Bilkent University

Department of Computer Engineering

CS319 Term Project

CS319 1I-TM

Spring 2020

Implementation Report

Team Members

Rafi Çoktalaş

Zeynep Cankara

Kamil Gök

Efe Macit

Arda Gültekin

Instructor: Eray Tüzün

Teaching Assistant(s): Alperen Çetin

Contents

1. Introduction

1.1. Local Multiplayer

We implemented a local multiplayer game for 4 players. All the actions taken on the UI perfectly transition to the logic of the game.

1.2. Online Multiplayer

We implemented all the logic, server and UI components to have online multiplayer. Moreover, we have established all the functionalities of the game between server and logic but we could not do so between server and UI. Online multiplayer functionalities completely and solely work as a console application.

1.3. Login

We implemented a perfectly functioning login system to our game.

1.4. Help

We provide users with a help page that briefly explains the game. This page is intentionally kept short as we provide much greater documentation for our users.

1.5. Choosing Faction

We provided a faction selection page for all the players before starting the game.

1.6. Transforming Terrains

We provided all the functionality regarding terrain transformation. Moreover, we made it possible to be immediately followed up with a build a Dwelling action.

1.7. Building Dwellings

We provided all the functionality regarding Dwelling building.

1.8. Advancing on Shipping Track

We provided all the functionality regarding Advancing on Shipping Track.

1.9. Upgrading a Structure

We provided upgrading to trading house and stronghold functionalities regarding Upgrading a Structure.

1.10. Send a Priest to the Order of a Cult

We provided all the functionalities regarding Send a Priest to the Order of a Cult.

1.11. Pass

We provided all the functionalities regarding Pass.

2. Design changes

Only major design change in our implementation is that we choose FlowManager as a facade class between UI controllers and our entities. This made FlowManager a god class as it both controlled if the conditions are met for functions and notified the UI controllers. The latter functionality of the class is given to its rightful owner GameEngine class. Number of minor changes happened in classes to further guarantee to assign single responsibility for each class. Moreover, there are small changes in the method signatures and return types. The reason behind those changes is to make the best use of our facade classes.

3. Lessons learnt

3.1 Communication

In every decisionmaking process, we discussed until everyone is on the same page. This slowed the processes. When the design process ended, we noticed that we should better utilize our time. Thus, we started to plan our meetings beforehand in more detail, i.e., topic, purpose, goals and time. Planning of meetings helped greatly but instead of unanimity, majority could have been a better approach in the meetings in terms of time. Yet, even with the beforehand planning, we still experienced the lack of face to face meetings.

3.2 Design

3.2.1 Introduction of the Design Patterns

Introduction of the design patterns are the biggest difference in our design between iterations. Firstly, we started to decide on design easily as the problems we have about design are almost always explained in design pattern sources. Secondly, we clear the

redundancies in our design simply by following the definition and principles of the design patterns we have used. Lastly, we started to have less doubt about our decisions as we rely on the design pattern. Thus, we found more to invest in implementation.

3.2.2 General Principles

3.2.2.1 Single Responsibility

Each time we discussed something about implementation we have noticed that there were redundancies and violation of our design pattern choices. Therefore, we have gone over our design several times just to ensure to give each class only one indivisible part of the overall functionality. Doing so helped us clear redundancies both in code and in communication. The latter, even though it was not the intended outcome of the endeavour, has benefited us in terms of time as the meetings have become more efficient.

3.2.2.2 Composition over Inheritance

During our implementation process we have made changes favoring composition over inheritance. The driving force behind those changes was to be able to make behavioral changes at runtime. Moreover, the code have become more maintainable as the changes happened in smaller scale.

3.2.2.3 Regarding Undelivered Functionalities

Even though we have implemented a great portion of the game logic and a perfectly sufficient server. We failed to reach a level of adequacy in connecting server and UI. We underestimated all the connections between logic,UI and server. We should have had our work and design time allocation accordingly.

3.3 Documentation

3.3.1 Analysis

The analysis report of the game was the first address for every member having a question about the game. In addition to the general questions about the game, the analysis report also contained all the edge cases in the game. Thus, the comprehensive analysis documents we have in hand provided us with a great start and a guide for designing the overall system.

3.3.2 Design

Aside all the mentioned in 3.2, the design report enabled us to communicate with ease. To clarify, we asked each other questions about the parts we implement directly on the class diagram. This way every decision making process is resulted considering the whole system and without going over the source code over and over again. The design process took the most of our time but the benefits definitely outweigh the cons.

