
EECS 581 Battleship Project 2 Documentation

General Overview:

- **Libraries Required**:
  - **pygame**: `pip3 install pygame`
  - **matplotlib**: `pip3 install matplotlib`
- **How to Run**:
  - `python3 battleship.py`
  - Ensure that the script is run from the correct directory.
- **Pygame Overview**:
  - The game is built using `pygame`, and a good starting point can be found at Pygame Getting Started.
  - Pygame offers tools for creating grid systems using `pygame.Rect` objects, which we use for creating and handling grids.
  - Each grid in the game is a 2D array filled with `pygame.Rect` objects.
  - **Player Ships**: Stored as a 2D array where each inner array represents one ship, which contains `pygame.Rect` objects for each part of the ship.
- **Helpful Resources:**
  - How to Make a Grid in Pygame
  - Creating a Rect Grid in Pygame
  - Text Handling in Pygame

Main File: `battleship.py`

Description:

- **Main Entry Point**: This file is the entry point of the game and should be run using `python3 battleship.py`.
- The core function called is `main()` which initializes and manages game flow, including setting up the game window and handling turns between Player 1 and Player 2 (or AI in singleplayer mode).

Broad Overview:

- **Main Loop**: The game operates within a loop until one player or the AI sinks all the opponent's ships.
- **Player Ship Placement**: Each player must place their ships using the functionality provided in `place_ships.py`.
- **Turn-Based Logic**: The game alternates turns using the `player1Turn` boolean, which determines whose move it is.
- **Input Handling**: During a player's turn, clicks are validated, and if the click corresponds to a valid target (a grid cell that hasn't been clicked before), it will register as either a hit or a miss.
- **Game Flow**:
    1. Players place their ships.
    2. Each player or AI takes turns attempting to hit the opponent's ships.
    3. Hits and misses are recorded, and ships are updated accordingly.
    4. The game ends when all ships for one player are destroyed.

Text Management: `add_text.py`

Overview:

- **Purpose**: Handles the display of text, labels, and timeouts throughout the game.

Key Functions:

- **`add_text(screen, text)`**: Displays the specified `text` at the top of the game window.
- **`time_out(screen)`**: Displays a timeout message and ends the game if more than 15 seconds pass between ship placements.
- **`add_labels_ships(screen)`**: Adds column labels to the player's ship grid.
- **`add_labels_target(screen)`**: Adds column labels to the target grid.
- **`add_labels_middle(screen)`**: Adds row labels to the middle of the game screen.

Ship Placement: `place_ships.py`

Overview:

- **Purpose**: Manages the placement of ships for both Player 1 and Player 2 (or the AI).

Key Functions:

- `placePlayer1Ships(screen, ships, placedShips, shipBoard)`: Handles the logic for Player 1's ship placement. Each ship is placed by clicking on valid grid cells, and the placement is validated to ensure no overlap and correct adjacency.
- `placeAiShips(screen, ships, placedShips, shipBoard)`: Similar to Player 1's ship placement but is automated for the AI.

Helper Functions:

- `addShip()`: Adds a ship part to the grid by validating if it touches existing ship parts and doesn't overlap with other ships.
- `inShips()`: Ensures that the ship part isn't part of another ship.

Ship Number Selection: `get_ships_num.py`

Overview:

- **Purpose**: Lets players select the number of ships for the game by clicking on displayed options.

Key Functions:

- `get_ships(player1ships, player2ships, screen, player1placedShips, player2placedShips)`: Retrieves the number of ships for each player based on user input.
- `place_options(screen)`: Displays rectangles representing each choice for the number of ships.
- `get_index(screen, pos)`: Determines which option the user clicked by checking which rectangle was clicked.

New File: `singleplayer.py`

Description:

- This file implements the logic for playing against an AI opponent, with different difficulty levels. It utilizes `battleship.py`, `add_text.py`, `place_ships.py`, and other files to function.

Game Flow:

1. **Ship Placement**: Player 1 places their ships manually, while the AI places ships automatically.
2. **Turn-Based Logic**: After both players (Player 1 and AI) have placed their ships, the game alternates turns.
3. **AI Difficulty**:
    - **Easy**: AI fires randomly at available target grid cells.
    - **Hard**: AI knows the exact location of Player 1's ships and doesn't miss.

Key Functions:

- **`run(ai_difficulty="easy")`**: Initializes and manages the singleplayer game. Sets up the AI difficulty and loops until a player or AI wins by sinking all the opponent's ships.
- **`ai_easy(player_ships, target_board, hits, misses, valid_moves)`**: AI fires randomly from a list of valid moves.
- **`ai_hard(player_ships)`**: AI targets specific coordinates where it knows Player 1's ships are located, ensuring it never misses.
- **`check_if_ai_wins()`**: Checks whether the AI has sunk all of Player 1's ships based on the number of hits.

End Condition:

- If the AI hits all of Player 1's ships, the game ends with a victory message displayed for the AI.