

Informatics Institute of Technology

Formal Methods

6SENG005C.1

Course Work

Report

Pasindu Anuradha Godakanda

Uow Number – w1871482

IIT Number – 20212026

**Table of Contents**

[**1** **Machine Structure Diagram** 2](#_Toc185283190)

[**2** **Justifications:** 3](#_Toc185283191)

[**2.1** **SETS & CONSTANTS** 3](#_Toc185283192)

[**2.2** **VARIABLES & INVARIENTS** 4](#_Toc185283193)

[**3** **Machine Testing results:** 5](#_Toc185283194)

[**3.1** **Type Check Result** 5](#_Toc185283195)

[**3.2** **Machine Testing** 6](#_Toc185283196)

[**4** **Graphical Visualization** 9](#_Toc185283197)

[**4.1** **Indicators** 9](#_Toc185283198)

[**4.2** **States of the Game** 9](#_Toc185283199)

[**5** **State Graph Visualization.** 12](#_Toc185283200)

# **Machine Structure Diagram**

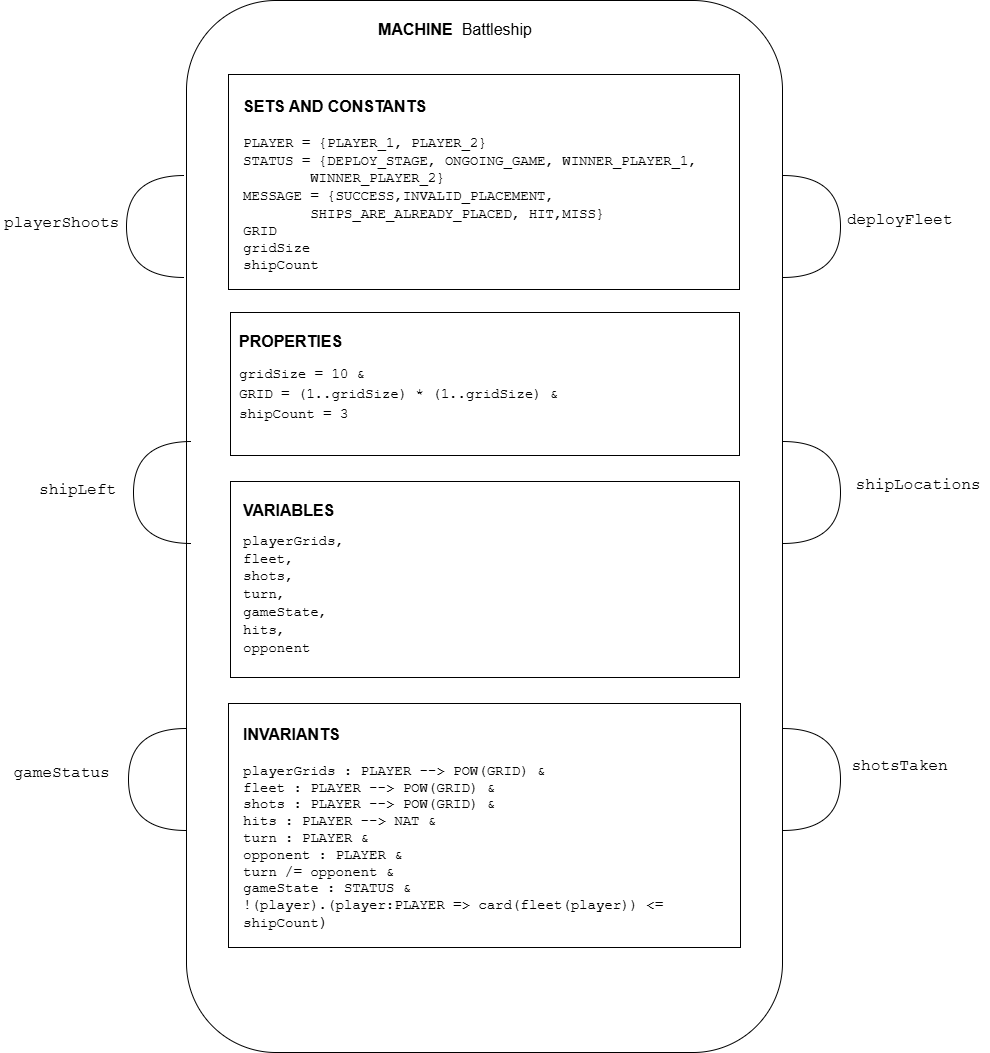


Figure 1: Machine Structure Diagram

# **Justifications:**

Detailed explanations for the assigned and utilized SETS, VARIABLES and INVARIENTS are provided below.

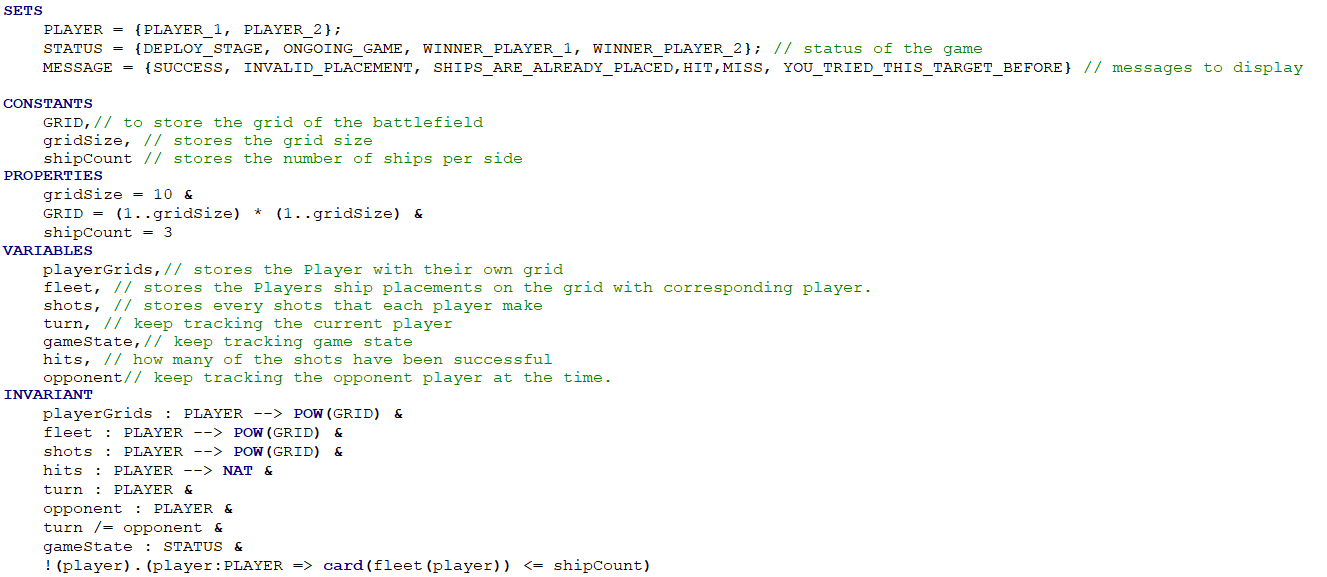


Figure 2: SETS, VARIABLES and INVARIENTS

## **SETS & CONSTANTS**

|  |  |
| --- | --- |
| **SET & CONSTANTS** | **Explanation** |
| PLAYER = {PLAYER\_1, PLAYER\_2} | This Represents the two participants in the game. Battleship game involves two players, requiring this set to distinguish between them. |
| STATUS = {DEPLOY\_STAGE, ONGOING\_GAME, WINNER\_PLAYER\_1, WINNER\_PLAYER\_2} | This Represents different game status. This is requires to keep track the game progression throughout the different stages of the game. |
| MESSAGE = {SUCCESS, INVALID\_PLACEMENT, SHIPS\_ARE\_ALREADY\_PLACED,HIT,MISS, YOU\_TRIED\_THIS\_TARGET\_BEFORE} | This Represents the feedback messages. This Requires to provide relevant game notifications during the play. |
| GRID | This is the game board coordinates which defines the playable area for ship placement and targeting. |
| gridSize | This constant used for set the grid dimensions. As for this specification it uses 10 x 10 grid, hence the grid size would be 10. |
| shipCount | This is the number of ships that each player can deploy. This fixes ship count for balanced gameplay. |

