# One person group

**1.Group members:**

Lana Salem

**2.GitHub URL:**

https://github.com/lana002/battleshipgame270

**3. High-Level Description of the Program**

The program is a console-based Battleship game that supports two modes: Player vs. Player (PvP) and Player vs. Bot (PvB). At its core, the game focuses on creating a smooth and strategic gameplay experience through well-designed logic and interactive mechanics.

In the PvP mode, players alternate turns to make moves, providing coordinates for their attacks. The game processes these moves by updating the game grid, marking hits, misses, and sunk ships. This mode relies heavily on user input handling and precise execution of game updates based on player actions and choices.

The PvB mode introduces an AI-controlled bot, which uses advanced strategies to challenge the player. The bot's logic is built around the same foundational functionality as the PvP mode, ensuring consistency in the game mechanics. However, the bot employs additional features, such as a FIFO (First-In-First-Out) queue to track unexplored cells near hits. This queue enables the bot to efficiently hunt down ships by focusing on areas likely to contain ship segments. For instance:

• When the bot registers a hit, it adds all surrounding unexplored cells to the queue.

• In subsequent turns, the bot prioritizes these cells, "tailing" the hit until it sinks the ship.

• This hunt logic extends to artillery and torpedo moves, where any resulting hits also enqueue surrounding cells for further exploration.

Additionally, the bot uses radar and smoke moves strategically:

• Radar: The bot avoids exhaustive scans at the start of the game. Instead, it uses a combination of unexplored cells percentages and thresholds to determine when radar is most effective. It ensures that radar is not used consecutively by introducing a checker and incorporates randomness towards endgame to avoid predictive patterns and enhancing its use.

• Smoke: The bot evaluates clusters of high ship density and where ships are most vulnerable to hits and strategically places smoke to obscure these areas. If no dense clusters are feasible, it defaults to random placement to maintain defensive utility.

At the heart of the game lies the shared functionality for direct-hit moves (e.g., fire, artillery, torpedo). All these moves follow a similar process:

1. The game identifies the affected area based on the move type (single cell for fire, a 2x2 grid for artillery, a line for torpedo).

2. It marks on a spare grid the cells affected by a direct-hit move. Then uses that grid to make updates and process hits and miss.

3. It evaluates whether a ship has been sunk and updates the game state accordingly.

By unifying this logic, the program ensures consistency and simplifies the addition of new features. This modular approach was the pillar of my project, as I started with it then built around it the rest of the program's logic.

## **4.Issues Faced**

One of the biggest challenges was designing the bot's hunt logic.
A key moment in the project was discovering a YouTube video about a Zelda minigame similar to Battleship. The video described a heatmap approach for optimizing moves, which inspired me to research further. I found an elaborate spreadsheet analyzing strategies to reduce the number of moves needed for a bot to win. This is where I learned about the hunt-and-destroy logic, which became a cornerstone of my bot design. Unfortunately, I didn't have the time to implement the heatmap itself, but the resources significantly influenced implementation.

Initially, I only used lasthitx and lasthity variables to track the bot's last successful attack. While this approach worked for basic moves, it fell short in tracking for artillery and torpedo attacks, which affect larger areas. It also made the bot's decisions overly simplistic and predictable. After trial and error, I switched to a queue-based system. This allows the bot to track unexplored cells around any hits systematically and tail ships until they are fully sunk.

Another issue arose with balancing the bot's use of radar and smoke moves. For radar, the bot sometimes relied too heavily on scans early in the game, making it seem overly efficient but unsustainable in the long run. I resolved this by introducing a threshold to limit radar usage based on unexplored percentages and sunken ships as well as ensuring randomness in decision-making at a certain point in the game. Similarly, smoke placement was initially random and ineffective. By prioritizing vulnerable clusters and evaluating how well protected certain areas are compared to the full grid, I improved the bot's defensive strategies.

## 5.Resolutions

The Zelda video and the accompanying spreadsheet provided invaluable insights. The hunt logic adapted from these resources allowed for more sophisticated bot strategies.

Transitioning from simple coordinate tracking to a FIFO queue resolved the limitations of single-hit tracking. This queue-based system efficiently handles larger move areas, ensuring that the bot remains methodical in hunting down ships.

Introducing randomness, threshold checks, and usage constraints for radar moves made the bot more balanced and unpredictable. For smoke, prioritizing vulnerable high-density clusters significantly improved its defensive potential.

## 6.Limitations

• I would have liked attempting to implement the heatmap approach, which could have improved the bot's move efficiency by identifying high-probability areas for attacks.

• Smoke placement focuses on immediate defensive priorities, without accounting for potential future threats.

• The bot could improve significantly if implemented with some sort of adaptive memory that updates with each turn. As it is right now, figuring its approach and predicting its direct hit moves is still quite easy.

## 7. Assumptions

• Players and the bot should adhere strictly to the rules, with no edge case exploitation.

• Ship placements are valid and randomized, with no overlaps.

- The bot operates without any prior knowledge of player ship locations which ensures fairness.

This project challenged me to think critically about game design and AI strategies. By building on foundational logic and drawing inspiration from external resources, I created a program that combines strategic depth and user interaction. While there are areas for improvement, this experience has been an invaluable learning opportunity :)