Battleship Project – 200 (+25) Points

Project Due Dates:

- Design Document - **Monday, November 8th, 2021**
- Final Report and Project - **Monday, December 6th, 2021**

Project Description

You will work in groups of 4 students to complete an implementation of a terminal-based battleship game.

Battleship is a common kid's game involving two players; look here for more details https://en.wikipedia.org/wiki/Battleship_(game). The game starts with each player placing five ships of different lengths on a grid of size 20x20: an aircraft carrier of length 5 cells, a battleship of length 4 cells, a submarine of length 3 cells, a destroyer of length 3 cells and a patrol boat of length 2 cells. Each player then takes turns selecting and attacking grid cells, attempting to successfully hit all their opponents' tiles before their own ships have been sunk. When a player takes a turn, the player selects a target, and is notified if they hit or miss a target. If any ships are sunk in the process, they are notified. In a players turn; they are only allowed to shoot once.

Anytime a ship has all their cells hit, it is "sunk", and the shooting player is notified that they sunk a ship and the type of ship.

The game is finished when all the ships of one player have been sunk.

Project Requirements

For this project, you will have several requirements. Complete each requirement to earn the points for that section.

- Total Possible Points – **200 (+25 Bonus)**

Design Document – Total Points – 20

- **Note: This is the only requirement with an earlier due date! (Monday, November 8th, 2021).**
- This will give me a chance to look over your designs, and if there are any major issues, point them out so you aren't stuck with a bad design late in the game.

To complete the requirements for the design document, you must submit a document that does the following. First, discuss and brainstorm with your team the approaches you think you might take toward completing the project. What sorts of structures, enums, etc. might you want to use? What kinds of functions might you use? What approach do you want to take for your custom AI implementation (this doesn't have to be very complex). What is an initial plan for your game's save file?

After discussing your initial approach, write it up in a formal design document. By formal, I mean provide a title page including all group members names. In addition, be careful to use formatting and grammar which is professional and clearly shows that you are addressing the following issues.

Note: Your actual design is allowed to (and are likely to) change as you work through your projects! That's ok! This is just to help formalize your thoughts and give you a good direction to start your implementation.

- Discuss what custom data types (structs or enums) you might use. **(5 Points)**
- Discuss the general flow of your program. Make sure to discuss the different paths through your program. **(5 Points)**
    - For example:
        - 1. Start by showing the new game screen
        - 2. Create two AI players
        - 3. Simulate each AI player taking their turn until one wins
        - 4. Print the winner
- Discuss what approach your custom AI will take. This should be roughly the same level of complexity of the Advanced AI, but doesn't need to be anything too crazy. **(5 Points)**
- Discuss your file format for saving the game data **(5 Points)**
    - What sorts of data should be stored in this file to correctly reload your game?

Battleship Implementation – Total Points - 150 (+25 Bonus)

It goes without saying, but you can only complete this project using C.

Title Screen – 10 Points

To complete this requirement, you must:

- Display a welcome message when starting the game
- Display a short description of the rules for how to complete the game
- Query a player for the choice to start a new game or continue a previous game

New Game – 10 Points

To complete this requirement, you must:

- Query the user for the number of players
    - 0 Players – A game is fully simulated between two AI agents.
        - For each AI agent, as which version of the AI should be used
    - 1 Player – Player 1 is controlled be a user, and player 2 is controlled by the AI
        - For the AI agent, query which version of the AI should be used
    - 2 Players – Each player is controlled by a user
- For each player, read in their name for use when printing whose turn it is or who has won
    - For AI opponents, name them whatever you'd like
- Query the user for the size of the board
    - The minimum size should be enforced (i.e. allow valid game configurations)
- For each player, either real or AI, you should query if their ship configuration should be random or manual

Load Game – 10 Points

To complete this requirement, you must:

- Design a file format for saving the state of your game
    - i.e. All data you would need to go from a game actively running, to restarting the game, to reloading it back to where it was before
- Allow at the main menu screen to select loading a game
- Allow the user to entire the name of a file to load (the saved game file)
- Load that file and continue the game from where the saved game left off

Save Game – 10 Points

To complete this requirement, you must:

- Design a file format for saving the state of your game
    - i.e. All data you would need to go from a game actively running, to restarting the game, to reloading it back to where it was before
- During any game run with real players (1-2 player modes, not 0 player), instead of taking a turn, allow the player to enter an input which allows them to save the game
- At this point, if you were to exit the game, you should be able to run your program again and reload back to the same state the game was in using that save file

Random Configuration of Players Ships – 10 Points

To complete this requirement, you must:

- Randomly place a player's ships onto their grid
- Ships cannot overlap!!
- The ships included are:
    - One ship of length 5 (aircraft carrier)
    - One ship of length 4 (battleship)
    - Two ships of length 3 (destroyer and submarine)
    - One ship of length 2 (patrol boat)
- Ships must fully be contained on the grid
- Ships must be place horizontally or vertically
    - No diagonal ships

Manual Configuration of Players Ships – 10 Points

To complete this requirement, you must:

- Allow a player to manually place all their ships
- Starting with the longest ships and working toward the smallest
- Allow a user to specify the starting cell and the ending cell of the placed ship
- Ships cannot overlap!!
- The ships included are:
    - One ship of length 5 (aircraft carrier)
    - One ship of length 4 (battleship)
    - Two ships of length 3 (destroyer and submarine)
    - One ship of length 2 (patrol boat)
- Ships must fully be contained on the grid
- Ships must be place horizontally or vertically

- No diagonal ships

Basic AI – 10 Points

To complete this requirement, you must:

