Battleship Lite Project

# W: Walk Through the Project

## Basic Requirements

To build a small, two-player console game that has its roots in the game Battleship from Mattel. There will be a 25-spot grid (A1 – E5). Each player will place five pegs on the board to represent their five ships. Players will then take turns firing on their opponent’s ships. The first person to sink all five ships wins.

## General Flow

Two users open up the console

Ask user 1 for where to place their ships

Ask user 2 for where to place their ships

Ask user 1 for a shot

Determine hit or miss

Determine if the game is over

Ask user 2 for a shot

Determine hit or miss

Determine if the game is over

Repeat until someone wins

Identify who the winner is.

Exit the application

# O: Open up the Requirements

## Additional Questions/Requirements

1. Is it the same console or two different consoles working together? **Same Console**
2. Does the other player get one last chance after being sunk? **No**
3. Do we want to capture/display statistics such as hit/miss ratio, etc.? **Just how many shots it took to win**
4. Only ship per spot
5. Do we allow a player to shoot the same spot twice? **No**
6. Do we show a visual of the grid? **Yes**
7. Do we store game data? **No**
8. Are we ever going to change the number of players? **Maybe**
9. Will we add a computer player? **Maybe**

## Full Requirements

1. 2-player game
2. 25 spot grid (A1 - E5)
3. Each player gets 5 ships
4. Each ship takes up one spot
5. Players take turns firing
6. First person to sink all 5 wins
7. One console for everyone
8. No completing the round after 5 sunk ships.
9. Show a visual of the grid with hits and misses
10. Do now allow the user to shoot the same spot twice

# U: User Interface Design

Clear console

Heading at top says which player

Display 5 × 5 grid with boxes.

A B C D E

╔═╤═╤═╤═╤═╗

5║ │ │ │ │ ║5

╟─┼─┼─┼─┼─╢

4║ │ │ │ │ ║4

╟─┼─┼─┼─┼─╢

3║ │ │ │ │ ║3

╟─┼─┼─┼─┼─╢

2║ │ │ │ │ ║2

╟─┼─┼─┼─┼─╢

1║ │ │ │ │ ║1

╚═╧═╧═╧═╧═╝

A B C D E

As each ship is placed, but black circle in box, clearing console and redrawing grid between each:

A B C D E

╔═╤═╤═╤═╤═╗

5║ │ │ │ │ ║5

╟─┼─┼─┼─┼─╢

4║ │ │ │ │ ║4

╟─┼─┼─┼─┼─╢

3║ │ │●│ │ ║3

╟─┼─┼─┼─┼─╢

2║ │ │ │ │ ║2

╟─┼─┼─┼─┼─╢

1║ │ │ │ │ ║1

╚═╧═╧═╧═╧═╝

A B C D E

Once grid has all 5 ships, clear console and repeat for player 2

Clear console and display 2 grids: one blank, one with player 1’s ships, to ask player 1 for first shot

If a hit, use ×; if a miss use ○:

A B C D E

╔═╤═╤═╤═╤═╗

5║ │ │ │ │ ║5

╟─┼─┼─┼─┼─╢

4║ │○│ │ │ ║4

╟─┼─┼─┼─┼─╢

3║ │ │×│ │ ║3

╟─┼─┼─┼─┼─╢

2║ │ │ │ │ ║2

╟─┼─┼─┼─┼─╢

1║ │ │ │ │ ║1

╚═╧═╧═╧═╧═╝

A B C D E

Clear console and display 2 grids: one blank, one with player 2’s ships and player 1’s hits/misses, to ask player 2 for first shot

On next round, player 1 sees his ships and player 2’s hits/misses on “his” grid.

Repeat as needed. At end of game, display both player’s grids showing hits/misses/remaining ships on each side.

# L: Logic Design

Get grid coordinate of player 1’s first ship

Check to see if there is already a ship in that grid square. If so, return error and ask again. If not, place ship there.

Repeat until all 5 ships are placed

Repeat for player 2

Ask for player 1’s first shot

Check to see if spot was already shot. If so, return error and ask again. If not, shoot there.

Check to see if hit or miss.

If hit, check to see if player 2 has unsunk ships. If no, game is over. If yes, continue.

Repeat for player 2

Keep repeating until game is over

# D: Data Design

Instantiated Class “Grid Model” holds 5 × 5 2-dimensional array of type char.

Each array element can be (1) empty ( ), (2) have a ship (●); (3) have a miss (○); (4) have a hit/sunk ship (×).

Instantiated Class “Player Model” holds player name (string), player’s own grid (GridModel), and “targeting” grid (GridModel). Also has int for number of shots taken