4. User Guide

4.1. Initial Screen

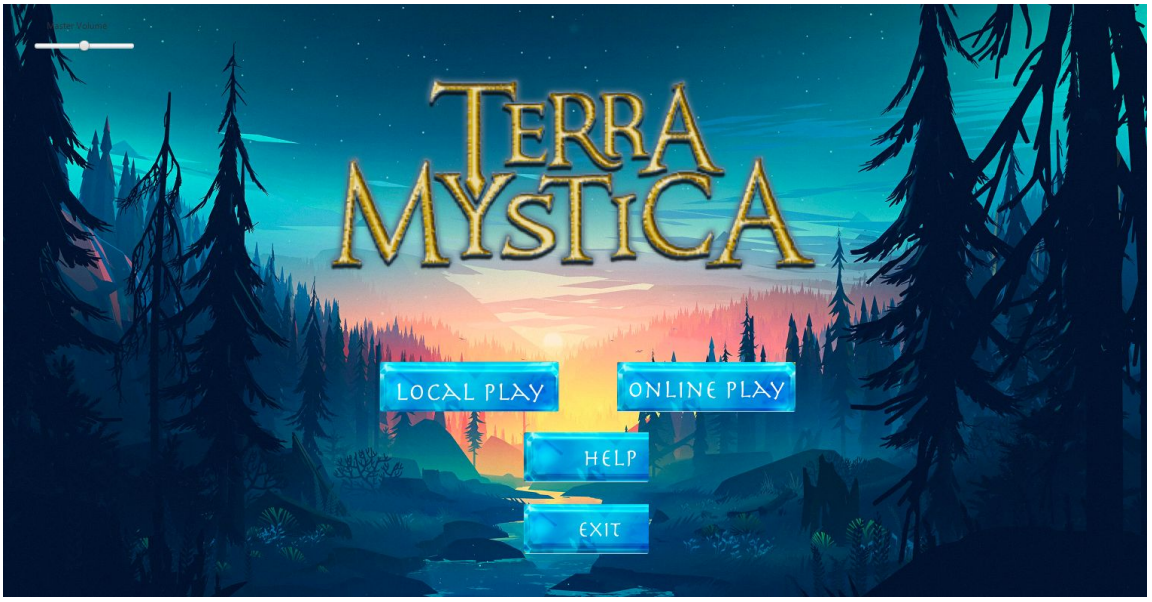

Figure 1: Initial screen

- Local play button takes the player to the Login Screen.
- Help button takes the player to the Help screen.
- Exit button terminates the application.
- Master Volume is adjustable from top left.

