Bilkent University

Department of Computer Engineering

Object-Oriented Software Engineering Project

CS 319 1J-TM: Terra Mystica

Final Report

Group Members

Olcay Akman

Zeynep Korkunç

Münevver Uslukılıç

Can Mergenci

İrem Seven

Hüseyin Ata Atasoy

Instructor: Eray Tüzün

Teaching Assistant: Alperen Çetin

Progress Report
May 18, 2020

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the implementation of the Object-Oriented Software Engineering course CS 319.

Introduction

During the design and the implementation process, our team divided into two sub-groups which are front-end and back-end teams. Olcay, İrem and Münevver focused on designing and implementing the user interface subsystem of our game while Ata, Zeynep and Can focused on Game-Logic Subsystem design and its implementation process. Due to the unfortunate situation of pandemic, our group passed through some difficulties that caused some incomplete parts in our implementation. Some of the features that our group have committed to add into the game cannot be finished before the deadline of iteration 2. However, by conducting remote meetings and distant communication a partially-working Terra Mystica game is designed and implemented. The game is multiplayer which can be played on the same screen.

Design Changes

After Iteration 1, our group 1J-TM has focused on improving the design issues according to the feedback obtained from the TA and our instructor. Afterwards, a better design is put into practise in order to produce an easier stage of implementation. Latest version of the design is updated in Design Report Iteration-2. The design is maintained during the implementation process except some minor changes. Furthermore, for the Game Logic Subsystem, Singleton design pattern is added in order to reach our one and only game instance from the needed areas and it resolves many of the design deficiencies.

Lessons Learnt

Distant Communication

An unexpected duration occured in the 2020 Spring semester and the Coronavirus spread all around the globe causing a world-wide alarm situation. Therefore, our institution and course has been affected from the unexpected occurrence of the pandemic and the later processes. The ambiguity of the situation causes hesitations about working on the project. In this duration, our team has searched for new ways of conducting meetings, working on the design and implementation of the project collaboratively. We have increased our use of online teleconferencing systems such as Zoom and Discord. Our online weekly meetings were held in either one of those before mentioned teleconferencing systems. Apart from the communication at the time of the meetings, the group was in touch via an instant messaging application (WhatsApp) and a business communication platform that allows for communication between peers about work related issues (Slack), from where any group

member could communicate with another instantly to address any urgent matter that has come up during the semester. This was especially helpful after the Coronavirus outbreak as all members have fallen apart. Although the situation was unexpected, it taught the team to be prepared for real-life crises.

Importance of Design

Spending a considerable amount of time during the design process has showed its positive impacts during the implementation state. Better subsystem designs prevented major time losses that may have occurred. Since uniting the subsystems and testing process are also requires a significant amount of effort for each individual, improving the major design mistakes should not be done at the implementation process.

Regular Meetings

The regular meetings of the teams have increased its significance during the pandemic. Therefore, almost every week the team has come together except the subgroup meetings in order to discuss the project and assign work to members for upcoming deadlines that the team decided based on the deadlines assigned for the project. Our meeting agenda, notes, action items that consist of assigned works to group members and the topics for the next week is stored for each meeting and accessible by every member. By keeping records of the meetings, the tasks for each individual are clear and visible. Thus, it prevented the possible collisions of the group members which may occur by working on the same tasks. Also it prevented some tasks from being incomplete.

Time Management

The issue of time management regarding the project workflow during the semester has been highly dependent on the nearest assignment due. This naturally created a priority list of the tasks ahead of us. Because the regular meetings were held, the due assignments were handled nicely until the Coronavirus outbreak. Unfortunately, when all students were suddenly sent home due to the pandemic, the group lost synchronization tremendously. The members had a hard time adjusting to the new system and order, thus the weekly meetings and the management of our time for the group project has been negatively impacted by this situation. Toward the end of the semester, some tasks were sadly tackled later than intended. It can be said that after the Coronavirus outbreak, time management has been damaged. Thus, this extraordinary time has shown us the importance of being fast adopters to new situations, and it taught us that failing to do so may negatively affect our work, as we experienced this semester.

Implementation

Allocating a significant amount of time has shown its importance during the implementation process. Our team passed through some difficulties during the merging of the subsystems. Even though the subsystems have put their time and effort during the implementation, the merging process has required more time then estimated. Our team has learned that implementation requires much more time and effort than expected. Therefore, an earlier start for the implementation process is required in the projects that need collaboration.

User's Guide

Figure 1: Main menu of the game

New Game: Creates a new game

Quit Game: Exits the game

Figure 2: New Game

User enters the number of players and confirms
Users can return the main menu by using the back button.

