CS 319 TERM PROJECT

Final Report

Section 3

Group 3B

Project Name: RISK 101

Group Members

Ramazan Melih DİKSU - 21802361

Aleyna SÜTBAŞ - 21803174

Yiğit ERKAL - 21601521

Baykam SAY - 21802030

Berk TAKIT - 21803147

Contents

Introduction

In the Risk101 project, we finished every part of the functional requirements. All features which are intended to be done are finished and working such as objectives (we changed the cards feature and we gave special objectives to every player), abilities (in Risk101, every player has special ability according to their faculties) and the Bilkent University themed map (territory names and places are adjusted according to Bilkent University) and gameplay (character selection, in other words, faculty selection has gained importance because of the abilities of the faculties). However, we changed some of the faculties' abilities from the ones that were specified in the previous reports. This decision was made to make some abilities more balanced and some abilities easier to implement. Our game does not have any major known bugs. There are some minor graphical bugs, but they do not harm the gameplay.

When we examine our goals at the beginning of the semester, we have accomplished building a modified Risk game. In our project, we tried to build a user-friendly interface so that everyone can play easily. Furthermore, our code is prepared with comments. These comments have been edited for the convenience of those who will review the code. Our project matches most of our non-functional requirements. All text is larger than 12pt. Territories are color-coded. The application does not crash. The clicks are acknowledged under 10ms and executed under 500ms. However, there are some nonfunctional requirements that are not met. The game currently runs only on Windows and saving the game is done at the start of each player's turn, not when the exit is clicked.

Lessons Learnt and Experiences

In the Risk101 project, initially, we tried to share our workload between our group members right after the first iteration of the design report. Then, we divided the Risk101 into sections that we will do. One of the group members worked on the user-interface, and the other group members worked on the game engine, logic of the game and the states of the game such as attacking, deployment and fortifying. First, every member started to implement their parts. When someone had some difficulties about his or her own part, we tried to help each other as much as we could. This solidarity has enabled us to do our duties both faster and more successfully.

The biggest gain of the Risk101 project was the learning importance of forward engineering for our group. Our class diagram helped each of the group member's implementation. If there was a change or need for an adjustment in our implementation, we directly updated the class diagram. Therefore, everyone got notified about these changes immediately. The importance and contribution of the UML diagrams were significant.

In our Risk101 project, we used JavaFX for our graphical user interface. As we used the JavaFX in our project, we learned the features of the JavaFX in more detail. In addition to JavaFX, GitHub was also another important tool for our group because we had to manage different versions of the same code.

After implementing them separately, we combined the UI and the game logic. Then, we started testing the features of our game. In the testing part, initially, we tried to use every feature of the game. If there was a problem, one part of our group tried to fix it and the other group kept searching for other bugs. After this investigation, finally, we tried to make the features of the game better in every possible way such as user-interface, sound, screen size. Combining different parts of the game in coordination with each other made us fully understand the importance of communication in engineering projects. If we did not communicate the way we did, we would not be able to fix our bugs and improve our game in such a short period of time.

User's Guide

This user's guide is intended for people who know how to play the original Risk board game. To learn how to play the original game, please refer to the rulebook provided below:

https://www.hasbro.com/common/instruct/risk.pdf

Differences in The Gameplay

1. Each player selects a faculty at the start of the game. This faculty represents the player. Each faculty has a special ability. Some faculties have passive abilities, while some have abilities that can be activated once in a game, while some can be activated once per turn. To activate an ability, the ability button is used.
2. Each player is given an objective card. These cards are a replacement for the territory cards in the original board game. We intended to minimize the luck factor by changing the territory cards to the objective cards. Players can view their objective by hovering on the objective button. When an objective is accomplished, they gain a specified number of armies to add the next turn. Objectives can fail when the specified condition is met, if this happens no additional armies are gained. When an objective is over, another one is generated the next turn.

Starting a New Game

A desired save slot is selected. Filled save slots are highlighted in red.

