, the battleship is sunk. A player wins when all of the opposing player's battleships are sunk.

Instructions

Required Files

- `client.py`
- `server.py`
- `player.py`
- `gameBoard.py`
- `battleship.py`

Dependencies

- Python
- pygame

Compiling and Running:

1. Make sure you have Python installed
    a. We ran ours using Python version 3.7, so if yours does not work, please update Python
    b. You can see what version of Python you are running by running: "python3 --version" or "python --version" in your terminal
    c. If you do not have pygame, install pygame using either pip or pip3 by running: "pip3 install pygame" or "pip install pygame"
2. Make sure you are connected to the UW VPN (F5)

3. Download the zip file and get to that directory on terminal
    a. The zip file is located on the Canvas discussion board
4. To start the game, run “python3 gameBoard.py” or “python gameBoard.py”

How to Play

1. Launch the game, and provide a username to register yourself in the game.
2. List all lobbies currently being played, and create a game if there are no open lobbies, or join a game using the game ID if there is an open lobby.
3. Once in the game, click anywhere on the right-side grid to place the start of a battleship. Click again to place the end of your battleship. The order of battleships placed is smallest to largest, starting with size 2 and ending with size 6.
4. Once both players have placed all their battleships, the player that created the lobby begins their turn. Click anywhere on the left-side grid to guess where the opponent’s battleship is. If it was a miss, the cell turns black. If it was a hit, the cell turns red.
5. Now the other player begins their turn, repeating the guess process in step 4. When they guess, if it was a miss, the cell on your right-side grid turns white. If it was a hit, the cell on your right-side grid turns red. If all the slots on a player’s battleship are hit, the battleship is sunk.
6. The game continues until all of a player’s battleships are sunk, and a player wins.
7. Once the game ends, you will be prompted if you want to play again. Answer Y to return to the lobby to provide a new username, or N to quit the game.

Protocol Implementation

Register

Our game allows players to register themselves with a username, as long as no other currently registered player has the same username. If the player tries to play with an already existing username, the program will ask them to give a unique name until they do.

List Games

After registering, the player will be prompted to either join a game or create one. If the player enters in that they would like to join a game, a list of games that only have one player is displayed. The player is then asked to enter the game ID of the game they would like to join, or create a new game.

Create Game

After registering, the player will be prompted to either join a game or create one. If the player enters that they would like to create a game, a new game will be created for them, and they will be put on wait while another player joins them.

Join Game

After listing the games to the player, the player has to enter in the game ID of the game they would like to join. Then, the game will start for both players.

Exit Game

After completion of the game, the player will be asked if they would like to play again. If the player does not want to play anymore, the player will exit the game.

Unregister

When a game is completed, the player is automatically unregistered from the system.

Application Specific Protocol

For our application specific protocol, we implemented a message system that transmits data between players using an action keyword with included information about the action:

- NEW
    - Format: `NEW player_id`
    - Example: `NEW cookiemonster12`
    - Description: The NEW message is sent to the server when one player is starting a connection with the server. Their desired name is sent with it, as player_id, for the server to check if someone has it or not.
- CONFIRMED
    - Format: `CONFIRMED player_id`
    - Example: `CONFIRMED cookiemonster12`
    - Description: The CONFIRMED message is sent from the server to a player once the server determines that the player_id has not been used yet, and now can be used by the player who requested it.
- JOIN
    - Format: `JOIN player_id game_id`
    - Example: `JOIN cookiemonster12 113c2a58-6895-11ea-b87e-fa163e8f2f46`
    - Description: The JOIN message is sent when one player wants to join an already created game. The player sends the game_id that they want to join and the game starts for both players.

- LIST
  - Format: `LIST player_id`
  - Example: `LIST cookiemonster12`
  - Description: The LIST message is sent from the player to the server when the player wants to join a game. The server gives back the dictionary of all of the game_ids and the associated player_ids waiting for someone to join them.
- CREATE
  - Format: `CREATE player_id`
  - Example: `CREATE cookiemonster12`
  - Description: The CREATE message is sent from a player to the server when a player wants to make their own game and wait for someone else to join it.
- FAIL
  - Format: `FAIL`
  - Example: `FAIL`
  - Description: The FAIL message is sent from the server to a player if the player_id they requested is already taken or if the game_id they want to join is non-existent
- GUESS
  - Format: `GUESS player_id game_id x y`
  - Example: `GUESS cookiemonster12 113c2a58-6895-11ea-b87e-fa163e8f2f46 3 5`
  - Description: The GUESS message is sent as one player guessing where the other player's battleship is. The other player checks if the grid cell at (x, y) contains a Battleship. If it does, they respond with a HIT message. If it does not, they respond with a MISS message.
- HIT
  - Format: `HIT player_id game_id x y`
  - Example: `HIT cookiemonster12 113c2a58-6895-11ea-b87e-fa163e8f2f46 3 5`
  - Description: The HIT message is sent as one player confirming that their Battleship was hit at (x, y) so that the other player can update their grid to display this hit.
- MISS
  - Format: `MISS player_id game_id x y`
  - Example: `MISS cookiemonster12 113c2a58-6895-11ea-b87e-fa163e8f2f46 3 5`
  - Description: The MISS message is sent as one player confirming that their Battleship was missed at (x, y) so that the other player can update their grid to display this miss.
- READY

    - Format: `READY player_id game_id`
    - Example: `READY cookiemonster12 113c2a58-6895-11ea-b87e-fa163e8f2f46`
    - Description: The READY message is sent as one player confirming that they are ready to begin the game after their Battleships have been placed in the grid.
- END
    - Format: `END player_id game_id`
    - Example: `END cookiemonster12 113c2a58-6895-11ea-b87e-fa163e8f2f46`
    - Description: The END message is sent as one player confirming that the game has ended as all of their Battleships have been sunk. The player sending the END message is always the loser.

Team Member Contribution

- Sean = 33%
- Jonathan = 33%
- Pratit = 33%