Battleship Version 2.0

Team 17
Abraham Lopez, Andy Trinh, Brisa Andrade, Jennifer Aber, Muskan Sharma

Overview

This document covers the AI portion of this application, including a brief description of the four levels of play, the main subroutines for the AI functionality, and some added helper functions.

When starting the game of Battleship, you have the option to select level of play, options 1-4. Select option 1 for a game against a human opponent. Levels 2-4 are in increasing levels of difficulty. AI level 2 opponent places their ships at random on the grid, and attacks at random. Level 3 opponent also places their ships at random, however, instead of attacking at random locations throughout the game, they are programmed to attack adjacent squares after an initial hit. Level 4 opponent can guess the location of player 1 ships, so winning a game against this opponent is nearly impossible.

Functions

`auto_place_ships(screen, board, clock)`

This function uses a random number generator to select various locations on the grid as well as vertical or horizontal alignment.

`updated_transition_between_turns(pnum)`

An update of the transition between turns function included with version 1.0 of the code, this eliminates the event listener to minimize the time between turns. Prior to making this change, the application lagged badly, making it difficult to play.

`auto_attack_lvl2(board, pnum)`

This subroutine uses basically the same logic as player turn, but eliminates the event handler to allow for automated attacks. A random number generator is used to select locations at random on the grid to attack.

`auto_attack_lvl3(board, pnum)`

This subroutine uses the `return_adj` helper function to search for adjacent squares after a hit. Besides this, the function is similar to `auto_attack_lvl2` in that it attacks at random if no prior hits are found.

`auto_attack_lvl4(board, pnum)`

This subroutine searches the board looking for the opponents ship, meaning a cell_value of 1. When it finds the ships, it sets the cell_value to -2, a hit.

```
return_adj(board, x, y)
```

This helper function takes the coordinates of a hit location, searches the adjacent squares for a 1 or a 0, and returns the coordinates of the new location as well as a flag indicating an attack is possible. If a location has been attacked previously (even if there was a miss) it is not attackable, and this function will return false.