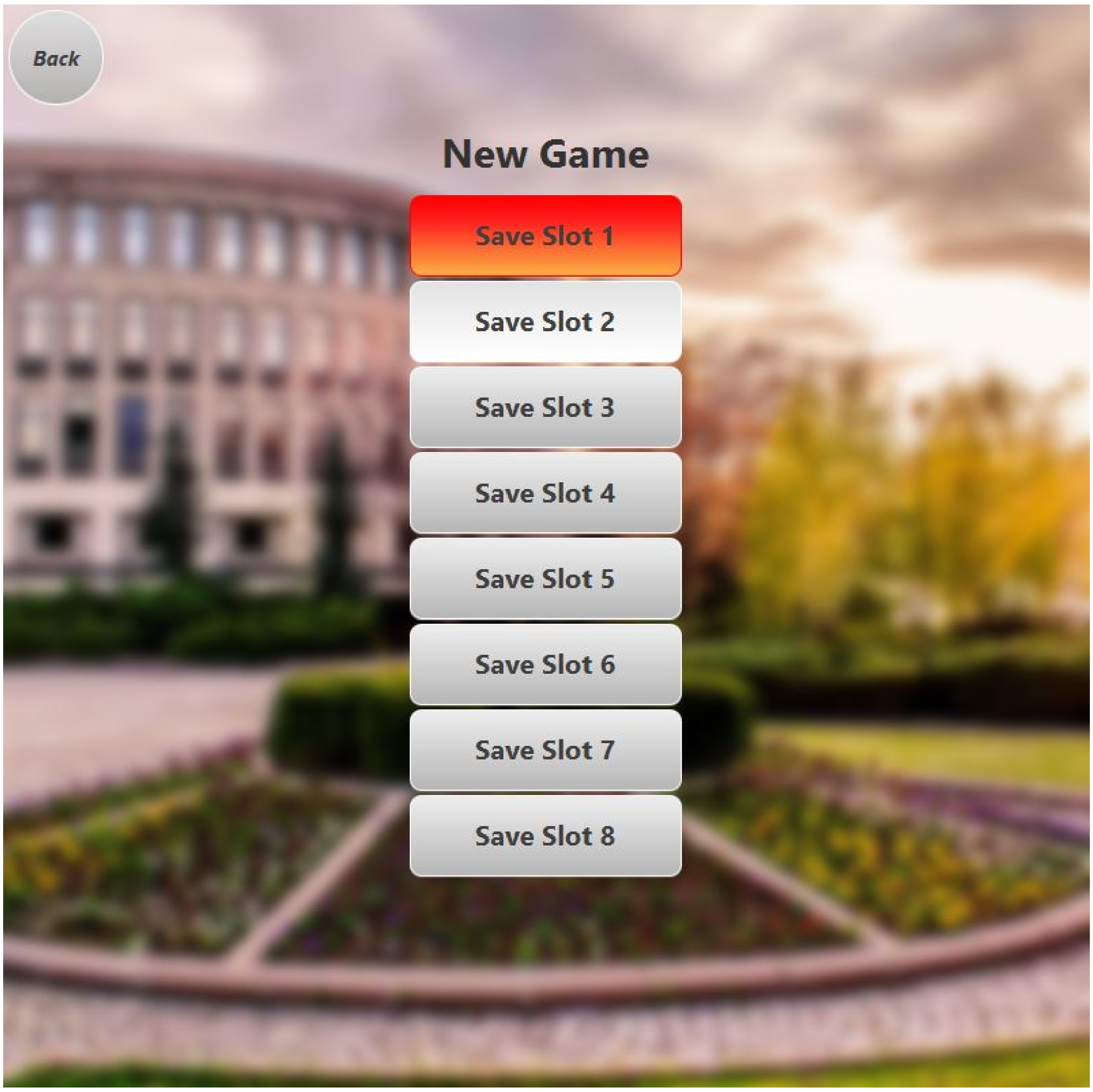
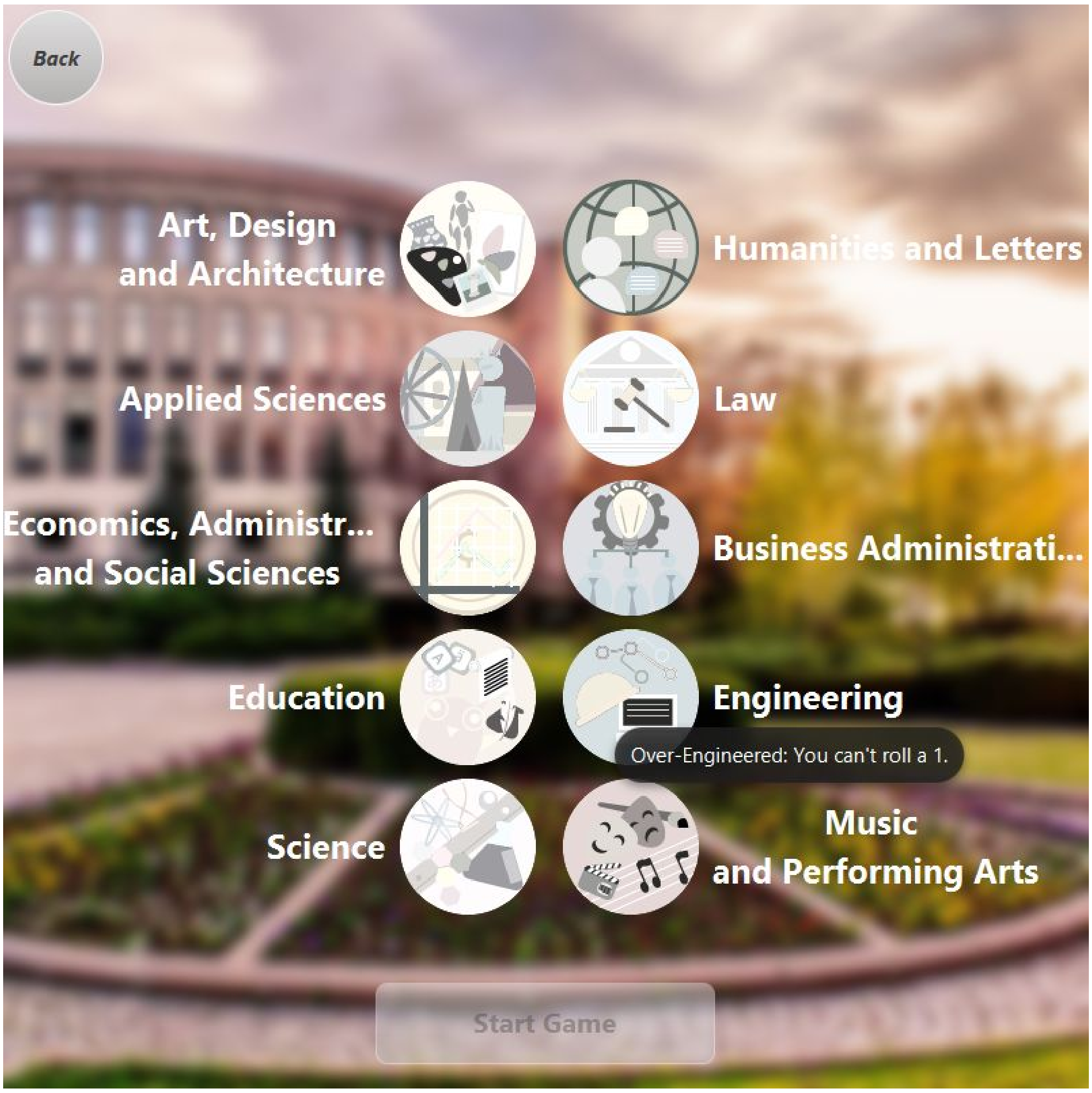


