Arnav Parashar, Aryan Saju, Manvir Chahal, Shashaank Aiyer

CS 3110

3 November 2021

MS2 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 built the game in OCaml, started designing the GUI, and implemented the communication between our OCaml and Java programs.

Summary of progress:

In MS1, we focused on building the basic functionality of Risk, primarily the game setup, and all the states of moving through a turn. We did not implement any functionality to give players troop bonuses or determine if a player loses or wins. In MS2, we built in both of these features. We also wrote functions to write all of the game information to JSON files, since that is how we are transferring information to the GUI. We used the OCaml library Unix to set up a small server for communication between our OCaml and Java programs. We also began building the GUI in Java. We set up the map and are working on a function to color a country corresponding to which player owns it.

Activity breakdown:

Arnav Parasher:

- Began work on the socket communication code between OCaml and Java code
- Created a tool in java to help mark out which country is where on the board image
- Added Card code to State and Main
- Tested all to_json methods
- Hours Worked = 15

Aryan Saju:

- Implemented code for when a player dies.
- Tested all functionalities of game
- Debugged code
- Hours Worked = 10

Manvir Chahal:

- Began laying the foundation for the GUI
- Implemented the visual, country locations, player colors
- Helped to debug communication between computers for the GUI
- Hours Worked = 10

Shashaank Aiyer:

- Implemented card.ml, which creates specialty cards and determines when a player can successfully trade in cards for a bonus
- Implemented all of the to_json functions that write information across the game to json files
- Helped debug to_json methods
- Hours Worked = 10

Productivity analysis:

As an entire team, we were quite productive. We generally did pair programming whenever working on a module. We divided and conquered some parts, and coded together on others. 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 was already playable and we simply added more features. We also began the GUI which required learning on our end since none of us had much experience with building them. We did a great job with learning this new process to implement the GUI.

Scope grade:

We give ourselves Excellent Scope because we completed all of the requirements for Satisfactory, Good and Excellent Scope. We implemented card.ml and implemented continental bonuses. We created the processes for a JSON file and laid down the foundations for the GUI. Therefore, we believe we deserve to receive Excellent Scope.

Goals for next sprint:

1. Satisfactory Scope: Build on the foundations of the GUI with displaying information and changing based on updates in the game
2. Good Scope: Add troops, highlights, and other smaller details to the GUI
3. Excellent Scope: Creating communication between multiple computers