Konstantin Novichenko

Francesco Pecora

Sarthak Taneja

CSC330

Professor Richard Weir

CSC330 Final Project - Design Document

Fundamental Concept

Our project is a Javafx implementation of the two player board game Battleship. The game has a graphical user interface with several optional features, a leaderboard system, a functional saving and loading state feature, and an AI competitor with two, manually selectable levels of difficulty.

Explanation of Game Implementation

Battleship is played by placing ships on a board at the beginning of the game, and then attempting to locate and destroy all of the opposing player's ships. Our implementation allows a player to compete against an AI and incorporates a GUI that shows the player's board and a representation of the AI's board. Before starting a game, a player can choose which of the available ships he or she would like to place, and in what orientation. Once the player is finished placing his or ships, the game begins; players can click on coordinates within the corresponding enemy's grid representation to attempt hitting the enemy's ships; three colors—blue, paleish red, and deep red—are drawn on the enemy's tiles to represent whether or not a potential hit was a miss, a hit, or if the ship had totally sunk, respectively. Color representations of such a kind are also visible on the player's board as well for wherever the enemy AI attempts to hit.

There's also a main menu implementation that runs before the game is started. Through this main menu, the player is able to select various options, such as specifying which difficulty level of AI they would like to play against, and two color options for the game board (essentially emulating a dark/light mode that is so common in many applications). The player is also able to access a brief tutorial explaining the rules of the game, in addition to the leaderboard from the main menu.

Lastly, there's the aforementioned leaderboard. It's self explanatory in function, but how it's implemented is as follows: "scores" are determined by a metric which weighs the number of turns taken to complete a game session alongside the time taken to achieve this win (there's a persistent timer tracks this). The leaderboard is self sustaining, and updates itself automatically.

Development Model

Right from the start, we decided that the Agile Development Model was the best fit for what we wanted to accomplish. Given the distance-oriented development process that we were forced to adopt, the self-organization principle of development that the Agile model implements appealed to us. Additionally, the Iterative Model of software design is especially suited for efficient game design (Valve Software is famously a proponent of it for instance), and as a result Agile's explicit implementation of the iterative design process was further appealing to us.

More concretely, our design process involved routine meetings, seperation of labor, individualized prototyping, and, most crucially, iterative phases of development. We started at the most fundamental level with the most basic of goals: creating design proposals and identifying potential classes and objects we would need to implement, with the ultimate intention of first reaching a functional prototype.

After this period, we were able to discreetly identify which workloads were both readily categorizable and could be split among three people. In the end we ultimately arrived at Game logic, AI logic, and GUI/User Friendly design as the three identifiable areas of work to be done, and assigned them to each group member accordingly.

Our expectations of what we could and couldn't implement constantly evolved as the project developed. Initially, our ambition was to implement sound effects and more comprehensive graphics (for ships, attacks, etc.). These two proved problematic in their own unique ways to implement; and as a result, we decided to forgo these, and instead focus on a streamlined, more easily polishable implementation of the game.