Figure 1: New Game Menu

When selecting a faculty, desired faculties are selected by clicking on the images. Hovering on the images shows the special ability of that faculty.



Over-Engineered: You can't roll a 1.

Figure 2: Faculty Selection Menu

Gameplay

At the start of the game, white areas represent the empty areas. Each player places one army by clicking on a desired empty territory, the game iterates between players one by one. The player turn sequence is shown on the right, along with each player's current number of owned territories. On the bottom left the current game stage along with the current player is shown. The remaining troops to be placed on the map is also shown.

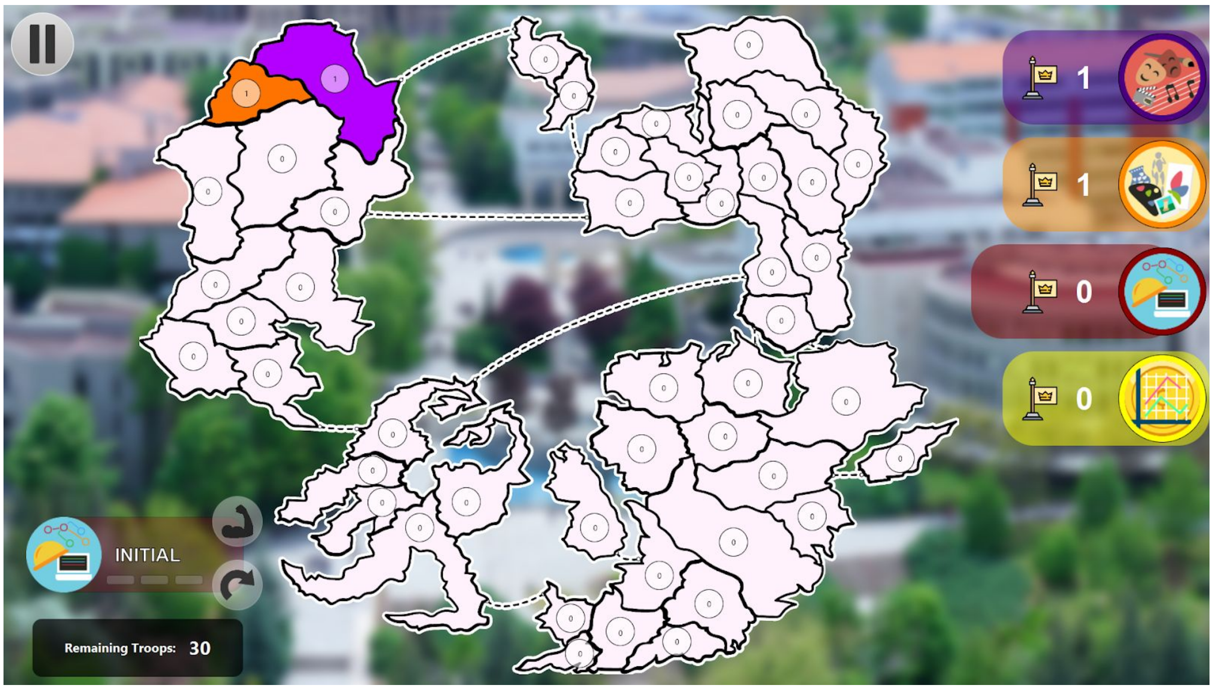


Figure 3: Initial Map Overview

Hovering on a territory shows the name of that territory.

