1. Introduction

The BattleC project involves the design and implementation of a two-player interactive Battleship game in C++, adhering to the principles of structured programming, object-oriented design, and efficient AI techniques. The game pits a human player against an AI agent, following traditional Battleship rules.

This report outlines the core features, implementation details, challenges faced, and future enhancements for the project.

---

2. Game Rules and Objectives

- **Game Rules**:
  - The game is played on two 10x10 grids, one for each player.
  - Ships of varying lengths are placed horizontally or vertically on the grid.
  - Players take turns guessing the location of the opponent's ships.
  - A ship is sunk when all its coordinates are hit.
- **Winning Condition**:
  - The first player to sink all of their opponent's ships wins.

---

3. Implementation Details

The project was implemented in C++ using structured programming and OOP techniques. It consists of the following components:

3.1. Class Design

1. **Ship Class**:
   - Represents a ship with properties such as name, size, coordinates, and the number of hits taken.
   - Methods:
     - addCoordinate(): Adds a coordinate to the ship.
     - occupiesCoordinate(): Checks if a coordinate belongs to the ship.
     - hit(): Registers a hit on the ship.
     - isSunk(): Checks if the ship is sunk.
2. **Board Class**:
   - Represents the player's grid and handles ship placement, attacks, and visual representation.
   - Methods:
     - placeShip(): Validates and places a ship on the board.

        - receiveAttack(): Processes an attack on the grid.
        - display(): Displays two boards side by side.
        - displayScoreboard(): Displays the scoreboard.
        - allShipsSunk(): Checks if all ships are sunk.
3. **Main Game Logic**:
    - Manages game flow, including player inputs, AI decision-making, scoring, and win/lose conditions.

3.2. AI Implementation

The AI agent, was implemented with two primary modes of operation:

1. **HUNT Mode**:
    - AI randomly selects a cell using a checkerboard pattern for efficiency.
    - Switches to TARGET mode upon hitting a ship.
2. **TARGET Mode**:
    - The AI targets adjacent cells to the last hit to sink the ship efficiently.
    - Determines the ship's orientation after two consecutive hits (horizontal or vertical).
    - Prioritizes adjacent cells based on proximity and logical progression.

3.3. Scoring and Winning Mechanism

- **Real-Time Scoreboard**:
    - Displays moves made and ships destroyed by both players.
- **Game Completion**:
    - Declares a winner once all ships of one player are sunk.
    - Displays final statistics, including the number of moves and ships destroyed by each player.

---

4. Challenges Faced

1. **AI Complexity**:
    - Designing a robust AI that transitions smoothly between HUNT and TARGET modes while ensuring efficiency required careful debugging and optimization.
2. **Board Visualization**:
    - Displaying two boards side by side with clear visual indicators for hits, misses, and ships required thoughtful design and formatting.

---

5. Results and Features

- **Core Features**:
    - Interactive gameplay with human vs. AI.
    - Smart AI with strategic decision-making.
    - Real-time scoreboard for tracking moves and destroyed ships.
    - Clear game completion mechanics with winner declaration.
- **Game Output Example**: During gameplay, the boards and scoreboard are displayed as follows:

Enter coordinate to attack (e.g., B6): d2
You missed.

AI attacks A3
AI hits your ship!

AI attacks B3
AI hits your ship!
AI sank your Cruiser!

AI attacks A5
AI missed.

```
Your Board:                          AI's Board:
   A  B  C  D  E  F  G  H  I  J         A  B  C  D  E  F  G  H  I  J
 1 x  x  x  x  x  .  o  .  o  .       1 o  .  .  o  o  o  .  .  .  .
 2 x  x  x  x  o  .  .  o  .  o       2 o  .  .  o  o  o  .  .  .  .
 3 x  x  x  o  o  .  o  .  o  .       3 o  .  .  o  o  o  .  .  .  .
 4 x  x  x  .  .  o  .  o  .  o       4 o  .  .  o  o  o  .  .  x  .
 5 o  .  .  .  .  .  o  S  S  .       5 o  .  .  o  x  x  .  .  x  .
 6 .  o  .  o  .  o  .  o  .  o       6 o  .  .  .  o  o  .  .  o  .
 7 .  .  o  .  o  .  .  .  o  .       7 o  .  .  o  o  x  .  .  x  .
 8 .  .  .  o  .  o  .  o  .  o       8 o  .  .  o  o  o  .  .  .  .
 9 .  .  .  .  .  .  o  .  .  .       9 .  .  .  .  o  o  .  .  o  .
10 .  o  .  o  .  o  .  o  .  .      10 .  .  .  .  x  .  .  .  .  .

=== Scoreboard ===
Player Moves: 39   Ships Destroyed: 0
AI Moves:     47   Ships Destroyed: 4
==================
```

---

6. Future Enhancements

1. **Improved AI Strategy**:
    - Incorporate probability maps to predict ship locations in HUNT mode.
    - Add machine learning for adaptive AI based on previous games.
2. **Multiplayer Mode**:
    - Allow two human players to play against each other.
3. **Enhanced Visualization**:
    - Implement a graphical user interface (GUI) for better interactivity and visuals.
4. **Game Variations**:
    - Add variations like Salvo Mode (multiple shots per turn) and Fog of War (hidden boards).

---

7. Conclusion

The BattleC project successfully implements a challenging, interactive game of Battleship with a competitive AI opponent. By combining efficient algorithms, object-oriented design, and engaging gameplay mechanics, the project demonstrates the principles of software engineering and game development effectively.