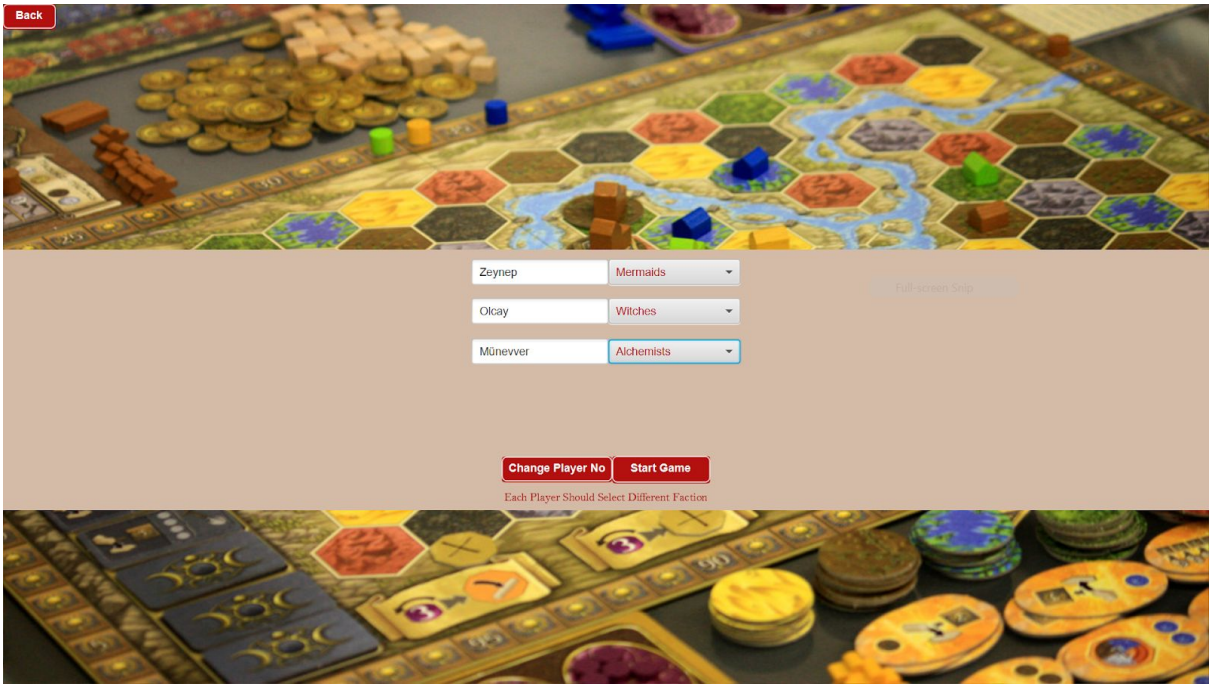

Figure 3: Creating a game by given number of players

Users can choose their factions and start game
User can return back to select of number of players

Figure 4: Game Play

User is informed to place dwelling on set-up phase.

Figure 5: Play action

User chooses an action to play. Here it is shown the case when the user clicks special actions.

Figure 6: Power Actions

Users is shown the available power actions they can perform. The ones they cannot current perform are shown with an X on top.