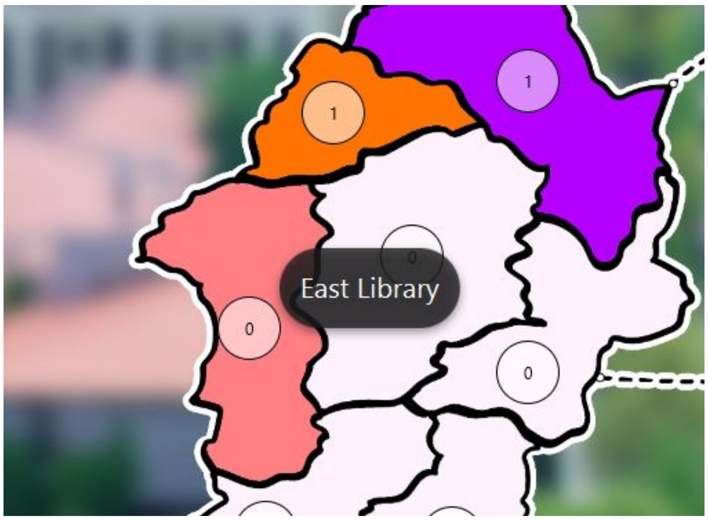


Figure 4: Territory Name Display

When all territories are selected, owned territories are reinforced one by one until no remaining troops are left. The game then proceeds to the army placement stage of the first player.

Army Placement

At the start of each player's turn, a certain number of troops are given to place on their own territories. When a territory is selected, a menu pops up to determine the number of armies to place.

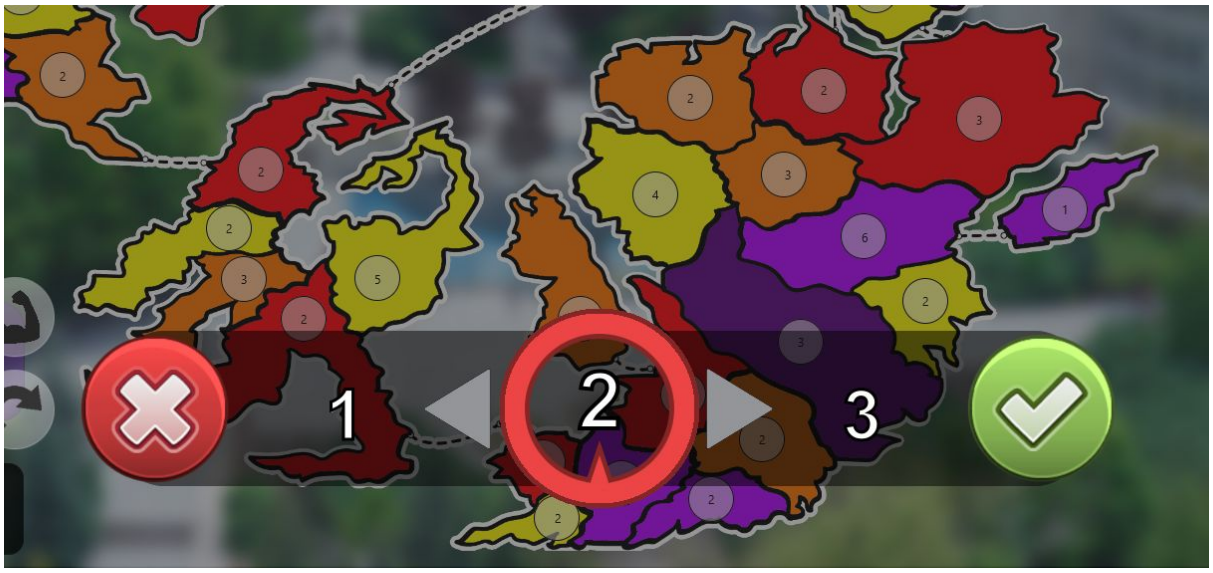


Figure 5: Army Number Selection Menu

Clicking on the left and right arrows changes the number of armies. This stage continues until no remaining troops are left.

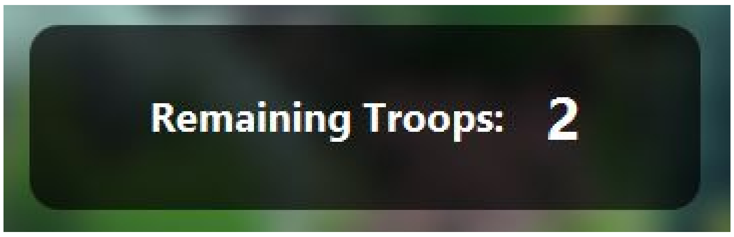

Remaining Troops: 2

Figure 6: Remaining Troops Display

Attacking

To attack a territory, first, the attacking territory is selected by clicking on an owned territory. Then, an adjacent enemy territory is selected.

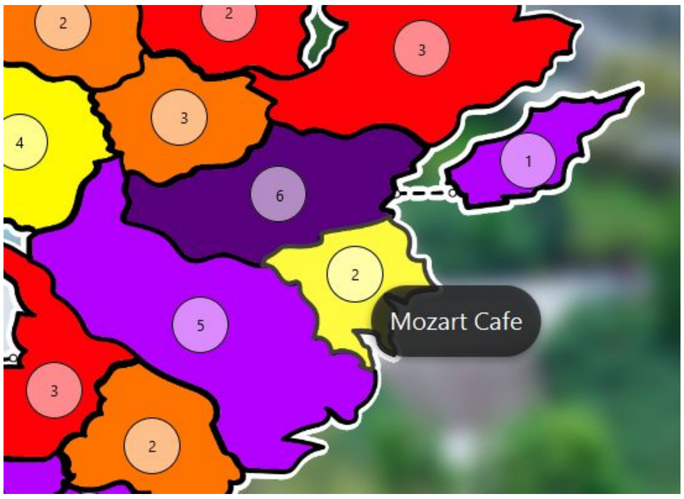


Figure 7: Attacking Scene

A dice selection menu appears. This menu determines the dice number of the attacker.

