Bilkent University

Department of Computer Engineering

CS 319 - Object-Oriented Software Engineering

Iteration 1 - Final Report

Project Name:	Risk Game
Group No:	1D
Group Name:	Risk Takers
Group Members:	Yigit Kutay GULBEN Nurlan FARZALIYEV Esad Burak ALTINYAZAR Burak YENI Burak MUTLU Anar HUSEYNOV
Supervisor:	Eray TUZUN

Contents

1. Introduction

Before all reports, we have played the online version of the game and then, decided to implement some classes, with a few attributes and methods, to make it easier for ourselves to understand the whole process and derive some diagrams from it. Basically, we have done reverse engineering. However after a while, things got a little bit complicated, so we decided to draw the rest of the diagrams first and implement the rest of the code according to the diagrams. During the group meetings, we first wrote down all possible tasks and then assigned roles to those tasks. After all, each group member has chosen some of those roles and then the rest of the implementation started. In order to collaborate synchronously, we have used GitHub.

2. Details of Implementation Process

After the first iteration, we initially started the implementation by fixing the bugs of the game that were generated after the first iteration. Then, we started to implement new promised features that were missing in the first iteration. After each newly added feature, the program was tested for any possible new emerged bugs. During the Implementation, we also added some features that were not mentioned in reports. For example, background animations were implemented for both main and in-game menu. We tried to satisfy all functional and nonfunctional requirements during implementation. The distribution of tasks was in accordance with the personal skills of every group member.

3. Implemented Functionalities

Most of the promised functionalities have been implemented. Some of them were already done after the first iteration like in-game actions (attacking, fortifying and moving unit) and some components of the main menu (“about us”, “how to play” and “quit”) unless there were some bugs related to them. So after the first iteration implementation started by fixing bugs in the existing program. As an additional feature, “options” components of the main menu has been implemented. So, it is now possible to turn on or off the game music, adjust its volume and change the difficulty level of the gameplay (only for single player case). The in-game menu has also been implemented after the first iteration so that user can pause, resume and exit from the gameplay. “Play game” component of the main menu has been modified so that the user is now also able to play single player mode in addition to the multiplayer mode which was already done in the first iteration. The single-player mode is AI based. Graphics design of the program has also improved. New animations related to pixel theme have been added to both menu and gameplay. In term of gameplay functionalities, territory card feature has been added. Each territory has its own card that enables a player to use new functionalities. To conclude, the program has been updated in terms of both functionalities and graphics design to satisfy requirements.

4. Design Changes

Since we started to implement the game before the design issues in the reports, the first implementation included some deviations from the right way of implementing relations. While modeling the game through

UML diagrams and techniques, we used such inspirational implementation to guide us on how to design relations. After modeling the game on the reports, firstly we have fixed the incorrectly implemented and designed parts of the code. After that, as a group, we progressed into a much better version of the game and we tried to comply with the diagrams while implementing further modules. As the game is progressed into an advanced version, the implementation of the game started to fit into a stable state but also immutable form so that inserting additional features into the game became much complex. So, we came up with some design changes as a solution for this issue and tried to break such complexity of the software we produced.