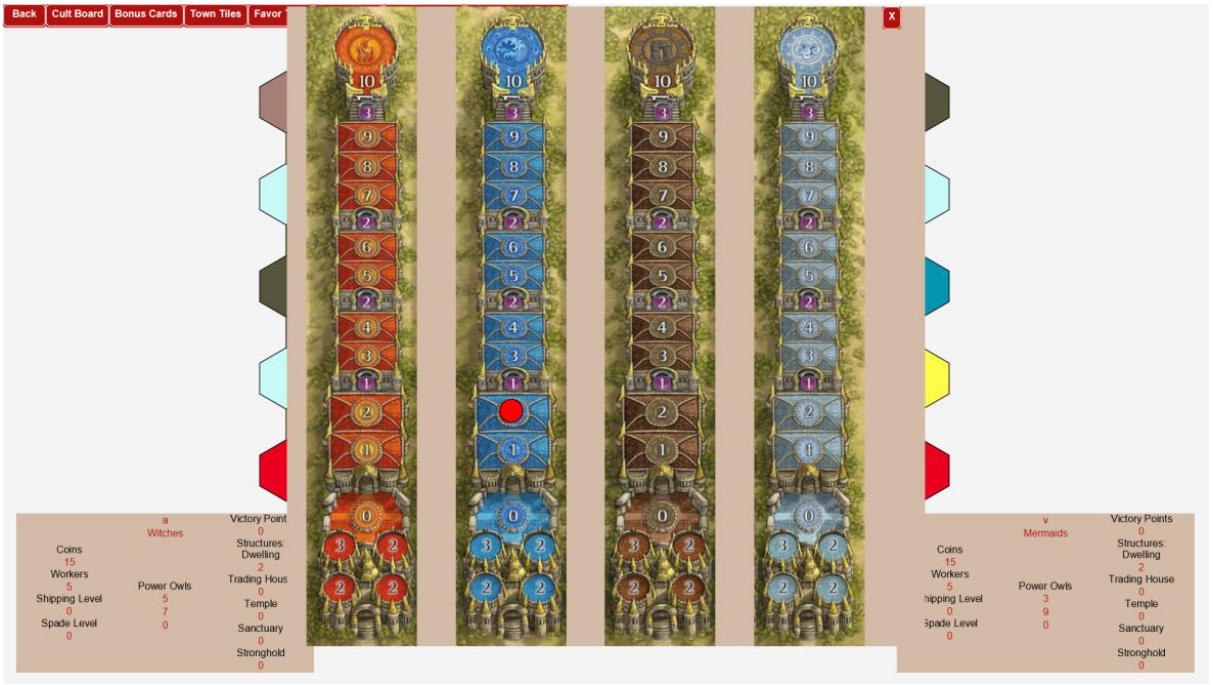

Figure 7: Cult Board

Users can track their position on the cult board and move their priests on cults as an action.

Figure 8: Town Tiles

Users can choose a Town Tile they like when founding a town.

Figure 9: Bonus Cards

Users can select the bonus cards given when they pass. Also they can select the cards before the first rouns.

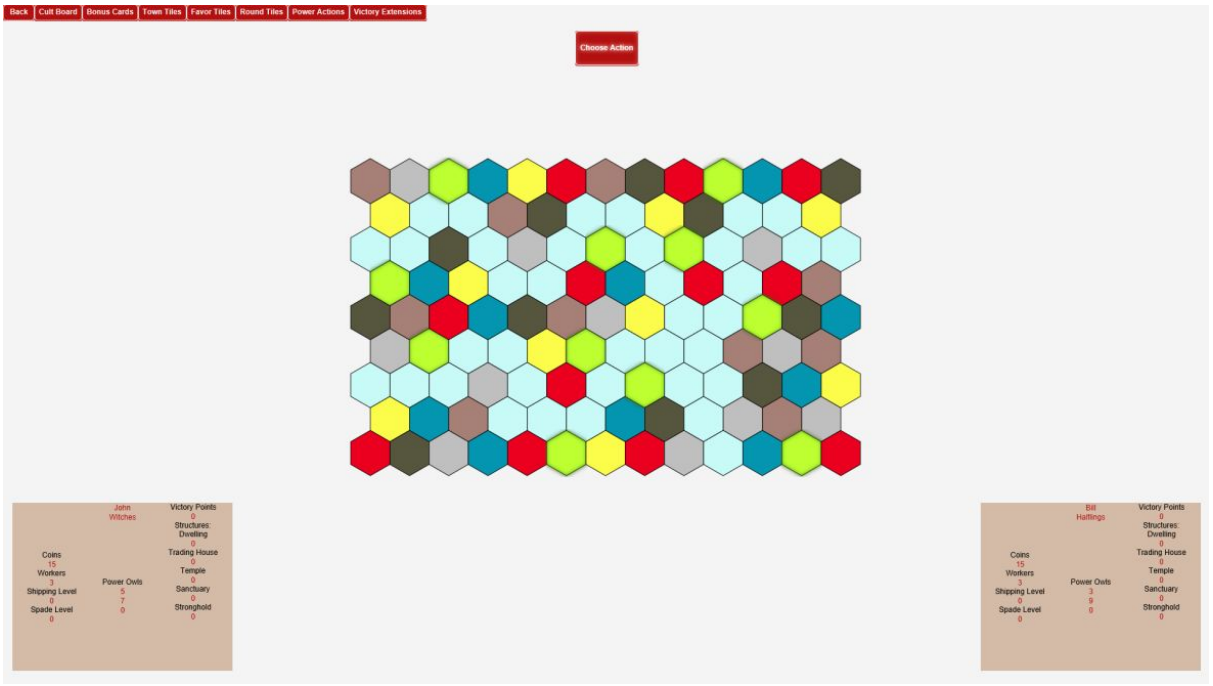

Figure 10: TerraLand

The default screen when the users enter the game, before building any dwellings.