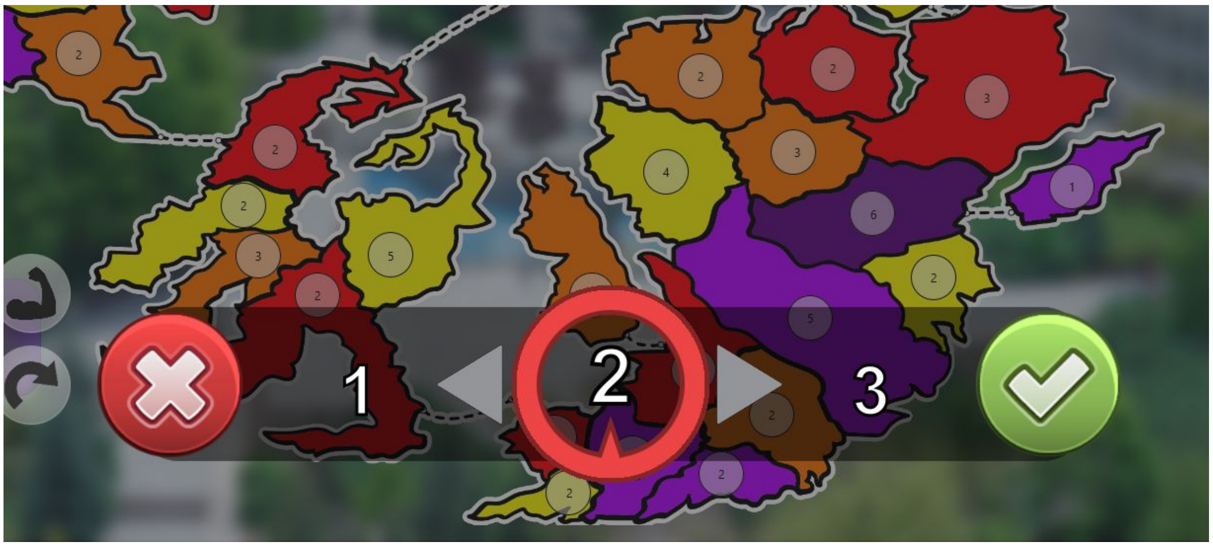


Figure 8: Dice Selection Menu

After confirming this menu, another menu appears that determines the dice number of the defender.

The war results are shown on the next panel. The winning dice are highlighted in red.

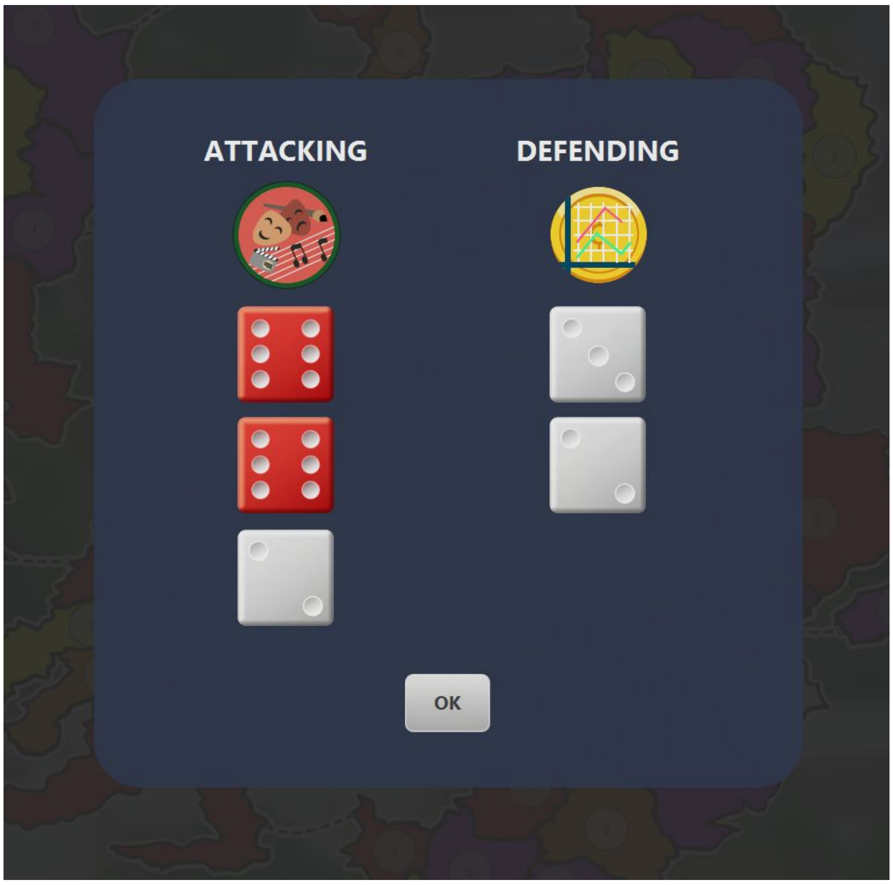


Figure 9: Dice Result Display

Clicking the "OK" button does different actions depending on the result of the war. If both territories can continue fighting, the dice selection menu is shown. If attacking territory has 1 army left, the game returns to the map and the attacking player can select a different territory. If the defending player has no armies left, an army selection menu appears. The attacker can select the number of armies to move to the newly gained territory in this menu.