4.2. Help Screen

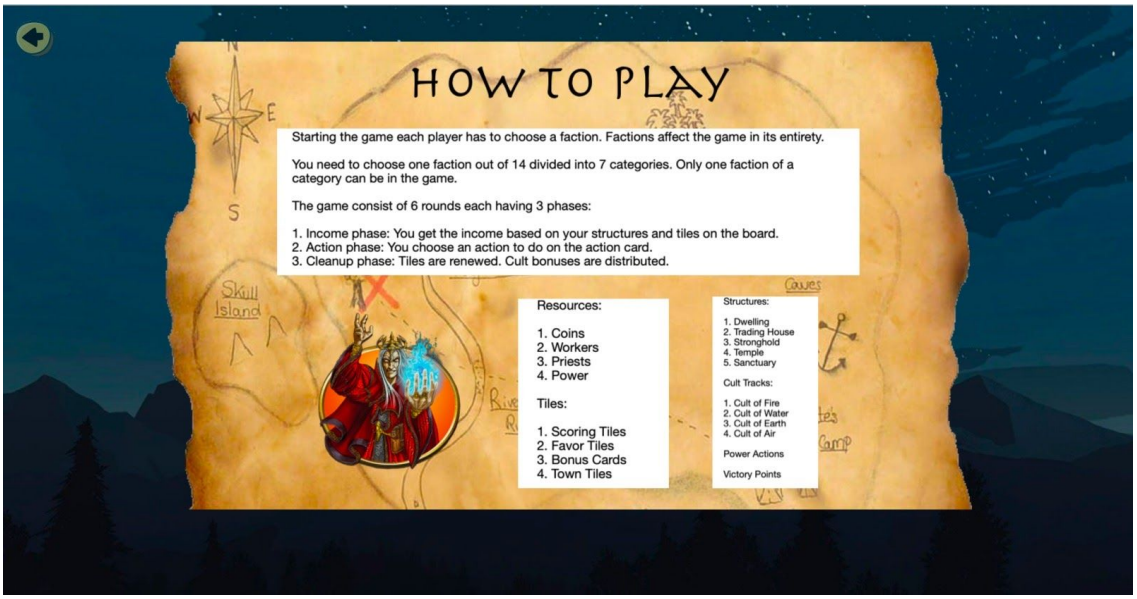

Figure 2: Help Screen

4.3. Login Screen

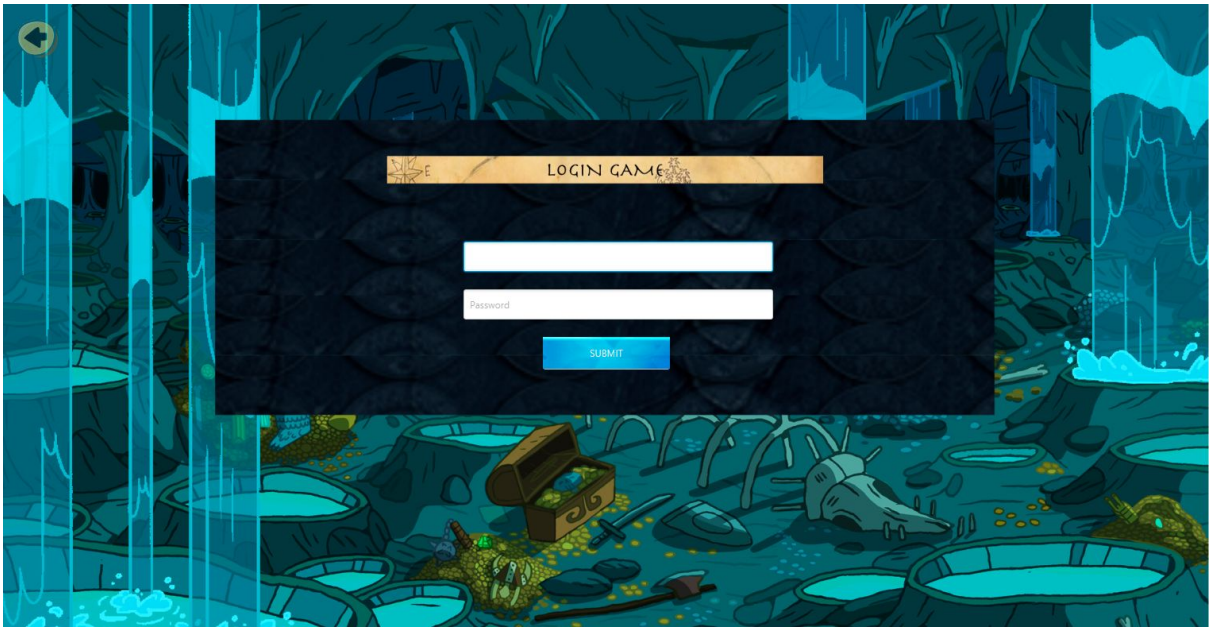

Figure 3: Login Screen

- There are 2 fields for username and password.
- Existing username and password combination leads to the Game Setup Screen.
- Nonexisting username and password combination results in failed Login Screen.
- Return button leads back to Initial Screen

4.3.1. Failed Login Screen

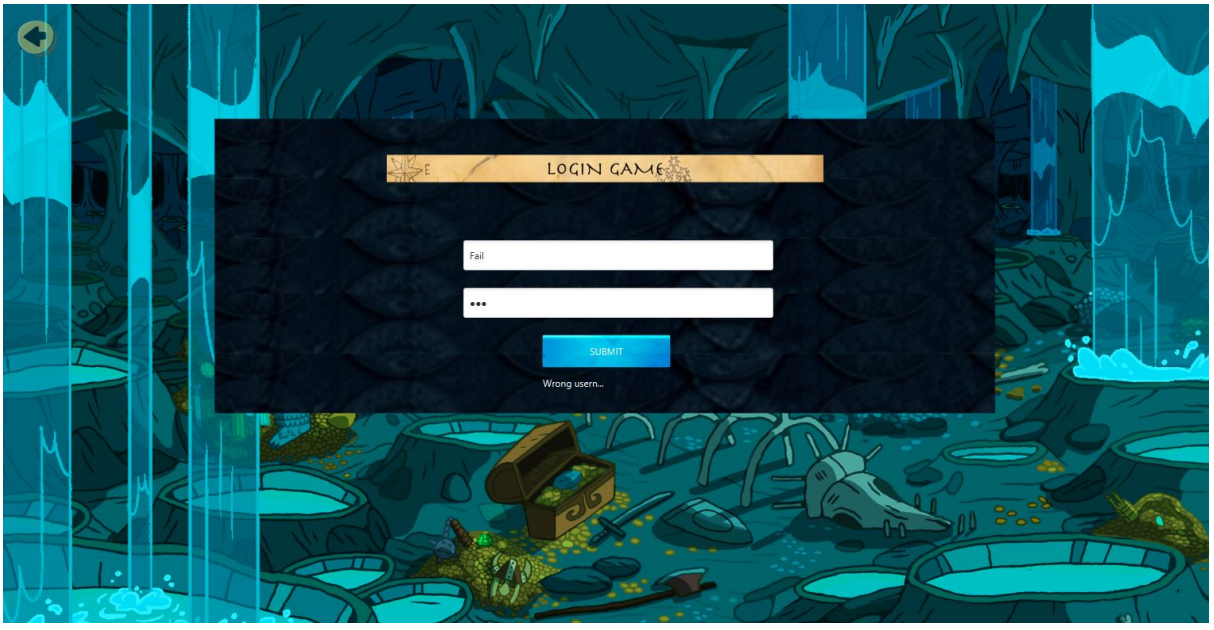

Figure 4: Failed Login Screen

- There are 2 fields for username and password.

- Existing username and password combination leads to the Game Setup Screen.
- Nonexisting username and password combination results in failed Login Screen.
- Return button leads back to Initial Screen

4.4. Game Setup Screen

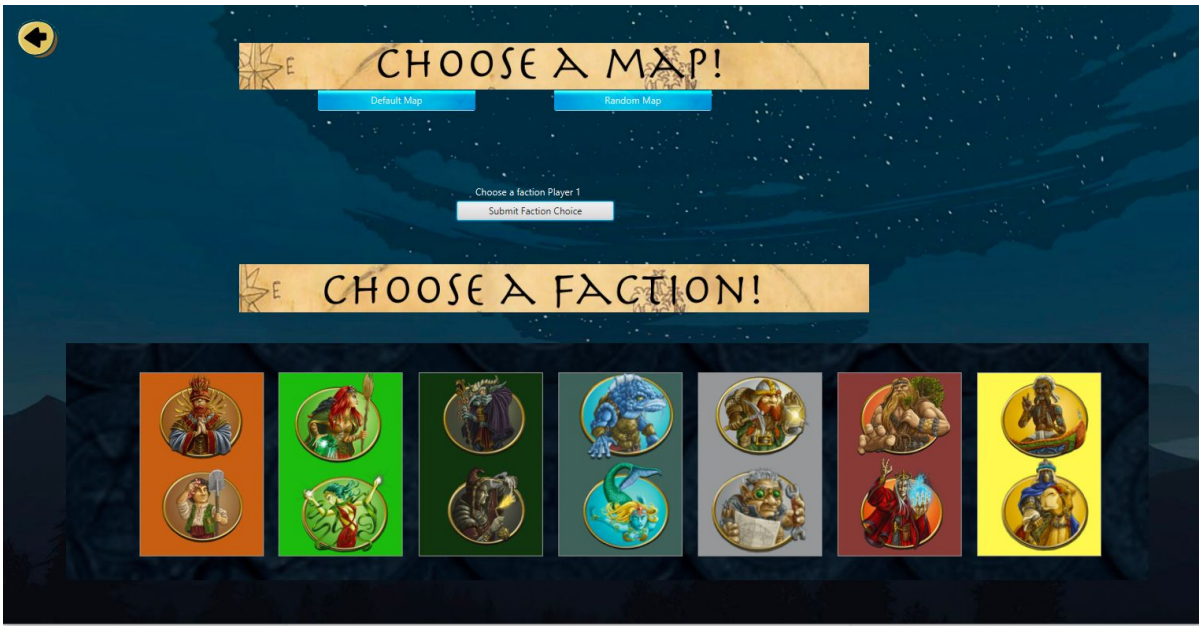

Figure 5: Game Setup Screen

- A player chooses their faction by the images below.
- A faction choice is submitted by Submit Faction Choice button.
- There are two buttons for choosing random or default map. Each button leads to the Game Screen.

4.5. Game Screen

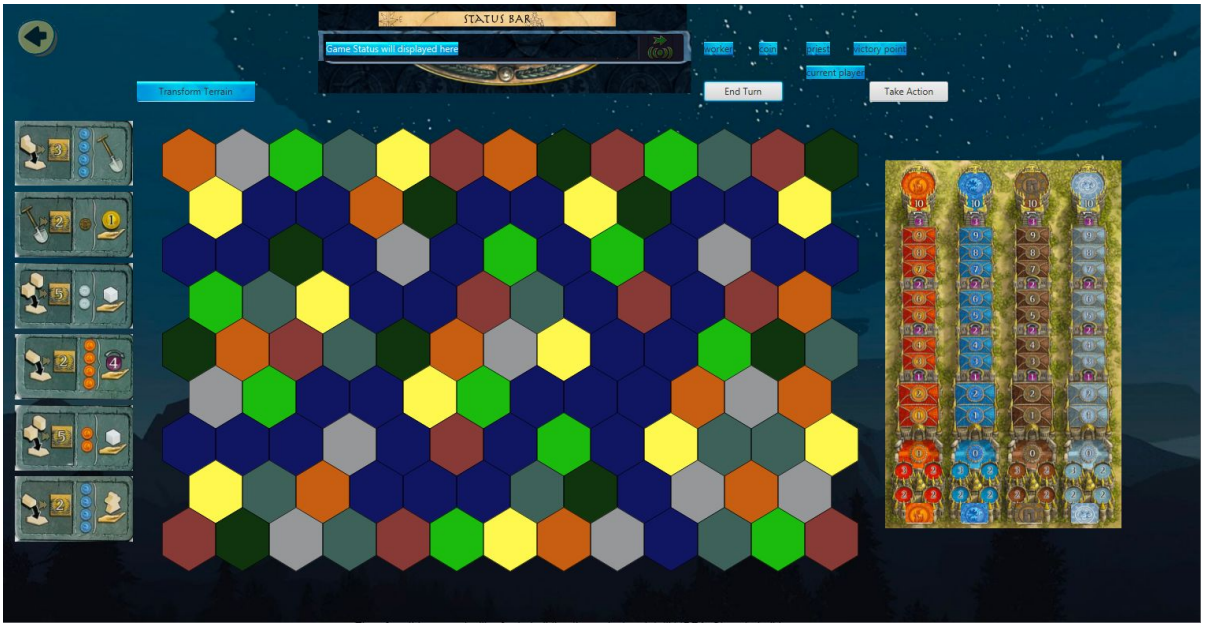

Figure 6: Game Screen

- This screen holds the map in the middle.
- On the left there are scoring tiles.
- On the right there are the cult tracks.
    - Buttom of the cult tracks, there are buttons as symbols of each track. These symbols are buttons for Send Priest to Order of a Cult action.
- On the top there is a status bar indicating whether an action is successful or not. If unsuccessful, the reason is also presented.
- Take Action button leads to Action Card pop up.
- On the top right there are resource informations of the current player. Values on the figure are placeholders indicating the information kept.
- End Turn button terminates the current player's turn and updates the resource information.

4.5.1. Action Card Popup

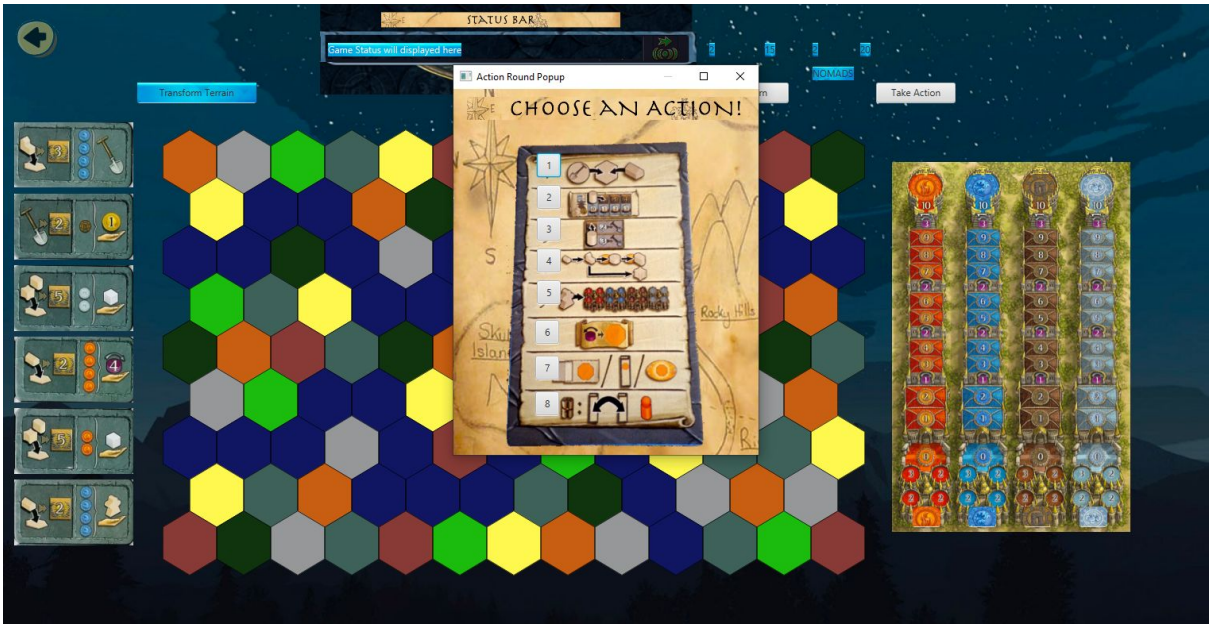

Figure 7: Action Card Popup

- There are 8 buttons for taking the 8 actions of the game.
- The 1st button leads to Transform Terrain Popup
- The 2nd and 3rd buttons has no UI interaction following itself.
- The 4th button leads is following up by clicking a terrain.
- The 5th button are to be followed by clicking the buttons below cult tracks.
- The 8th button is for Pass action and has no UI interaction following itself.

4.5.2. Transform Terrain Pop up

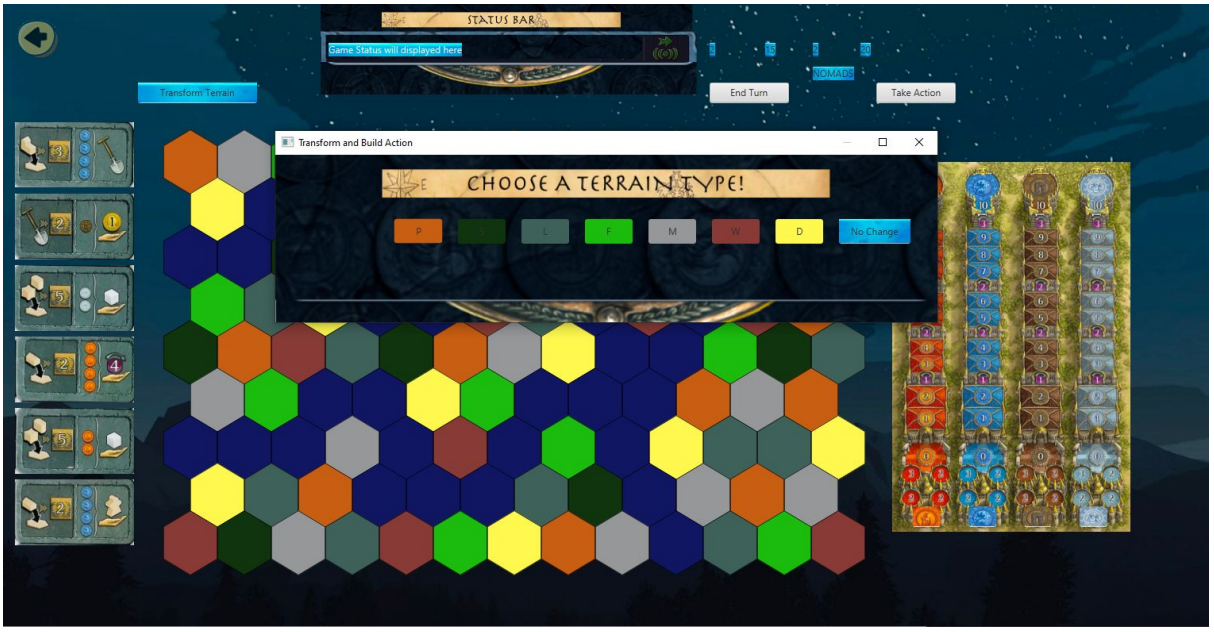

Figure 8: Transform Terrain Pop up

- There are 8 buttons to choose the desired terrain type. All of which results in Build a Dwelling Pop up

