Maxim Baduk, Alex Vaziri, Brandon Hong

MS1: Alpha — Progress Report

November 7 2019

CS3110

Vision

Our current vision for the system is a in-terminal port of the classic game “battleship”. Eventually, we want it to have a graphical interface based in the terminal, which the user would be able to interact with to play the game. For example, the main game screen would show a 10x10 grid which represents the game board. This would be drawn to the terminal by printing symbols (underscores and vertical lines) such that they make the shape of a square. Then, within the square, different symbols drawn to the screen would represent the places where the player either missed, or hit the enemy ship. There will also be various features that make the game more interesting to play. Color will be used to better show where the player has and hasn’t shot before. There will also be an AI that plays against the player, and fire automatically at the player’s board. Overall, our vision hasn’t changed significantly from when we designed our initial roadmap.

Summary of Progress

During this sprint we were able to complete the tasks that we outlined in our roadmap for MS1. Namely, we designed the board module as a matrix of (hit/unhit) types, and designed a system for modifying the board depending on whether the player shot at an undamaged ship, damaged ship, empty spot, or previously shot spot. Although we do not yet have the graphical interface,

that's fine since we did not require that for our first sprint. Rather, the game currently includes a text-based play format. The player can enter the coordinates they want to fire at as text, and the game functions normally from there. There is also not yet a method for placing ships onto the board, rather the player must just play with a pre-made board that we hardcoded. There is also not yet a win condition, which will be added in a later sprint. We are also working to solve an issue with one of the libraries ("curses") not working on machines running a VM.

Activity Breakdown

Maxim Baduk - My first task was to define scopes of work for each member of our team. Thus, I made a document which described our vision for the first sprint of this project, what I expected to come out of the sprint, the tasks each team member was responsible for, and the timeline on which those tasks were to be completed. Then, I formed out git repo and make wrote the starter files that allowed the other team members to begin their work, which included the makefile which is used to compile work, and a small example project (similar to the starter code for our class assignments). Then, once the other members completed their tasks, I reviewed their work, merged everyones code, tested the functionality, and worked on integrating their parts and making improvements to the playability of the game. I also defined a scope of work for our next sprint, so that we could start on it quickly.

Alex Vaziri - I was responsible for designing the game board module, which included the game board itself, data types for the game board elements, and methods for changing the board depending on what command was passed to those methods (such as the "fire" command). I also developed a method to parse user input and transform those commands into an internal

“command” variant. Finally, I wrote definitions for the “ship” data type, and make a demo board with ships placed on it, so that the player would have a board to play on at this stage.

Brandon Hong - My job was to make the main game loop which is the entry point into our program. Thus, I designed a loop which presents a welcome message to the player with basic gameplay instructions, and then prompts the player for input. Based on that input, the loop parses and then modifies the internal game board structure, and provides verbal feedback to the user on the now-modified structure of the board. I also designed a method which displays a simplified graphical version of the game board to the standard output. Finally, I made a method that catches exceptions thrown when the user inputs commands that are malformed or unrecognized by our program.

Productivity Analysis

During the first week of our sprint we were still battling with the task of A5, so our initial progress was generally quite slow. Though we did meet during that time, we agreed to begin working more seriously on the project once A5 was out of the way. This worked out quite well. Once the second week rolled around we had already laid the foundations for our project, and a few days later most of the work we wanted to accomplish for the sprint was already done. In retrospect, we could’ve taken on more work for this sprint, but we wanted to keep the workload relatively light since we didn’t know exactly how long it would take to lay the groundwork for this project. Luckily, once we completed the work required for this sprint, we began thinking about and writing the basis for the tasks outlined in the next sprint. Thus, we have a good foundation laid out for the coming weeks.

Scope Grade

Technically speaking, we accomplished everything that we wanted to accomplish in this sprint, including designing the game board, defining different types for camels (ships), unhit/hit board entries, etc. However, we also underestimated how much we were capable of accomplishing within this time, so we kept our workload for the first sprint rather low. As such, we think we accomplished an amount on par with the "good" scope that usually comes with our assignments for this class. Overall, each team member put in a good amount of effort into the project thus far, and we cooperated well together to get our tasks done, which included a lot of meeting together and discussing (and sometimes arguing) about what we wanted the game to look like and how to best implement certain features.

Goals for Next Sprint

For the satisfactory scope of our next sprint we want the game to feature the actual terminal-based graphical interface that we originally envisioned. Rather than having the game reprint a symbolic representation of the board every turn, we want the terminal to feature a persistent, dynamic, full-screen image of the board built from different symbols (similar to the ascii-art style) which changes based on user input. Furthermore, the user input should no longer be textual. Rather, the user should simply be able to aim their "cannon" by moving an on-screen cursor using the arrow keys, and should be able to "shoot" by pressing enter. Thus, there will no longer be a need for any parsing of text input. The good scope will include a pregame mode in which the user is able to place their ships onto the board. We expect this stage to be quite difficult to implement, since it must also be correctly displayed in the terminal-based graphical

interface. Finally, the excellent scope will include the ability for the user to rotate ships when placing them onto their board in this “pre-game” mode, and will include a win condition that allows the user to win the game.