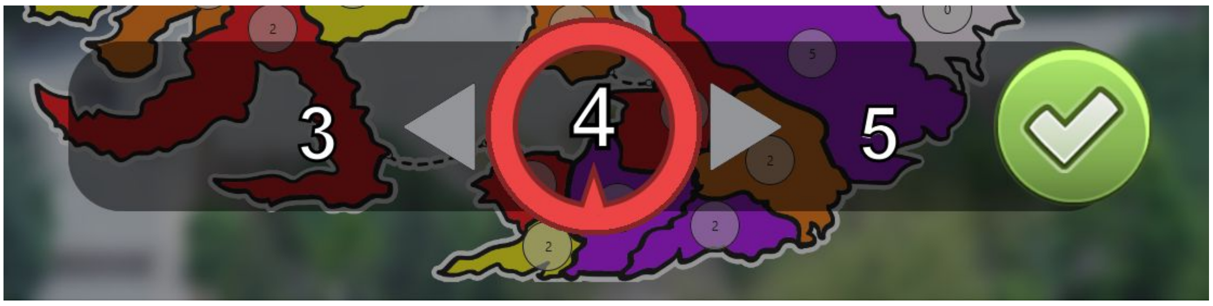


Figure 10: Moving Armies Selection Menu

When the attacker is satisfied, they can end their attacking stage by clicking on the pass button.

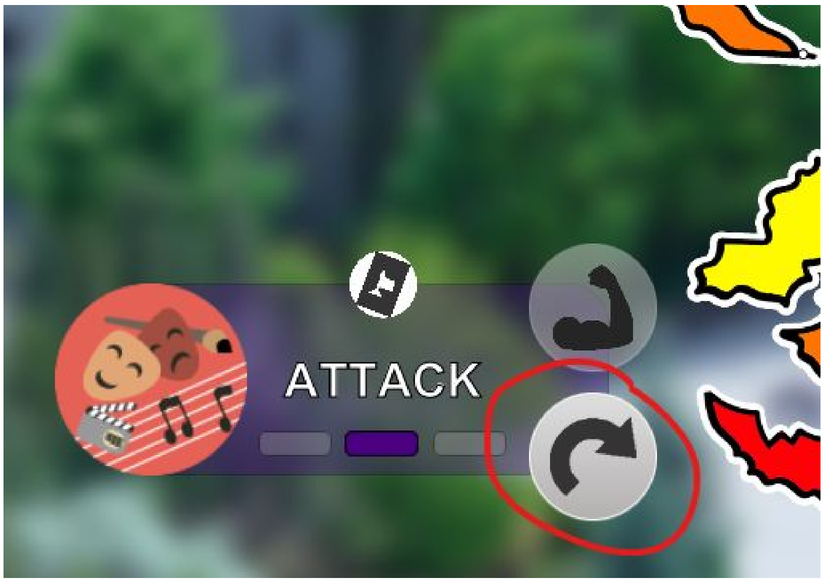


Figure 11: Pass Button

Fortifying

At the end of their turn, players can choose to move armies from one of their territories to another adjacent one. If they do not wish to do so, they can end their turn by clicking on the same pass button above. To fortify a territory, first, the source territory is selected. At this stage, clicking on the source territory again deselects it. Then the destination territory is selected and an army selection menu pops up. When the number of armies is confirmed the player's turn ends and the next player starts their turn.

Objectives

Hovering on the objective button shows the current objective of the player. There are two types of objectives: "Hold" objectives are given to one of the current places of the player and require the player to hold that territory for a certain number of turns, "Capture" objectives are given to an enemy place and require the player to capture that place before the turn limit is reached. The objectives can be given for both territories and areas (east campus, upper main campus, etc.). The reward that will be given after the objective is completed is specified at the bottom.

