Introduction

A variation of Battleship will be implemented, consisting of 2 players, connected via TCP to play together, or 1 player playing against an AI opponent. All classic Battleship rules will be followed, with some additional features: cards, mines, and abilities. Cards will be drawn at each turn, revealing game events and possible buffs and de-buffs. Some examples are: player misses one turn, player gets one free shot, and player places 1 free mine. If a user guesses a coordinate which contains a mine, their next turn is skipped. Abilities would be along the lines of: clear all de-buffs, next guess reveals an area that is 1x2 instead of 1x1, all with a cooldown of a certain number of turns. Finally, players will be able to quit and/or save the game state at any time and load a saved game state from the start menu.

The starting menu will resemble Figure 1. The player will have the option to load a game that was previously saved in memory or start a new game. A help button is also available to assist players with the 2 options.

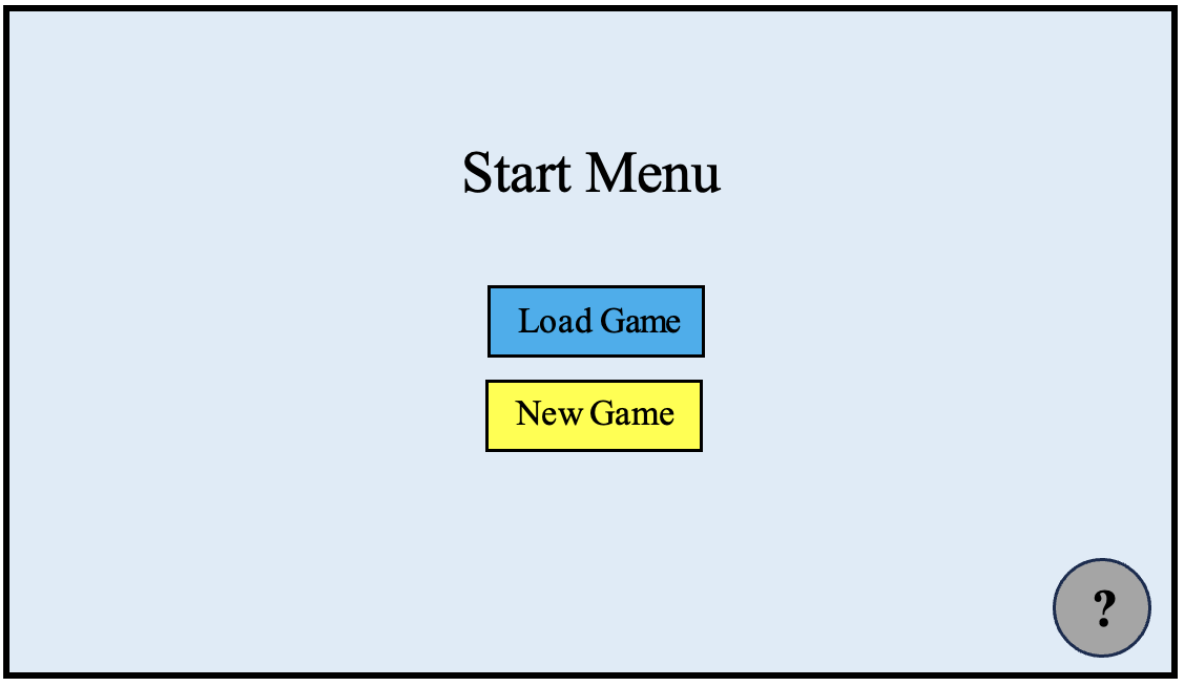


Figure 1. Start menu layout.

Starting a new game will prompt the user to either select an AI opponent, invite another player to join their game, or join another player's game (Figure 2).

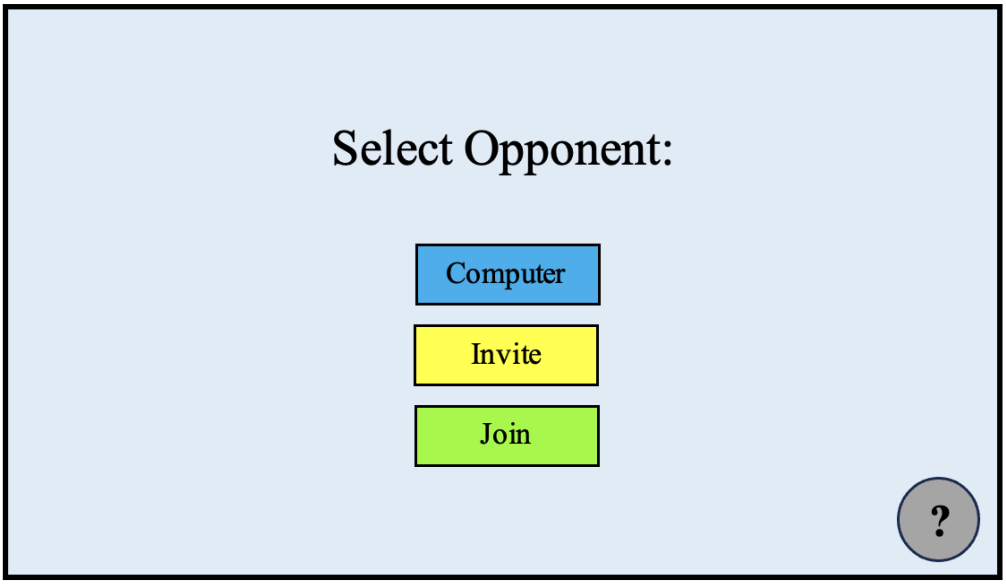


Figure 2. Opponent selection layout.

Once the game begins, the screen will resemble Figure 3. The "A" labels will be replaced by ability illustrations. The yellow outline surrounding the player's score shows which player's turn it is for the round, and the timer indicates the time remaining for the current player's turn. The "Help", "Save", and "Quit" buttons will show the player the game rules, save the game state to memory, or abandon and close the current game, respectively.

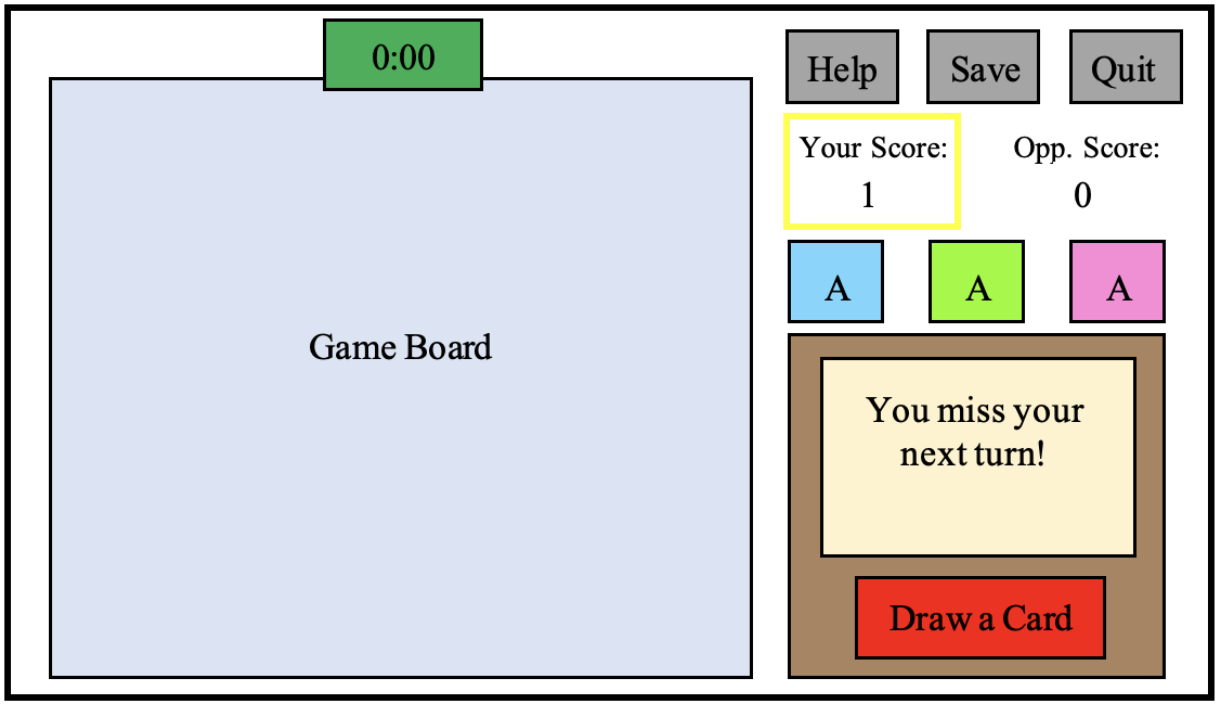


Figure 3. Game layout.