Figure 11: Round Tiles

Pop up screen shows the scoring tiles for each round.

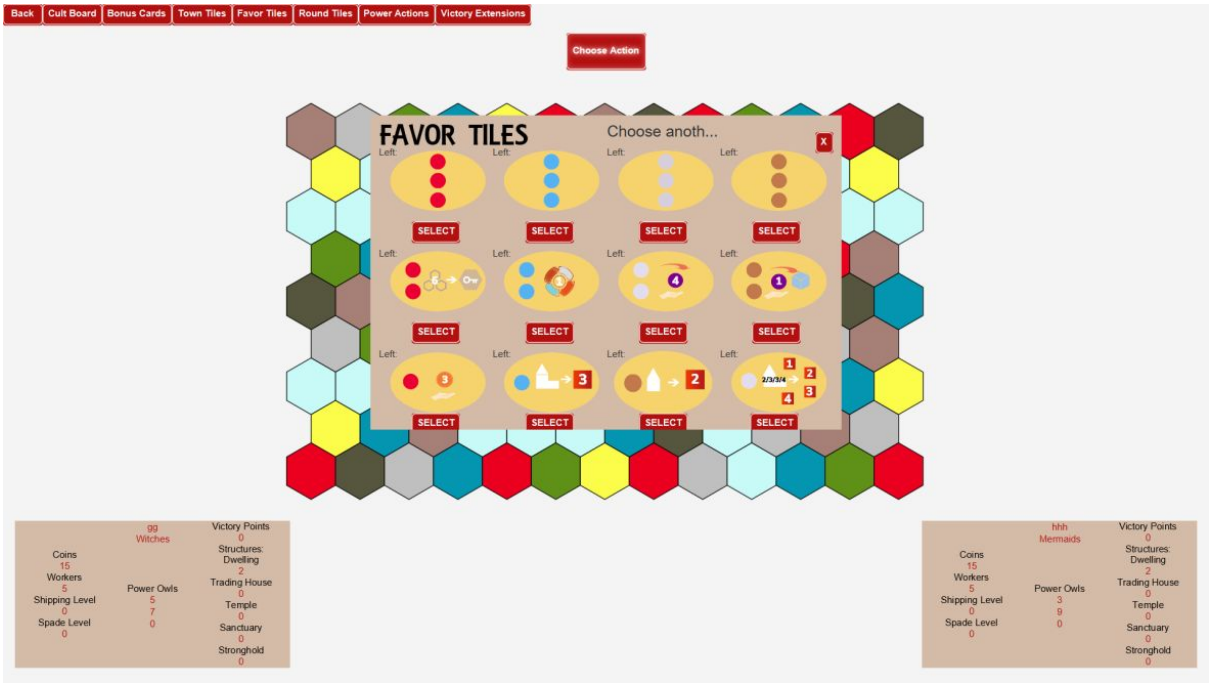

Figure 12: Favor Tiles

Pop up screen shows the favor tiles and the user can select a tile that he/she requires.

Figure 13: Income Phase

Pop up screen that shows the current phase.

Build Instructions

The program for the Terra Mystica Desktop game developed by group 1J-TM is programmed using Java and it runs on JVM, thus the game can be played on different platforms. Although the game runs smoothly on Windows, minor issues are existent when the game is run on Linux or MacOS platforms. (The text displays are corrupt in some scenes on MacOS, and scene display sizes are corrupt in some views on Linux). Therefore, the game is said to be available only on Windows at the moment.

The implementation duration of this game was completed using the IntelliJ IDEA as the integrated development environment, and the project runs on IntelliJ IDEA version 2019.2.4 or newer. The project was built via the Gradle build tool, and the project can be built with version 6.2.2 or newer. Additionally the graphical user interface was developed using JavaFX version 11. The project uses JDK 13, and since JavaFX is not included in the Java Development Kit as of JDK 11, JavaFX has to be installed separately in order to play the game. Detailed information about how to install and set up the requirements mentioned so far can be accessed via their official websites, they will be linked in the References part.

Assuming you have the required installations specified above, the steps required to build and run the project of the game are as follows:

1. Download the project folder from the GitHub repository named 1J-TM.
2. Open IntelliJ IDEA (version 2019.2.4 or newer) and select the ‘Import Project’ choice from the start up screen. Then, select the folder named tmDesktop as the project to import, which is placed inside the 1J-TM folder. Then choose ‘Import project from external model’, and select Gradle.
3. IntelliJ IDEA should ask to complete the configurations automatically. Allow this. Also make sure that the Project SDK is set to JDK 13 from Project Structures.
4. Once the Gradle setup is complete, you will have to add a configuration to build and run the project. To do this, click the Gradle symbol on the vertical column to the rightmost side of the IntelliJ IDEA window. There, click on tmDesktop and open the Tasks dropdown menu. Open application dropdown menu and right click on run, then select ‘create tmDesktop [run]’.
5. Now the game can be built & run by clicking the green play button on the top right corner of the window.