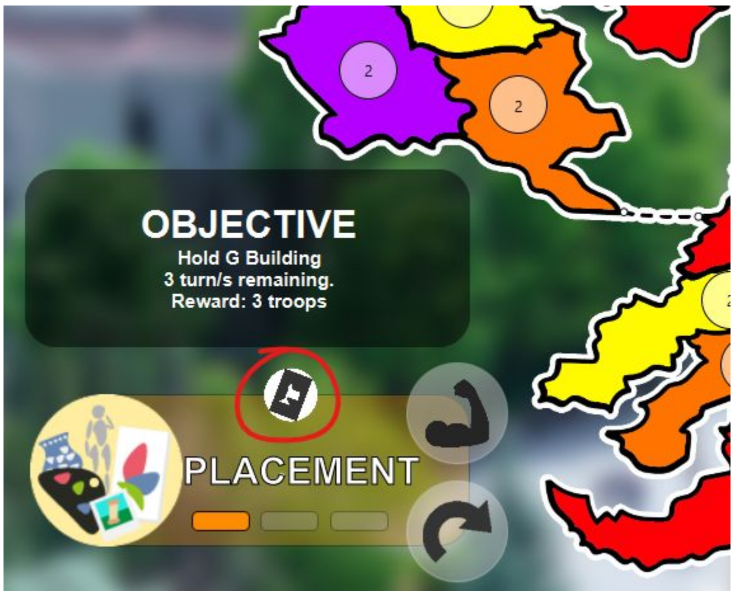
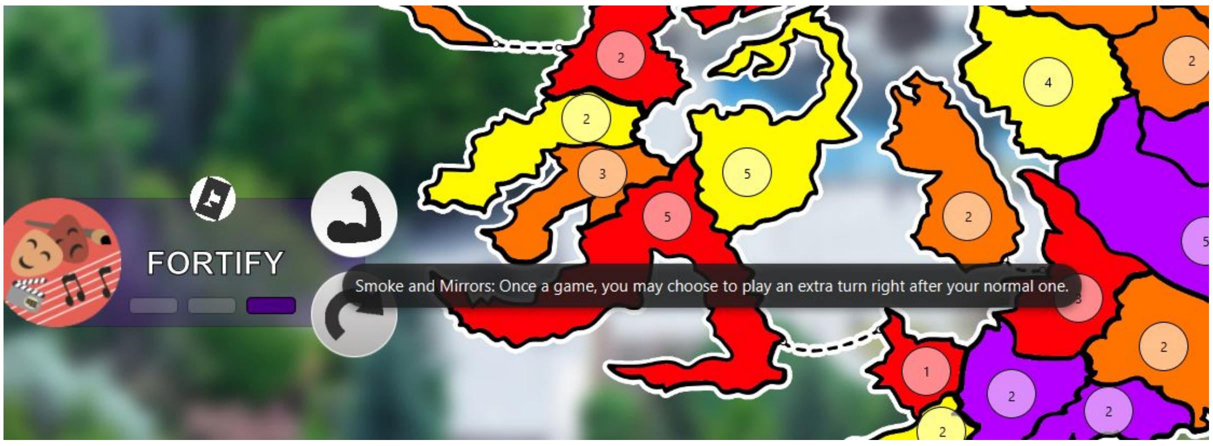


Figure 12: Objective Button

Abilities

Each faculty has their own special ability. The player's ability can be viewed by hovering on the ability button. If the ability can be used at that time, the button becomes clickable. Clicking on the button activates that ability. Some abilities can only be activated once per game, while some can be activated once per turn. Some abilities are passive and cannot be activated.



Smoke and Mirrors: Once a game, you may choose to play an extra turn right after your normal one.

Figure 13: Ability Button

Loading a Previous Game

The game is saved at the start of each player’s turn before they place their armies. To load the game, the load game menu is selected and an available save slot is selected.