## **VARIABLES & INVARIENTS**

|  |  |
| --- | --- |
| **VARIABLES & INVARIENTS** | **Explanation** |
| playerGrids  playerGrids : PLAYER --> POW(GRID) | playerGrids uses to map players to their own grids and this keeps track of each player’s board setup. Each player must have a grid of valid coordinates. This ensures proper grid assignment. |
| Fleet  fleet : PLAYER --> POW(GRID)  !(player).(player:PLAYER => card(fleet(player)) <= shipCount) | Stores ships’ positions of each player for ship placement validation and hit detection.  Each player’s fleet must consist of valid coordinates within the grid.  A player’s fleet cannot exceed the allowed ship count. |
| Shots  shots : PLAYER --> POW(GRID) | Tracks all shots fired by each player for prevent duplicate shots and helps determine game progress.  All the shots must be valid grid coordinates. |
| Hits  hits : PLAYER --> NAT | Tracks successful hits per player. This requires for determine when a player wins the game.  Since negative hits are logically impossible and there can be an event that a player don’t have any successful hit. Hence this tracks natural numbers including 0. |
| Turn  turn : PLAYER | The game is alternates between player’s turn, hence the turn variable tracks the current player who’s attacking at the moment.  The current turn must belong to a valid player and it must be always assigned. |
| opponent  opponent : PLAYER  turn /= opponent | This is the opposing player during the current turn. This requires for determine whose ships are targeted.  Opponent also should be a valid player and since players cannot play against themselves, opponent cannot be the turn at the same time. |
| gameState  gameState : STATUS | This Represents the current state of the game. This is requires to manage game flow from deployment to conclusion.  gameState should be always a valid STATUS. |