User Stories

1. Tutorial Mode

Alex turns on his computer to play battleship. He is brand new and does not yet know the rules of the game. From the start menu, he selects the "tutorial" option. This takes him to an interactive battleship board which simulates setting up and playing a real game, explaining each step. Alex uses this tutorial to learn the rules of the game as well as how to use the interface.

2. Single Player Mode

Alex wishes to practice playing battleship against a computer. From the start menu, he selects the "single player" option. He is then given a choice between 3 different levels of difficulty: beginner, intermediate, and advanced. He selects the beginner level and the game begins. He sets up and plays the game (see 4. Setting Up, 5. Playing). Once a winner is declared, the game ends and Alex is brought back to the main menu.

3. Multiplayer Mode

Alex and John wish to play battleship against each other. From the start menu, they select multiplayer mode from their own devices. They are given instructions on how to connect to each other. The game begins and both players set up their boards. They play the game until a winner is declared. Once this happens, both Alex and John are returned to the main menu.

4. Setting Up

Alex has just started a game of Battleship. He is given 60 seconds to set up his position. He is shown a graphical representation of his battlefield, which is a grid where rows are indexed by numbers and columns indexed by letters. Cells can be identified using a letter-number pair (ex. C6). Alex is given ships which he must place on the board. Ships are 1 cell wide and 1-5 cells tall and can be oriented either horizontally or vertically. For each ship, Alex rotates it to his desired orientation and then clicks and drags it to his desired position on the board. Ships may not overlap (ie. one cell may be occupied by at most 1 ship). Once a player is finished setting up, they may click the "done" button. If all players finish early, the game may begin without waiting the full 60 seconds. If a player is not finished setting up in 60 seconds, the game will choose a random position for their ships. A player may also click a button to generate a random position.

5. Playing

Alex has finished setting up his board and now is playing the game. He is shown a graphical representation of his opponent's board. When it is his turn, he is given 20 seconds to select a cell on which to fire. If he does not choose a cell in time, a random one is chosen. If his shot is in the same cell as one of his opponent's ships, the game displays a "hit" graphic in that cell. If not, the game displays a "miss" graphic. Once his turn is over, the screen transitions into a view of Alex's board which includes the position of his ships and the cells in which his opponent has shot. The game continues until one player sinks all the opposing ships.

6. Saving

Alex is currently in a game of battleship against the computer but needs to pause and would like to come back to it later. He selects the “save” option to save the current game. Later on, he presses “load game” from the start menu and the game continues where it was last left.

Functional Requirements(WIP)

Game logic	
Function	Displays and facilitates gameplay.
Description	Runs the game and communicates the current game state to the player when actions are taken either by a player or ai.
Inputs	User inputs (u0,u1), Board state (b)
Source	User interaction and AI decisions
Outputs	Current game state.
Destination	Game board.
Action	On user input displays where a shot was taken. If the shot hits then Game Board communicates the hit to the user, if the shot missed then Game Board communicates the miss and stores the data in the game.
Requires	Two user inputs and the game state so the input can be compared to the game state to check for hits/misses.
Precondition	The Board has been set up in a previous stage.
Postcondition	Turn changes if u0 misses then u0 is replaced by u1.

Condition	**Action**
User guesses correctly (u0==b&&b==1)	b=0
User guesses incorrectly (u0==b&&b==0)	b=2

Start menu	
Function	Allows users to start and choose which type of game to play.
Description	Select the type of opponent and whether to load or start a new game.
Action	Depending on user input loads a new/old game against an ai, another player, or tutorial
Postcondition	Screen moves to game board.

Settings	
Function	Allows users to customize/save their game experience.
Description	Allows users to adjust game sound, window mode, and enable accessibility settings (colour blind mode). Also will be where the save option is found.
Action	On user input, adjust the sound level the game is outputting (music, game, effects etc…), and toggle color blind mode.
Postcondition	Change the settings to the user's likings/ Saves the current game.