- Implement a very basic AI which moves in the following manner
- Randomly Select a grid cell
- If the cell has already been attacked, keep reselecting until the cell has not been attacked before
- Attack that cell

Advanced AI – 10 Points

To complete this requirement, you must:

- Implement a more advanced AI which moves in the following manner:
- Begin like the basic AI, attacking until a hit is discovered
- Then, continue attacking specifically that ship until it has been sunk, before reverting to the basic AI, random attack pattern
- The difference between this AI and the basic AI is that this advanced AI focuses on destroying discovered ships before continuing

Custom AI – 10 Points

To complete this requirement, you must:

- Implement a new AI scheme which you propose in the design document
- This AI should be different than both the Basic AI and the Advanced AI, although it is allowed to use concepts from either of the existing AI methods

0 Real Players Mode – 10 Points

To complete this requirement, you must:

- Implement the simulation of a game of battleship between two AI opponents
- This is a very good way to test the behavior of two different levels of AI
- In this mode, all turns should be automatic (AI), based on each AI player's level of AI
- It might be helpful to allow the manual placement of the AI players' ships as this could help with debugging AI behavior

1 Real Players Mode – 10 Points

To complete this requirement, you must:

- Implement the game of battleship between one real player, and one AI opponent
- The player should be allowed to choose if they want manual or random placement of their ships
- The player should always manually select their target
- The AI should always automatically select their target based on their AI level

2 Real Players Mode – 10 Points

To complete this requirement, you must:

- Implement the game of battleship between two real players
- Each player should be allowed to choose if they want manual or random placement of their ships
- Each player should always manually select their targets

Print Current Player's Grid – 10 Points

To complete this requirement, you must:

- Implement the drawing of the current player's grid on each turn that a player takes
- This should be formatted to convey information neatly and clearly
- This grid should show a detailed view of all a player's ships, hits which have been made on their ships, etc.
- A detailed grid should only be shown for a given player, never to their opponent (to avoid giving away key information)
- This should be shown at the end of the game after a winner has been found

Print Opponent Player's Grid – 10 Points

To complete this requirement, you must:

- Implement the printing of the opposing player's grid on each turn that a player takes
- This should be formatted to convey information neatly and clearly
- This grid should show a less detailed view the grid
    - Only misses, hits, and non-selected squares
- Only show the less detailed grid to the opposing player as it spoils nothing about ship location

Declare and Print Winner – 10 Points

To complete this requirement, you must:

- Detect when a player has won the game
- The detection of a winner should take place after every player turn
- Print a message saying which player has won the game using their entered name
- Print out the detailed view of each game board

Final Report, Demo Video, Individual Contribution Weights and Source Code – 30 Points

After you have completed your implementation of the project, you should submit the following:

1. A final report, written as a group
2. A demo video made as a group
3. The source code of your group's final implementation
4. Weightings of the contribution of each group member (submitted individually), out of 100 total points

   For example, if two members did 60 percent of the work, and two did 40, the final weights should be:

1. Member 1: 30
2. Member 2: 30
3. Member 3: 20
4. Member 4: 20

Your final report and deliverables should address the following!

- Re-address each of the issues from the design document. If a particular issue has not changed in its design (for example, you are still using the initial file format design), discuss why your approach was successful. If the design did however change, discuss what changes you made, and why you made them. **(10 points)**
- Discuss each team member's individual contributions to the project. I don't mean their point contributions; I mean what parts of the project did they complete. Be specific. **(10 points)**
- Provide a video demoing each of the required features. This does not need to be a deep dive into the code, however, it should show each of the required features working to earn points in that category. **(10 points)**

Battleship Bonus Variants – 25 Points

In addition to the base version of the game, your group can optionally implement **one** of the following variants of the base game for bonus points. Each should be implemented as a separate mode selectable from the new game menu. If you complete the mode in its entirety, you will be awarded the points. You should only focus on this after completing the main project requirements!!

If you choose to complete one of these modes, include it in the final report and the demo video.

Recon Mode – 25 Points

In this mode, a player is allowed to perform a plane recon maneuver instead of firing a shot to obtain information on whether any ships in a 3x3 grid area.

Each plane is only allowed to recon two cells before they must land on the aircraft carrier to refuel. If the plane cant refuel (the aircraft carrier has been sunk) it crashes. Each plane should start on a separate cell of the aircraft carrier.

If the opposing player targets a cell containing a plane, that plane is destroyed. As such, players are allowed to select any cell for attack (even if they have already attacked that cell).

Salvo Shooting Mode – 25 Points

In this mode, players shoot three shots per turn. Rather than telling if all the shots hit, they are only notified of the number of shots which hit. The player is still notified of any ships which have been sunk during their turn.

For this mode, you need to somehow augment the visualization to signal the probability that a hit has taken place for a given cell, as simply marking a cell as a hit or miss is incorrect.

The following set of ships for this game mode includes two ships of length 2, two ships of length 3, and two ships of length 5. You will need to support this configuration of ships as well as the original set.

Graphical Display – 25 Points

Warning: Easily the most difficult of the battleship variants, as this mode requires the integration of a third-party library to handle the creation of a window, as well as the drawing of images on the screen. I am including it for any brave groups who wish to attempt this, however, be warned that it requires advanced concepts, so attempt carefully.

A recommended library for this project would be SDL 2.0: https://www.libsdl.org/

In this mode, no changes to the rules of battleship are required. Instead, you must display the active battleship grids via a graphical display as a game would, rather than displaying them in a terminal.

For each cell, draw an image representing the type of value of each cell. For instance, for a water cell, draw an image of blue water. For ship tiles, draw an image of a ship in that location. For a hit, draw an image of a burning ship or fire.

An example of what the grid might look like is shown below: