Design document for the   
computer game *Battleship*

# Game description

Battleship is a human-vs.-computer game that simulates a naval battle between two enemy fleets.   
The goal is to sink all the computer’s ships before it sinks yours.

To begin, each player secretly places 5 ships on a 10x10 peg-board, which their opponent cannot see. The board has 10 rows labeled A-J, and 10 columns labeled 1-10. Each player’s fleet has:

* 1 ship of length 5 (can take 5 hits before being sunk)
* 1 ship of length 4 (can take 4 hits before being sunk, etc.)
* 2 ships of length 3
* 1 ship of length 2

The ships may not overlap.

The players then take turns firing at their opponent’s fleet by announcing a target on their opponent’s board, for example, “H7”. If the opponent has a ship covering that location, he replies “Hit!” Otherwise, he replies “Miss!” The shooting player then marks that result on a separate 10x10 board.

A ship is sunk when it has been hit at every location it covers.

# Sketch of the game screen

The game screen will look something like this.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | The computer just scored a hit on C9!  Your turn…  Click a square on the left to fire at. |
| New Game | Quit |  | |

# Instructions for play

To open the game, open the Python file **Battleship.py** and push F5.

You will see two empty 10x10 grids, one for your own ships and one for recording your hits and misses against the computer’s fleet. On a side-bar, you will see 5 unplaced ships.

To place a ship, drag the ship’s icon to where you want to place it, and then release the mouse.

When it is your turn, click the square on your results-board where you want to fire. If it was a hit, that square will turn red. If it was a miss, the square will turn white. If that hit sunk a ship, that square and all other squares belonging to that ship will turn black.

To quit, click the **Quit** button. To start a new game, click the **New Game** button.

Warning: Clicking **Quit** or **New Game** permanently erases the current game.

# Pseudocode algorithm for the runGame() procedure

Computer randomly places its ships

Draw the player’s feedback board and the player’s ocean board

Player places his/her ships

**While** both players still have at least 1 surviving ship:

#Player’s turn

Player chooses a location to fire at

Determine if a hit has been scored

**If** hit scored:

Reduce that ship’s health by 1

Display the hit on the player’s results-board

**If** that ship’s health = 0:

Declare the ship sunk

Reduce the computer’s ship count by 1

**Else**:

Display a miss on the player’s results-board

#Computer’s turn

Computer randomly chooses a location to fire at

Determine if a hit has been scored

**If** hit scored:

Reduce the hit ship’s health by 1

Display the hit on the player’s ship

**if** that ship’s health is now 0:

Declare the ship sunk

Reduce the player’s ship count by 1

**else**:

Report where the computer fired and missed.

**If** computer’s ship count = 0:

Declare the player the winner

**Else**:

Declare the computer the winner