# **Machine Testing results:**

## **Type Check Result**

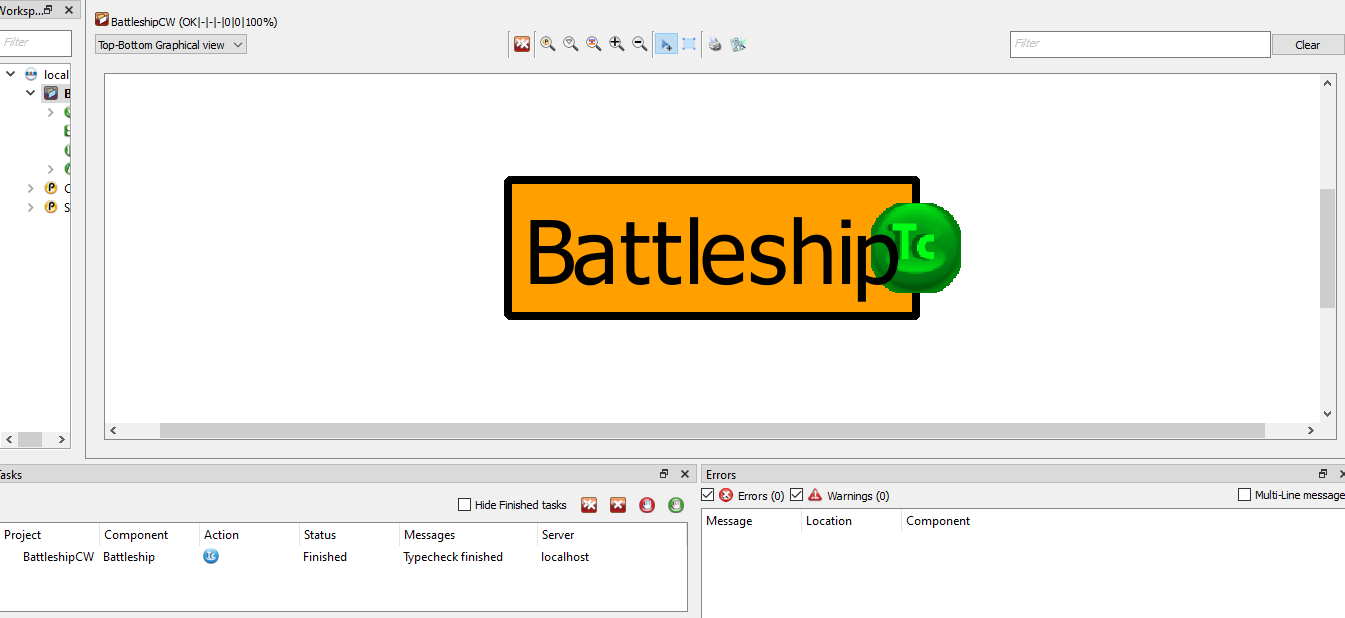


Figure 3: Type Check result

## **Machine Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output | Pass/Fail |
| TC-01 | Verify Initial State | - | gameState = DEPLOY\_STAGE, turn = PLAYER\_1 | Pass |
| Actual Output | | | | |
| TC-02 | Valid Fleet Deployment | deployFleet(PLAYER\_1, {(1,1), (2,3), (3,3)}) | report = SUCCESS | Pass |
| Actual Output | | | | |
| TC-03 | Invalid Fleet Deployment | deployFleet(PLAYER\_1, {(1,1), (1,2), (1,11)}) | report = INVALID\_PLACEMENT | Pass |
| Actual Output | | | | |
| TC-04 | Valid Shooting (Hit) | playerShoots((1,1)) (if (1,1) in opponent fleet) | report = HIT | Pass |
| Actual Output | | | | |
| TC-05 | Valid Shooting (Miss) | playerShoots((2,1)) (if (2,1) not in opponent fleet) | report = MISS | Pass |
| Actual Output | | | | |
| TC-06 | Repeated Targeting | playerShoots((1,1)) (already targeted) | report = YOU\_TRIED\_THIS\_TARGET\_BEFORE | Pass |
| Actual Output | | | | |
| TC-07 | Winning Condition Check | Sink all PLAYER\_2’s ships | gameState = WINNER\_PLAYER\_1 | Pass |
| Actual Output | | | | |
| TC-07 | Ship Location Query | shipLocations(PLAYER\_1) | Remaining ship positions | Pass |
| Actual Output | | | | |
| TC-09 | Shot Count Query | shotsTaken(PLAYER\_1) | Correct shot count | Pass |
| Actual Output | | | | |
| TC-10 | Game Status Query | gameStatus() | Current game state | Pass |
| Actual Output | | | | |

# **Graphical Visualization**

Regarding the Battleship game B specification, Graphical visualization also implemented using an Animation Function. Following are the main Indicators and various states of the graphical visualization of the Battleship B specification.

## **Indicators**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Opposing Player (who takes the attack) |  | Current Player (who attacks) |
|  | Missed shot |  | Unharmed Ship |
|  | Successful Hit (sinking Ship) |  |  |

## **States of the Game**

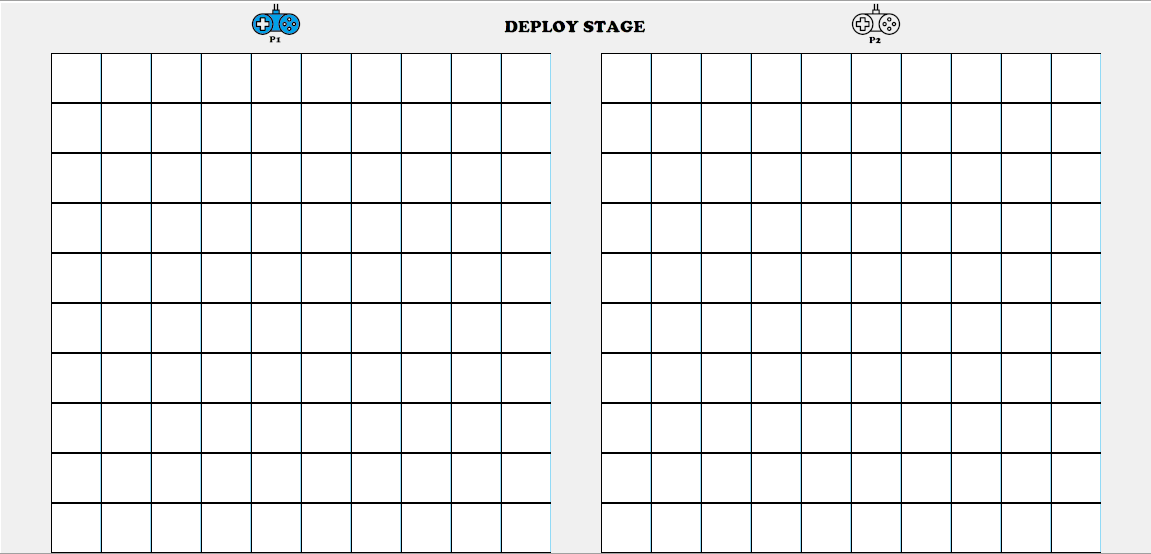


Figure 4: Deploy Game Stage (Initial Stage)

