Programming with C/C++

Academic year: 2024-2025, Fall semester

Programming assignment

Battle C

Implementing AI and Strategy in a Classic Battleship Game

Student name: Bilal Bilican
Student number: 2082071

* Cover page does not count towards the final number of pages.

1. Overview of project results

Use this section as a summary of the report.

- **Briefly outline whether and to what extent you achieved the goals of this project.**
  The primary goal of this project was to develop a Battleship game that is text-based or with a GUI interface that includes both a human player and an AI opponent. I achieved this objective by successfully implementing functionalities, such as ship placement, shooting mechanism, scoring mechanism, turn-taking and AI algorithm to my fully working text-based game.

- **Outline your idea/the logic behind the solution.**
  The game uses classes to separate parts like the board, player, ship and score. This makes it easy to manage each piece separately and to update or add features later on. The AI agent was made to remember where it hit a ship and then focus on nearby cells to sink the ship.

- **Briefly outline what were the challenges.**
  The challenges I faced where a few. Firstly it took me a long time to find out how to implement an error message and ask for new input if the user entered coordinates that were outside the grid of the playboard. Also keeping track of bombs for both players after every hit or miss and making sure the game ends when bombs run out. Lastly I worked on the implementation of the AI algorithm to avoid random guessing by applying hill climbing.

- **Briefly outline how you organized your code.**
  I divided my project into 5 classes and 1 main cpp:
  - Board.h → handles the game board and ship management
  - Ship.h → info about each ship
  - Game.h → manages the overall game flow, including turn-taking
  - Score.h → to keep track of scores and bombs
  - Player.h → for managing human and AI agent
  - Battleship Main.cpp → to start the game

- **Briefly outline how you organized your time and how you worked.**
  I tried to do 1 task every week, but some tasks such as making the AI agent smarter by implementing an AI algorithm took me more than 1 week.

* Cover page does not count towards the final number of pages.

2. Tasks and objectives

- **Present your solution to each of the tasks.**

Task1

I implemented the Board class that manages a 10x10 grid, initializes it with empty cells (.), and includes a method to display the board in the terminal with labeled rows and columns. The Ship struct represents each ship, storing its name, size, coordinates, and sunk status, with a constructor to initialize these attributes. The Score class tracks gameplay stats, including the number of moves, bombs remaining (starting at 100 for each player), and destroyed ships, with methods to update these values during the game.

Task 2

I implemented functionality for positioning ships, handling attacks, and checking the game state. The placeShip method makes sure that ships can be positioned on the board without overlapping or going out of bounds, and ships are marked on the grid as 'S'. A placeRandomShip method was also added for automated ship placement, which randomly selects valid positions for ships. To handle gameplay actions, the attack method checks if a cell contains a ship and updates the grid to reflect hits ('X') or misses ('O'). It also prevents players from attacking the same cell multiple times. The checkDestroyedShip method determines if a ship is completely destroyed by verifying that all its coordinates have been hit. Finally, the allShipsSunk method checks if all ships have been destroyed, signaling the end of the game.

Task 3

I implemented a turn-taking mechanism where players alternately attack each other's boards. Using the Player class, the takeTurn method validates input coordinates, ensures they are within the grid, and processes the attack. If a hit occurs, the player continues their turn until they miss or destroy a ship, as determined by the attack method and the checkDestroyedShip function. The main game loop in the Game class alternates turns between the human and AI players, checking after each turn if all ships are sunk using the allShipsSunk method. If all ships are destroyed, the game ends, and the winner is announced.

Task 4

I implemented a scoring mechanism and functionality to determine the end of the game. The Score class tracks the number of moves made, bombs remaining, and ships destroyed. After each turn, the displayScore method is used to show the current player's score, including moves taken, bombs left, and a list of destroyed ships. The game checks for a win condition using the allShipsSunk method, which determines if all ships on the opponent's board have been destroyed. If the AI's ships are all sunk, the human player wins and if the human's ships are all sunk, the AI agent wins.

Task 5

I implemented an AI player named "A. Smith" to make the game intelligent. The AI Player class inherits from the Player class and introduces functionality for decision-making. It keeps track of previously attacked cells using a 2D attacked array and prioritizes subsequent attacks using a targets list. If the AI hits a ship, it adds adjacent cells to the targets list using the addAdjacentTargets method, ensuring smarter, focused attacks to sink the ship efficiently. When no priority targets exist, the AI selects random coordinates that haven't been attacked yet. The AI's turn is managed by the takeTurn method, which handles hits, misses, and destroyed ships.