4.5.3. Build a Dwelling Pop up

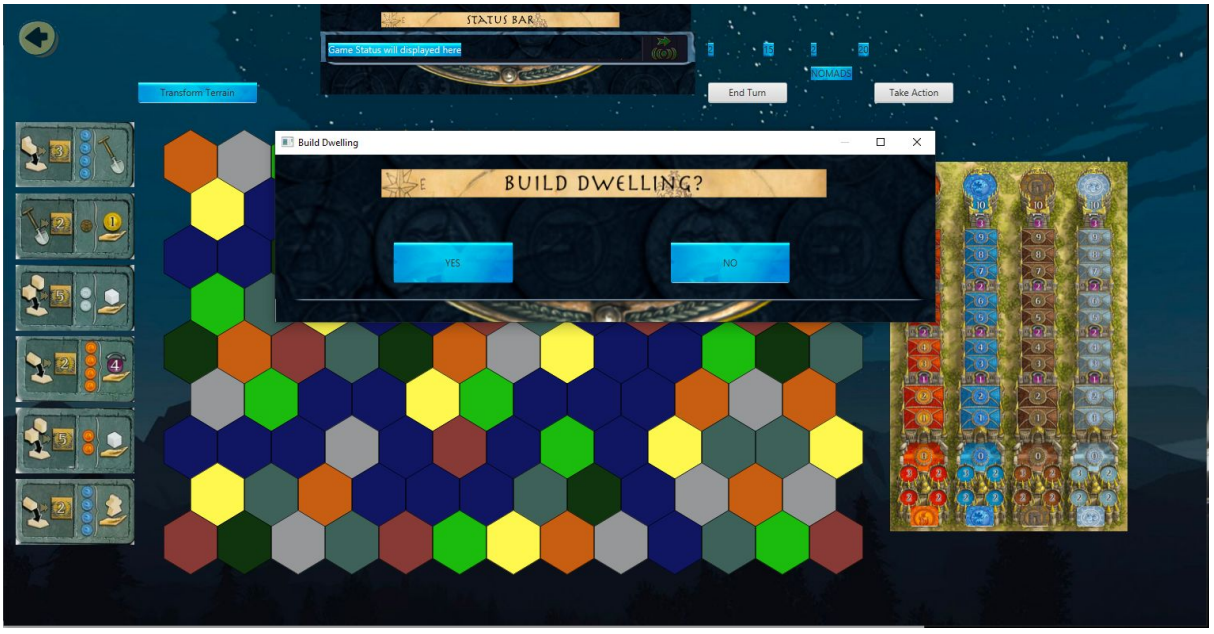

Figure 9: Build a Dwelling Pop up

- There are two options for building a dwelling. Each option is to be followed up with clicking on a terrain.