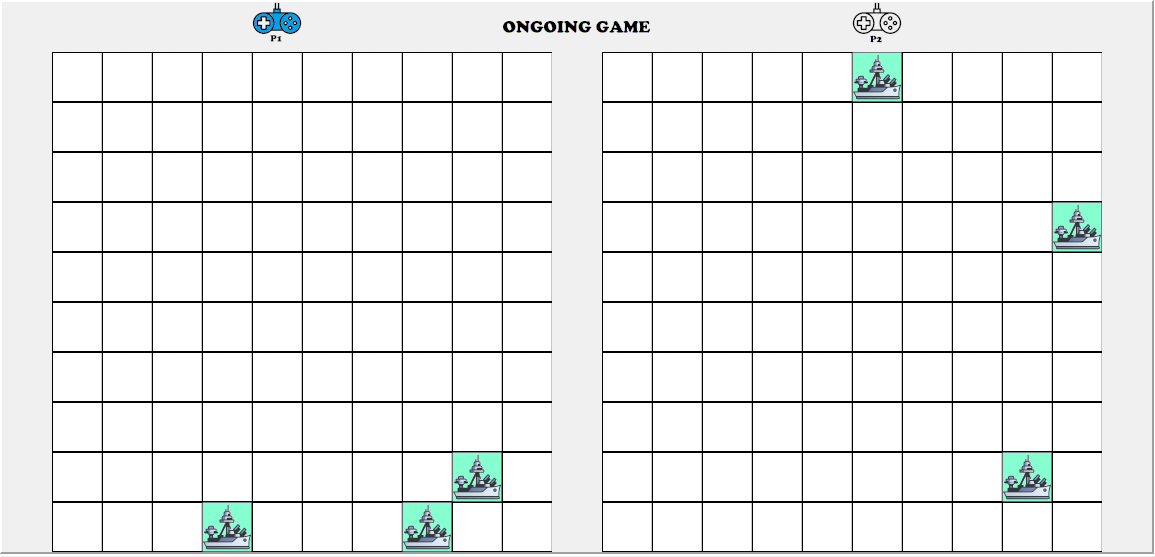


Figure 5: After the both the players placed their own fleets.

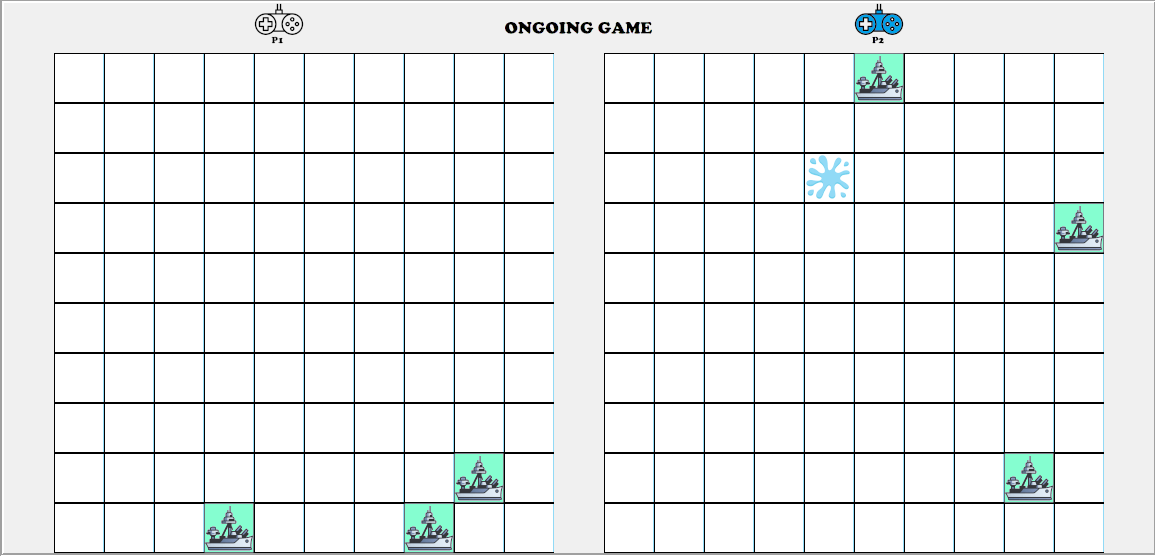


Figure 6: Player 1 fires a shot and Misses.