Non-Functional Requirements

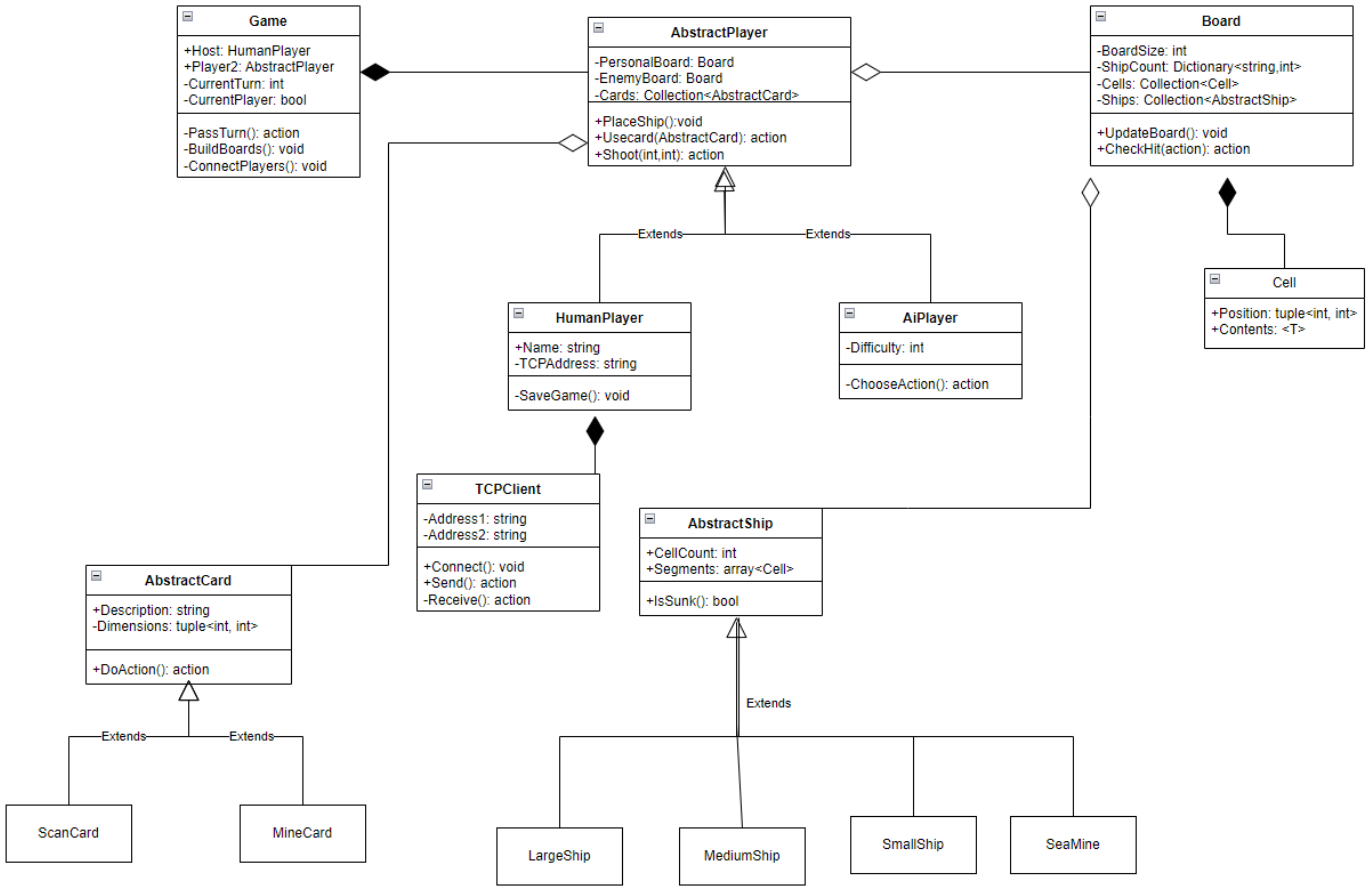
Project		
Technologies used	Programming language	Python 3
	Graphics	Pygame
	Version control system	Git https://github.sfu.ca/kza42/warthog276.git

Start Menu	
UI/UX	Intuitive and user friendly. The user should not have to read the help for this, but it is still available.
Saving/Loading	The file should not be larger than 50 kB. It should have only the strict necessities to avoid an overly large file.

Game Play	
Animation	Movements such as dragging and dropping ships, text appearing on the screen, should be smooth and responsive.
Quality	Runs the game and communicates the current game state to the player when actions are taken either by a player or AI.
Experience	Games should be able to be completed in 20-30 min.

Computer Opponent	
Behavior	Behavior should be appropriate for the difficulty level selected by the user.
User Experience	To seem more natural, moves should not be carried out instantaneously, but instead wait for a few seconds after the timer has started.

UML Class Diagram



UML Use-Case Diagram