- **Show some parts of your code and link it with tasks.**

- To improve the visibility of hits, misses and ship placements, color coding was added.

```
const string RED = "\033[31m"; //color for hit
const string BLUE = "\033[34m";  //color for miss
const string GREEN = "\033[32m";  // color of the ships
const string RESET = "\033[0m";
```

```
// use of colors to highlight hits, misses and ships
                if (grid[i][j] == 'X')
                    cout << RED << grid[i][j] << RESET << " ";
                else if (grid[i][j] == 'O')
                    cout << BLUE << grid[i][j] << RESET << " ";
                else if (grid[i][j] == 'S' && showShips)
                    cout << GREEN << grid[i][j] << RESET << " ";
                else
                    cout << ". ";
            }
            cout << endl;
        }
    }
```

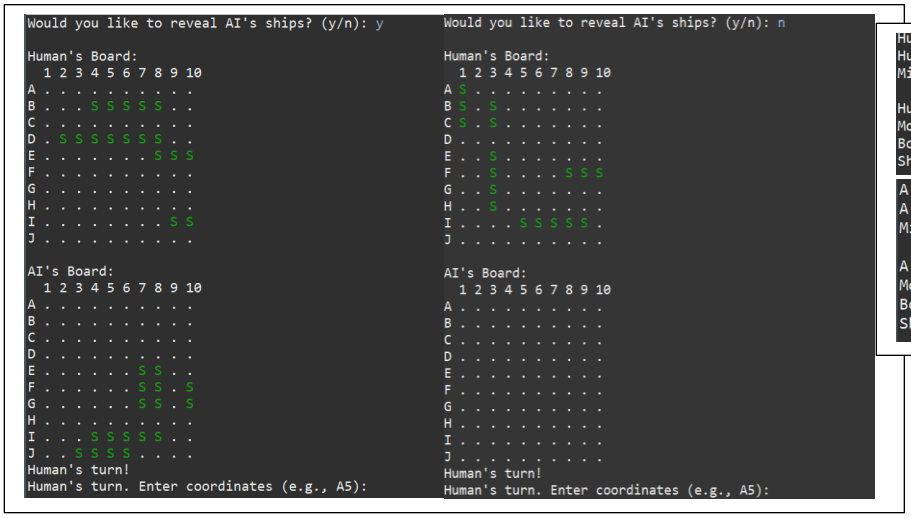
- Scoring mechanism to keep track of the moves, bombs and destroyed ships.

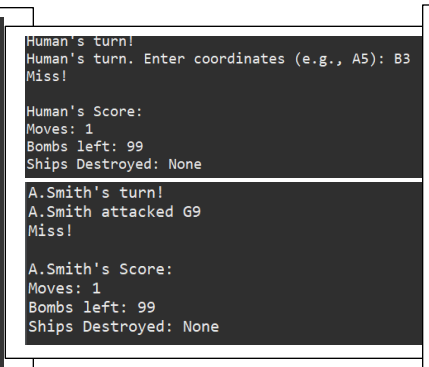
```
const int STARTING_BOMBS = 100; //each player starts with 100 bombs


// score class to keep track of the moves, bombs and destroyed ships
class Score {
public:
    int moves; // number of moves taken
    int bombs; // bombs remaining
    vector<string> destroyedShips;

    // constructor initializes moves and bombs
    Score() : moves(0), bombs(STARTING_BOMBS) {}

    // increase the number of moves and reduce bombs
    void incrementMoves() {
        moves++;
        bombs--;  // Decrease bombs with each move
    }
```

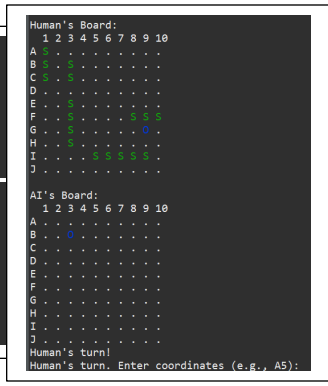
```
    // display the player's score on the screen after every move
    void displayScore(const string& playerName) {
        cout << "\n" << playerName << "'s Score:" << endl;
        cout << "Moves: " << moves << endl;
        cout << "Bombs left: " << bombs << endl;
        cout << "Ships Destroyed: ";
        if (destroyedShips.empty()) {
            cout << "None";
        } else {
            for (const auto& ship : destroyedShips) {
                cout << ship << " ";
            }
        }
        cout << endl;
    }
};
```

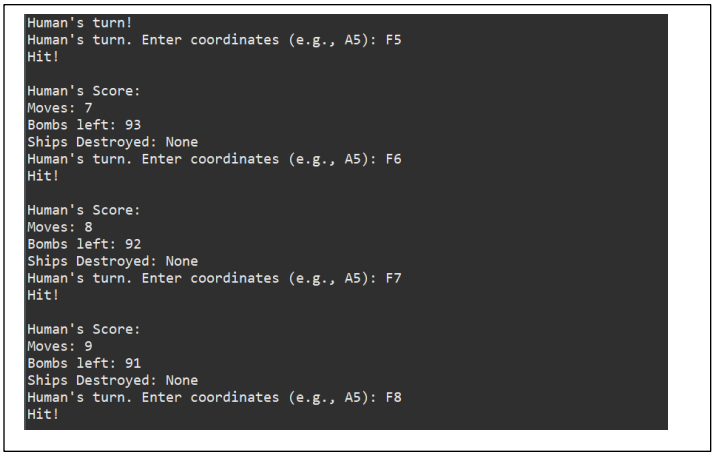
* Cover page does not count towards the final number of pages.