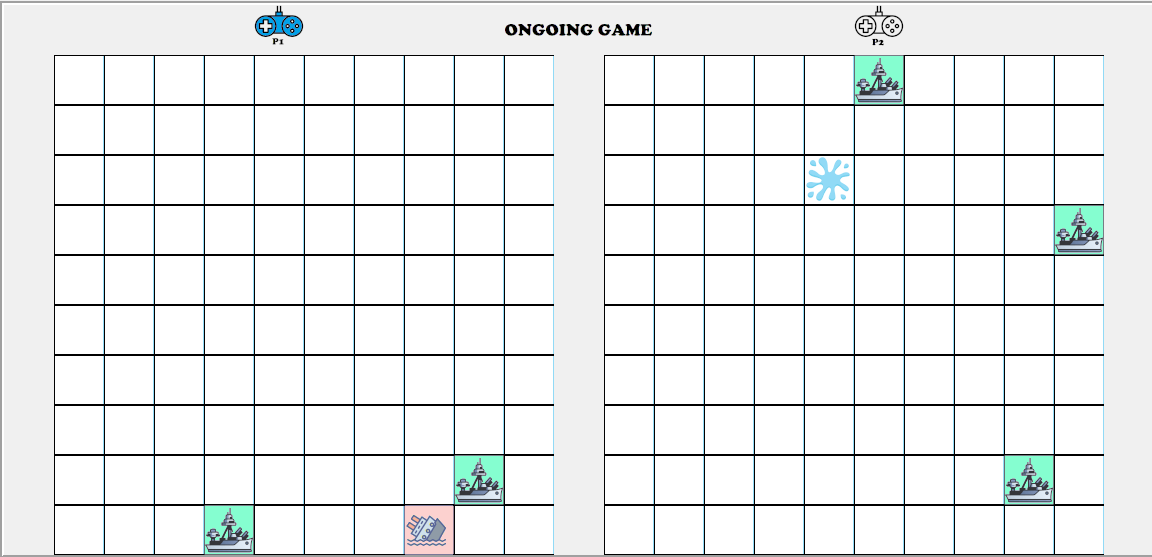


Figure 7: Player 2 Fires a shot and makes a successful hit.

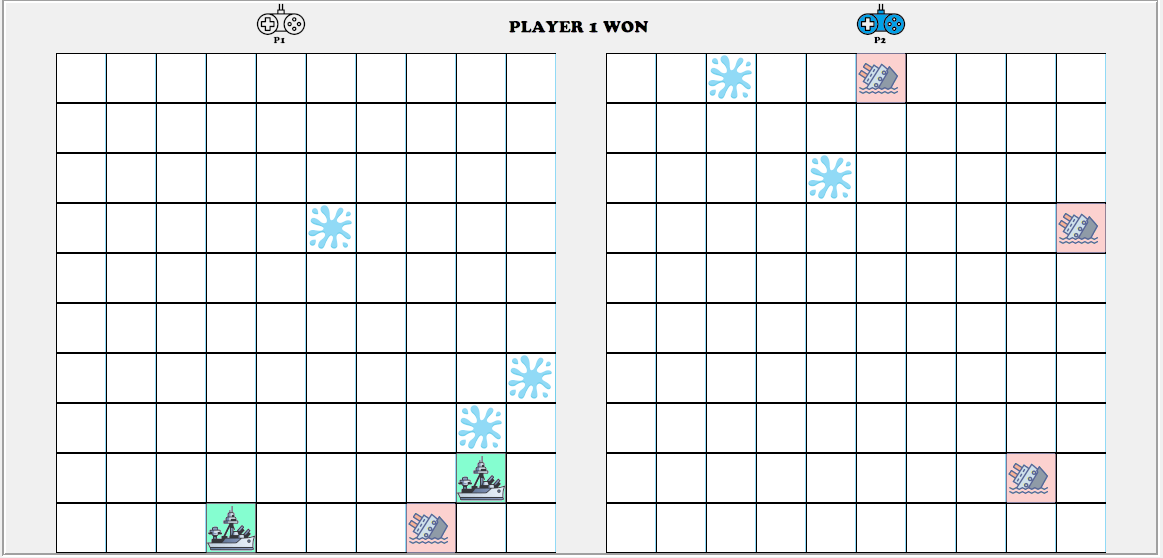


Figure 8: Player 1 sink all the ships of Player 2's Board and Win.

# **State Graph Visualization.**

Since the State Graph visualization of the battleship B specification is too large to display inside this report it is attached as a separate PDF down below. This state graph visualization is captured during ONGOING\_GAME state after deploy fleets and fire some shots.

[](Battleship.pdf)