Maxim Baduk, Alex Vaziri, Brandon Hong

CS 3110

MS2-Beta Progress Report

November 21, 2019

Vision

Our vision for our system is still the classic game battleship. It has evolved from previous sprints slightly, as we better understood what is and isn't feasible for us to accomplish. While most of the features we wanted to implement are still in the plan, we made up our mind about the uncertainties that we had previously. For example, the game will now only be one-player, and the player will play against an ai that we design. Also, we decided to change the layout of the board, and have a true "square" board (rather than the rectangular one we had previously), with information panels on the side. As we developed the in-terminal gui, we developed a system to track the running time of each call to the render loop, so we plan to use this to add animations to the game (as we have already done with the blinking cursor). Finally, we plan to add details which will enhance the "feel" of the game such as the ability to rotate ships, a pregame menu, and animations.

Summary Progress

Since the last sprint, we accomplished our main priority of transforming the game's graphical component from a text-based scrolling grid which gets reprinted for every run through the game loop, to a true persistent gui. The updated gui includes a blinking cursor which responds to keyboards inputs (wasd keys for movement, "p" to place a ship, and "f' to fire at ships once all five of them are placed). Thus, we eliminated the need for textual input from the users, and thus the player need not remember any nebulous commands to play. Overall the feel of the game is greatly improved due to the amount of detail we put into making the gui. The blinking cursor wraps nicely around the edges of the board once the player reaches the edge, and highlights in red to indicate a spot where shooting is impossible (since the user already shot there). Finally, we added a placement phase where the player can place their ships beforehand, so that they may design their own board. In order to implement the animation features, we

had to introduce function-timing functionality which measures the time it takes for each call to the render loop to terminate. We then use this value to compute a “delta time” which gets passed into subsequent calls to the render loop, which makes animations possible and standardizes them regardless of the player’s computer’s refresh rate or processing speed.

Activity Breakdown

Maxim Baduk:

- Developed the “display.ml” file which handles all of the visual features of the game.
- Developed the render loop timer which makes animations possible, and standardizes animation speed across computers.

Alex Vaziri:

- Developed the input system in “command.ml” which registers key inputs from the user, and translates them to internal game commands.
- Redesigned command responses in the main game loop to account for the new mode of play (no longer textual-based commands, but rather keyboard-press combined with cursor position).
- Added color responses to cursor positions, so that the cursor highlights in red when the player moves the cursor to a position where they cannot shoot.

Brandon Hong:

- Developed the placement system which allows the user to place their own ships before playing.
- Tested durability of the system, and made it resilient to unknown inputs such as invalid key presses, and resizing of the terminal window.

Productivity Analysis

Overall we are happy with our team productivity in this sprint. At the very start of the spirit we had a meeting where we discussed everything we wanted to get done, and when we wanted it done by. We decided that since we would be completing A6 during the first week, we would only make our satisfactory scope due then, which included the graphical interface and inputs. Then, during the second week we would complete the good and excellent scopes, which included placement phase, rotation of ships, and the

win condition. We accomplished most of these features, excluding a few in the excellent scope, by the assigned dates. Thus, our estimates were quite accurate, and our team was able to deliver upon what we agreed to in the beginning of the sprint.

Scope Grade

We believe we accomplished enough to warrant the good scope. As stated above, we were able to complete everything we agreed upon except for a win condition and the rotation of ships. That corresponds to what we defined in our "good" scope. Although we finished all the remaining features, implementing them took approximately the time that a regular assignment in this class takes, around 8-10 hours of work per person. Every member of our group showed up to meetings, agreed upon tasks that they would take on for each scope, and ultimately implemented those tasks by our set deadlines (except for the excellent scope ones).

Goals for Next Sprint

Satisfactory:

- Implement a win condition, along with a "win phase" which will allow the user to see their "score" and exit the program or play again.
- Implement an ai to actually play against the player (so that the player will not simply be playing against themselves). Includes implementing turns and dual-board display.
- Visual placement of ships (meaning that the cursor takes up as many places as the ship you are currently placing would take up on the board).

Good:

- Design a start screen with menu, options for play phase and reading instructions.
- Design an info screen adjacent to main game board, which displays helpful feedback and metadata.
- Add quantitative stats to win screen describing the game (steps to win, time, etc) with leaderboard. User should be able to input their name and save their score (non-persistent).

Excellent:

- Develop additional animations, screen light up effect on successful hit, “radar”-like board animation, etc.
- Add new modes of play with variable levels of ai difficulty, new fire modes (multiple-space fire).
- Persistent save states, written to external file with encoded output.