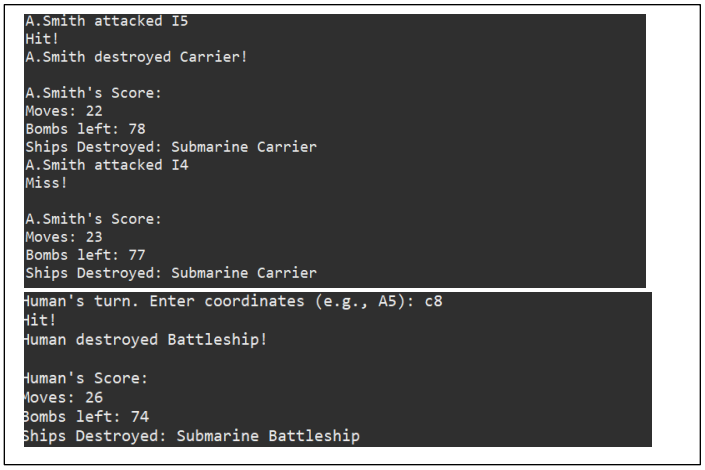
- **Show some screenshots of your program's execution.**

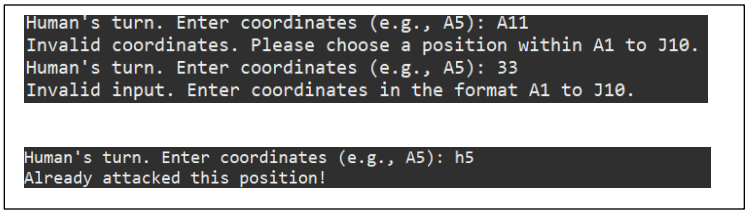


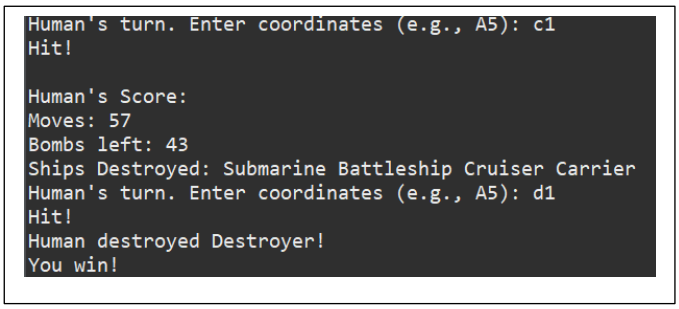


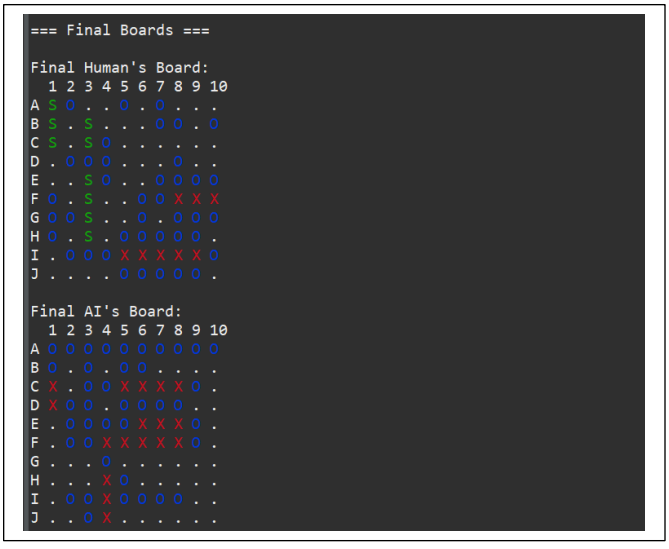












* Cover page does not count towards the final number of pages.

3. Challenges

In this section, describe the challenges you faced and how you overcame them. You can use a bullet list, a table, or a free text. Or you can organize this section into subsections:

3.1. Challenges solved

I solved all the challenges mentioned in section 1.

3.2. Challenges addressed but not solved

I could not solve the challenge to end the game if players reach a bomb count of zero. Almost every game I played finished before the bomb count reached zero, so I think it's not such a big issue for the gameplay.

3.3. Challenges not solved (neither attempted)

Before starting with the text-based interface I tried the GUI, but I did not get further than displaying the two boards with the ships placed.

4. Discussion and future work

- **Present additional topics that were not covered in the previous sections.**
  I would like to add a GUI where players can click on cells to place ships and shoot. This would make the game more interactive and fun.
- **Discuss the efficiency of your code.**
  I think my code is efficient for a small, text-based game on a 10x10 board. However, I noticed that the AI agent could be more efficient, because it sometimes takes longer to find and sink ships, especially if they are placed diagonally.
- **Present what can be improved in your current version or what new functionalities can be added.**

- Adding a feature to automatically end the game when a player runs out of bombs
- Including different modes such as: salvo mode and fog of war
- Improving the AI agent by implementing a neural network

* Cover page does not count towards the final number of pages.