Work Allocation

Olcay Akman

Analysis Report

- Introduction
- State Machine Diagram & Description
- Class Diagram in Problem Domain & Class descriptions

Design Report

- Object Design Trade-Offs in Low-Level Design
- User Interface Subsystem in Low-Level Design
    - User Interface Subsystem Class Diagram & Class descriptions

Implementation

- Created the graphical user interface of the game collaboratively with İrem and Münevver.
- Worked with Ata, Zeynep, Münevver and İrem to unite the game logic and user interface codes.

Final Report

- Lessons Learnt
- Build Instructions

Zeynep Korkunç

Analysis Report

- Game Elements in the Overview section
- State Machine Diagram & Descriptions
- Class Diagram in Problem Domain & Class descriptions

Design Report

- Game Logic Subsystem Design & Improvements for the second iteration
- Detailed class descriptions of the Game Logic Subsystem
- Explained the packages developed by our team

Implementation

- Worked with Ata on the Game Logic Subsystem Design. Implemented Town Tile, Favor Tile, Cult Board and Faction subclasses.
- Worked with the UI Team(Olcay, İrem, Münevver) in order to unite the codes.

Final Report

- Introduction
- Design Changes
- Lessons Learned
- Users Guide

Münevver Uslukılıç

Analysis Report

- Class Diagram Design & Descriptions
- Sequence Diagram & Descriptions
- Functional & Non-functional Requirements

Design Report

- User Interface Class Design & Descriptions in Low-Level Design
- External Packages developed of the JavaFX project of User Interface

Implementation

- Collaborated with İrem and Olcay to implement scenes and layouts User Interface of the game.
- Worked with Zeynep, Ata, Olcay and İrem to implement the Controllers of the scenes.

Can Mergenci

Analysis Report

- Use Case Diagram & Use Case Descriptions
- Activity Diagram and its Highlighted Versions
- Interactive Menu Structure Mock-ups

- Some minor contributions in the brainstorming of generating application domain classes.
- Preparing a document format template (fonts, heading styles, etc.).

Design Report

- Purpose of the System, System Design Goals Criteria & Trade-offs
- Subsystem Decomposition, Hardware/Software Mapping, Persistent Data Management, Access Control & Security, Global Control Flow, Boundary Conditions
- Package Diagram, Deployment Diagram, Component Diagram
- Subsystem Services
- Preparing a document format template (fonts, heading styles, etc.).

Implementation

- Implementing a GameLogic model and a command line GameController that lets users play the game from beginning to end. (~900 SLOC)
  Challenging features include:
  - Connected components graph algorithm to calculate Area Scoring.
  - Polymorphic terrain adjacency algorithm for given shipping value.
- Testing the GameLogic classes with automated JUnit tests with approximately 80% code coverage. (~400 SLOC)

Final Report

- None

İrem Seven

Analysis Report

- Describing flow of the game
- Class Diagram Design & Descriptions
- Sequence Diagram & Descriptions

Design Report

- User Interface Subsystem in Low-Level Design
- User Interface Subsystem Class Diagram
- User Interface Class descriptions

Implementation

- Created the graphical user interface of the game collaboratively with Olcay and Münevver.
- Worked with Ata, Zeynep, Münevver and Olcay to unite the game logic and user interface codes.

Final Report

- None

Hüseyin Ata Atasoy

Analysis Report

- Updating the Class model for the Application Domain in the 2nd iteration.
- Contributed to the Application Domain analysis, such as detecting the required classes.

Design Report

- Designed the Game Logic Subsystem class diagram. Applied the Singleton Design Pattern in the 2nd iteration and made the necessary changes.
- Documented the Class Interfaces for the Game Logic Subsystem.

Implementation

- Worked with Zeynep on the Game Logic Subsystem Design. Implemented Bonus Card, Asset, GameHandler, Game and Player Classes.
- Worked with the UI Team(Olcay, İrem, Münevver) to implement the Controllers for the Game.

Final Report

- None.

References

JavaFX information: https://openjfx.io/openjfx-docs/
JDK 13 download: https://www.oracle.com/java/technologies/javase-jdk13-downloads.html
IntelliJ IDEA IDE download: https://www.jetbrains.com/idea/download/