Arnav Parashar, Aryan Saju, Manvir Chahal, Shashaank Aiyer

CS 3110

20 October 2021

MS1 Progress Report

Vision:

Our current vision is to implement the popular board game “Risk” with OCaml handling all gameplay functionality and state and with a separate language being used for a GUI. We would like to create a fully interactive system that allows multiple players to enjoy this game on a computer. As of right now, we have mostly stayed on track with our original plan. We have laid out the foundation of the game in OCaml and are now working on building up.

Summary of progress:

We worked on building the core functionality. We created code that lets us take in a json file storing info about a risk board, and load that in as a board object able for us to use to play the game. We also created a player module for storing some basic information about a player. We also created a state module which holds all the information such as the current state of the board, the players of the game, the current player whose turn it is, and what mode the player is in. This state class allowed us to make our main method basically use a state machine to play the game, prompting the user with requests based on the state. We are able to play the game of risk with most functionality built in. The only functionality we are missing is the ability to receive cards upon winning an attack, the ability to trade in cards for more troops, the continental bonus, and enforcing that a fortify is done between connected countries (currently our fortify lets you move troops to any country on the map, regardless of connection, as long as you own both).

Activity breakdown:

Arnav Parasher:

- Helped implemented “board.ml” which stores information about the board and also manipulates the troops on the board
- Helped implement “state.ml” which stores information about the state of everything involved in the game, the board, the players, the mode the current player is on. Also allows for manipulation of the game
- Helped implement “main.ml” which is responsible for running the game via a state machine using the State module
- Hours Worked = 15

Aryan Saju:

- Helped implemented “board.ml” which stores information about the board and also manipulates the troops on the board
- Helped implement “main.ml” which is responsible for running the game via a state machine using the State module
- Hours Worked = 10

Manvir Chahal:

- Implemented “command.ml”, which handles parsing user input into usable commands for main
- Completed test cases “board_test_ml” which tested methods in “board_ml”
- Hours Worked = 7

Shashaank Aiyer:

- Implemented “attack.ml” which handles attacks between countries, dice rolling, and determination of results of each battle
- Helped implement “state.ml” which stores information about the state of everything involved in the game, the board, the players, the mode the current player is on. Also allows for manipulation of the game
- Helped implement “main.ml” which is responsible for running the game via a state machine using the State module
- Hours Worked = 12

Productivity analysis:

As an entire team, we were quite productive. We generally did pair programming whenever working on a module. We did a bottom up approach, starting with board.ml, then state.ml, then main.ml and command.ml. We did a great job of bouncing ideas off each other which made it easier to catch bugs as we wrote code. We accomplished what we had planned exactly, the game is currently playable as is, there are some aesthetic changes to be made but we had always planned to lay out the groundwork of the game for this sprint.

Scope grade:

We give ourselves Excellent Scope because we completed all of the requirements for both Satisfactory and Good Scope. We had originally required the completion of the card implementation for Excellent Scope; however, we realized that this is not a core component for the game which was the whole idea for MS1. Instead of completing the card implementation, we went above and beyond with our exception handling to ensure that the players cannot crash the

game. This was supposed to be completed in MS2, but we swapped this task with the card implementation. Therefore, we believe we deserve to receive Excellent Scope.

Goals for next sprint:

1. Satisfactory Scope: Implement card.ml that adds in cards players receive at the start of each phase. Implement continental bonuses.
2. Good Scope: Create the processes required to convert all the pertinent data for a GUI into a JSON file
3. Excellent Scope: Begin making the GUI and lay the foundations for further development