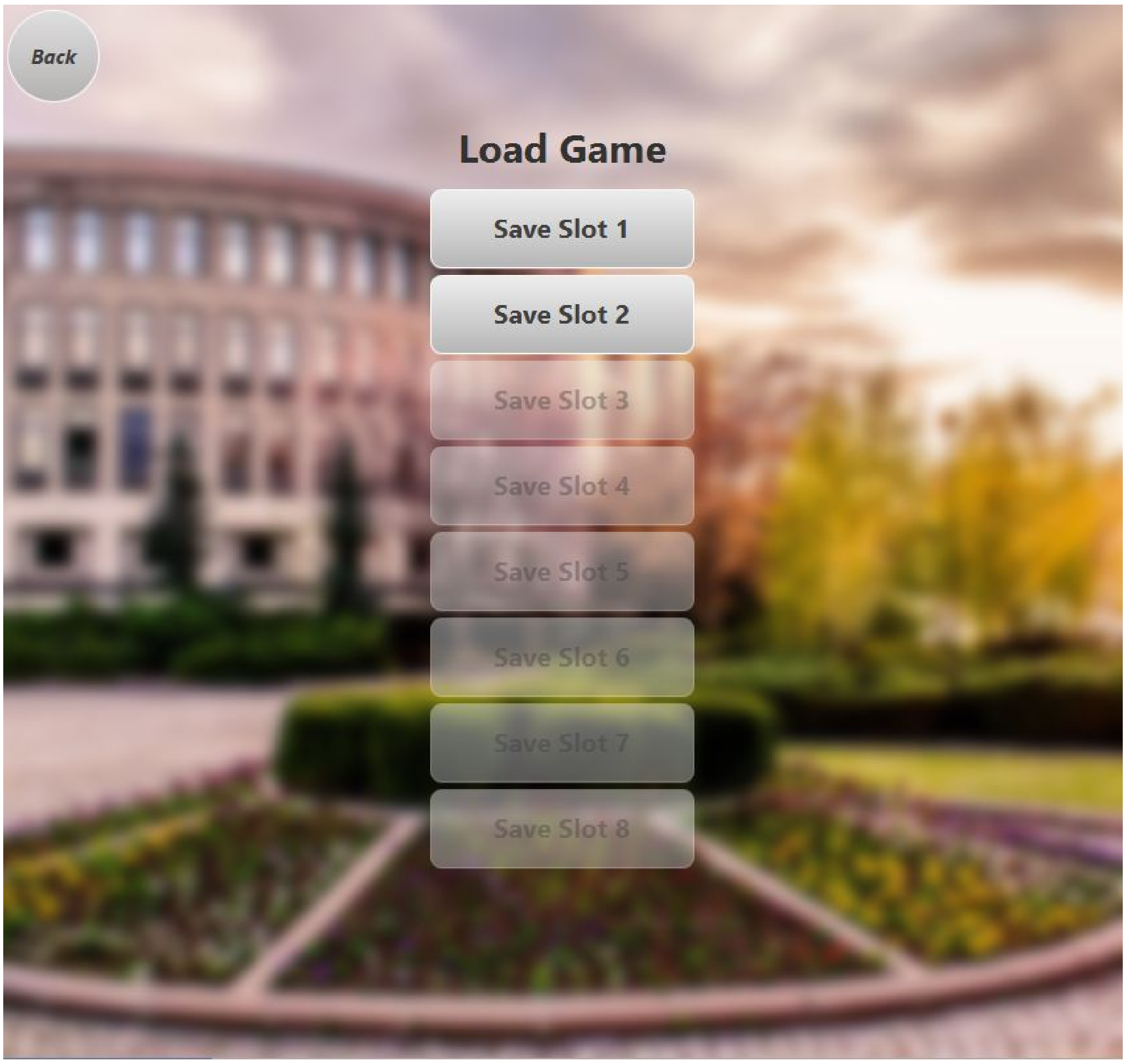


Figure 14: Load Game Menu

Build Instructions

Minimum System Requirements

- Dependencies: JDK 15 and JavaFX 15
- Supported OS: Windows
- Ram: 1 GB RAM
- Hard Disk: 500MB available space
- Processor: x86 compatible
- Peripherals: Mouse, Full HD monitor, optional headset

Build Guide

1. If no JDK 15 is available, download a JDK 15 distribution and unpack.
2. If no JavaFX 15 is available, download a JavaFX SDK and unpack.
3. Open the source code from an IDE as “import project”.
4. Set the “resources” folder as the resources and the “src” folder as the source folder.
5. In the libraries section of the project structure menu, add the JavaFX library. (specify the path as “\path\to\javafx-sdk-15\lib”)
6. Add the necessary VM options for running the project.
    a. --module-path path\to\javafx-sdk-15.0.1\lib --add-modules javafx.controls,javafx.fxml,javafx.graphics,javafx.media
7. Run the project.
8. To run the packaged project specify necessary arguments from the command line:
    a. java --module-path \path\to\javafx-sdk-15\lib --add-modules javafx.controls,javafx.fxml,javafx.graphics,javafx.media -jar .\3B-risk.jar
9. We used Intellij as our IDE, the following guide has visuals to support the steps given above: https://www.jetbrains.com/help/idea/javafx.html

Run The Game From Release

1. The game is released without any dependencies.
2. Download the latest release from Github.
3. Unzip to the desired location.
4. Run the Risk-101.exe.

5. Since the executable is not signed, Windows Defender can show a warning saying that the application is unrecognized. In this case, click “More info” and then “Run anyway”.

Work Allocation of Our Team

Ramazan Melih DİKSU

Analysis Report:

- Drew sequence diagrams for Attacking and Fortifying sequences.

Design Report:

- Wrote Subsystem Decomposition section.
- Drew an MVC Diagram.
- Co-wrote Low-Level Design section.

Implementation Process:

- Worked on game logic.
- Co-wrote the game states.
- Co-wrote the abilities.

Other:

- Designed a custom map for the game.
- Designed custom faculty icons for the game.

Aleyna SÜTBAŞ

Analysis Report:

- Wrote the map and rules sections.
- Drew the state and activity diagrams.
- Drew the navigational path diagram.
- Prepared the screen mock-ups.

Design Report:

- Wrote the Persistent Data Management Section.
- Co-wrote the Low-Level Design Section

Implementation Process:

- Worked on game logic.
- Co-wrote the game states.
- Co-wrote the objectives.
- Co-wrote the abilities.

Yiğit ERKAL

Analysis Report:

- Wrote the Introduction section.
- Drew sequence diagrams for Execute New Game, Change Settings, and Initial Deployment.

Design Report:

- Wrote the Hardware Software Mapping section.
- Wrote the Access Control Security section.
- Co-wrote the Low-Level Design section.

Final Report:

- Wrote the Introduction section.
- Wrote the Lessons Learnt section.

Implementation Process:

- Worked on game logic.
- Co-wrote the base classes.

Baykam SAY

Analysis Report:

- Wrote the functional requirements section
- Wrote the nonfunctional requirements section
- Drew the object and class model
  - Written the explanation to the model

Design Report:

- Wrote the introduction section (purpose of the system & design goals)
- Wrote the object design trade-offs section.
- Drew the final object design

Final Report:

- Wrote the user’s guide.
- Wrote the build instructions.

Implementation Process:

- Worked on game logic.
- Co-wrote the game states.
- Co-wrote the objectives.

Berk TAKIT

Analysis Report:

- Wrote the gameplay section
- Drew the use case diagrams

Design Report:

- Wrote the Boundary conditions section
- Co-wrote the Low-Level Design section.

Implementation Process:

- Implemented the menu, game screens.
- Implemented the game map.
- Implemented the sound engine
- Implemented the save system of the game.
- Implemented the interactions between game logic and UI.