4.6. Exemplary In Game Screen

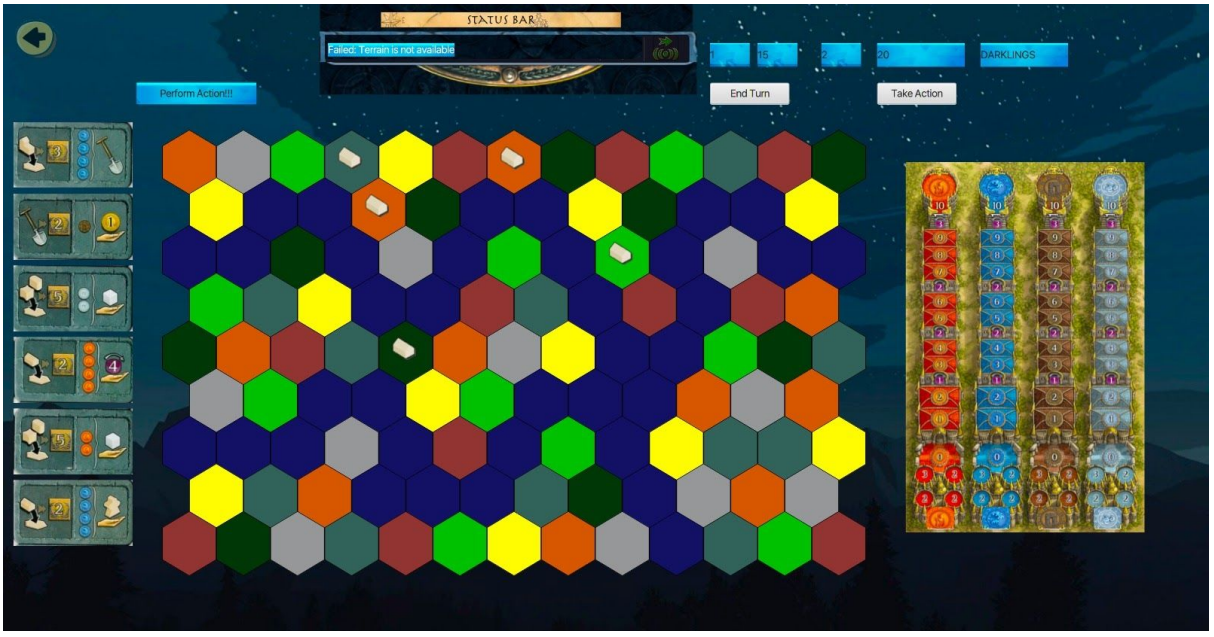

Figure 10: Exemplary In Game Screen

5. Build Instructions

Both for Windows and MacOS operating systems procedure is the same.Since the project is developed in Java, it can run as a desktop application on MacOS and Windows platforms. The project was developed in IntelliJ IDEA (version 2019.1.3). Therefore this manual will refer to building the project on IntelliJ IDEA. Steps to build the project are as follows:

- Download the project folder from the Github.

- Open IntelliJ IDEA.
- Create a new JavaFX project with Maven.
    - In IntelliJ, go “File -> New -> Project -> Maven”.
    - Enable “Create from archetype”.
    - Select “Add archetype”.
    - Set the groupId “org.openjfx” and set the artifactId “javafx-maven-archetypes”.
    - Once installed, select the artifact.
    - Select the archetype artifactId “javafx-archetype-fxml”
    - Create a property with name “javafx-version” and set it to “14”
- In IntelliJ go, “File -> Open” and open the downloaded project folder.

- Click on the maven logo on the very right of the screen.
    - Open Plugins and double click,
        - compiler->compiler:compile
        - javafx->javafx:compile
        - javafx->javafx:run
- Add new configuration.
    - Add new Application configuration.
    - Set JRE to 11.
    - Set main class to App.java.

Server side of the project was implemented by JSON structure and Rest API methods were used.Actual methods are written in C# language in Visual Studio Code. Therefore, in order to access the server you should import some dependencies.

- You should first install the file in the link before getting started.
- https://drive.google.com/open?id=10SZhp5LGK5W_PHbKKSzwoDbvnK1HUMHI
- Open the downloaded file and complete the installation by following steps.
- First click file in on the left top then click the Project Structure.
- After that you should add the libraries from the tiny plus sign and choose java.
- Choose the External Libraries file.
- Go to org.json.rar file and include org.json.jar and add it.
- Go to commons-io-2.6-bin and include commons-io-2.6-bin
- Go to httpcomponents-client-4.5.12 and after that choose lib file and include it.
- After all these steps you directly access the methods from application and can use this url “http://ymacit-001-site1.ctempurl.com/cs319/” to access the JSON files.
- Run the application with the added configuration.

6. Work Allocation

All the design choices are decided on unanimously and all the documentation is handled with all members' participation.

6.1. Rafi Çoktalaş, 21601537

- I manage the overall progress of the group. This has included the task assignments in reports and deliverable and report reviews.
- Together with Zeynep, I have created the skeletal project for the group to start working on. Moreover, we started to implement UI.
- I partake on the progress of the logic of the game.
- My main contribution to both UI and logic was the mathematical modeling of the gameboard.
- At last, I participated in the testing process.

6.2. Mahir Efe Macit, 21601510

- I was participated in the design of the class and object diagrams and diagram models especially sequence diagrams.
- I implemented the server connection of the application via Rest API.
- I implemented the imports for connection.
- I helped the class methods in the backend of the application,with the help of them we decided the methods and storaged JSON structures on the server side.
- I helped with the testing process.

6.3. Kamil Gök 21600879

- I designed the diagrams particularly object and class diagrams and I helped other diagrams.
- I revised the object and class diagrams after iterations.
- I mainly focused on the backend of the application and I participated in integration of these classes.
- I revised codes after testing processing and I mainly worked on how to fix the bux and maintain the overall efficiency.

6.4. Zeynep Cankara 21703381

- I was responsible for managing the scene transitions and User Interface. I worked on SceneManager Package which handles the Applications stages.
- Together with Rafi, connected to logic related packages to UI.
- Designed the UI components.
- In documents I prepared the “Use Case Diagram”, “Subsystem Decomposition Diagrams” and documentation of “Boundary Cases”.
- Additionally, I was keeping group Wiki while we used to meet in person before work from home compliance.

6.5. Arda Kaan Gültekin 21601137

- I modeled the activity and the state diagram models before and after the feedback.
- Also worked with my friends in object diagram models and also explanations of the classes in the model.
- Also worked in the back-end of the code(logic-design) in class methods.
- At last I participated in the testing process of the code both in UI/logic tests.