4.1 Extended Dependencies

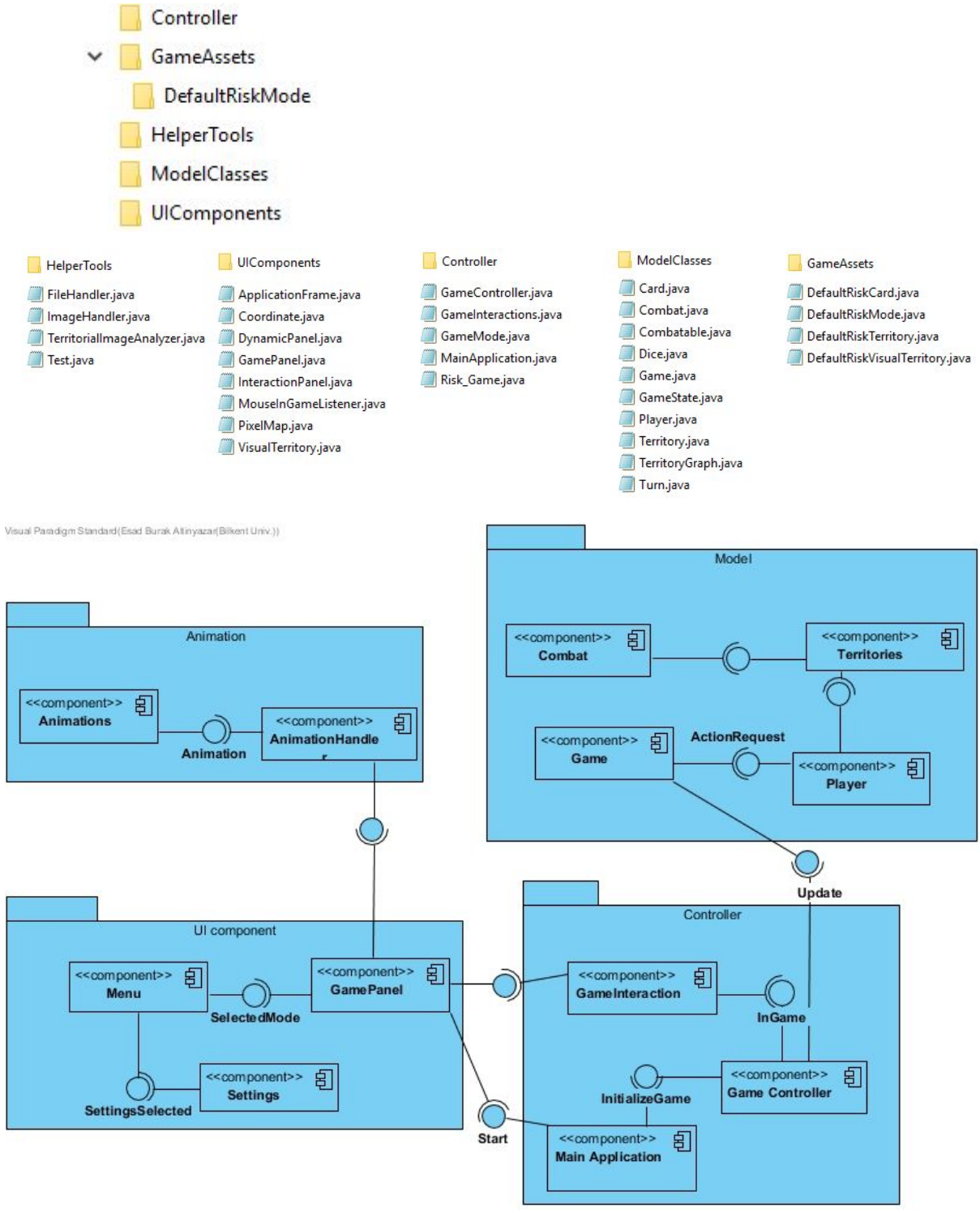
While we designing our class relations regarding with which objects included in the game, we had a thorough analysis of the game and we established our classes' relation over such analysis in a well-designed manner. That's why the implementation of the game did not deviate from the designed properties in UML diagrams. That is, we did not face the need for changing proposed relations as implementing the game by complying our design. The only modification applied differently from design is extensions of some base classes with child and specific-use purpose ones. For instance, we have classes named as Territory, Card, and TerritoryGraph for the implementation of game essentials. However, we realized that we may have different map and game mode assets as a furtherly added feature and also we had to define constants for the map and mode-specific relations. Such constants may indicate whether there is a connection between any territory within TerritoryGraph for the map selected. That is, we have to define the unique properties of the modes and maps in a specialized version of base classes. Currently, DefaultRiskMode, DefaultRiskTerritory, DefaultRiskCard are used for handling such issues. Also, the base classes for those have been converted into abstract-defined ones since they will not be used for any instantiation but will help a lot for implementing common operations with the abstraction

they provided. If there are furtherly added features as being alternative for their current ones like different map and its requirements, they will extend the base classes which are ready to use for same purposes, to shape them in the same manner.

4.2 Utility Functions

The essential mechanisms of the game implemented through model classes and their utility functions. Such utility functions include the core implementation for game functionalities but again we need other utility functions and side-definitions while implementing those. In the matter of involving side-definitions, in its basic sense interfaces, it may look like such relation does not contribute at all and it can be implemented in other ways. However, defining interface gives us an idea about what is designed role for such interface, and then, when we look the classes implements this interface, we easily comprehend what such object refers to in our model space. As being present in the current implementation, we have Combatable and GameMode interfaces. But, we decided to convert GameMode into an abstract class for its further relatives but again it supposed to be an interface-like abstract class. Additionally, we have some private sub-functions and inner classes with the purpose of helping the implementation of the proposed design. Splitting the implementation into sub-components always constitutes a fundamental technique of software development and it makes the things seem much clear.

4.3 Packages



In order not to mix every segment of the program we divided the programs to sub-packages. For instance, *UIComponents* package has only contained the programs that handle the user interface and *ModelClasses* package has the main game elements.

5. User's Guide

5.1.1 System Requirements

In order to activate the game, the user needs to have the operating system that supports the Java runtime environment. In terms of hardware requirements, both keyboard and mouse are needed. After that, all files for establishing the game should be downloaded. The memory requirement for installing the game is negligible (about 20 MB in Local Disk). The Pix-Risk game will require JVM(Java Virtual Machine) and Java SDK. Any computer which has these will be able to install and run the game.

5.1.1 User's Installation Guide

- Enter to the link below to see our reports and project https://github.com/BuR4K97/CS319_Project-Group_1.D

- Clone the project via command line or bash or Download the project as a ZIP file and run it in any Java Compiler (for example; IntelliJ, Eclipse, etc.)

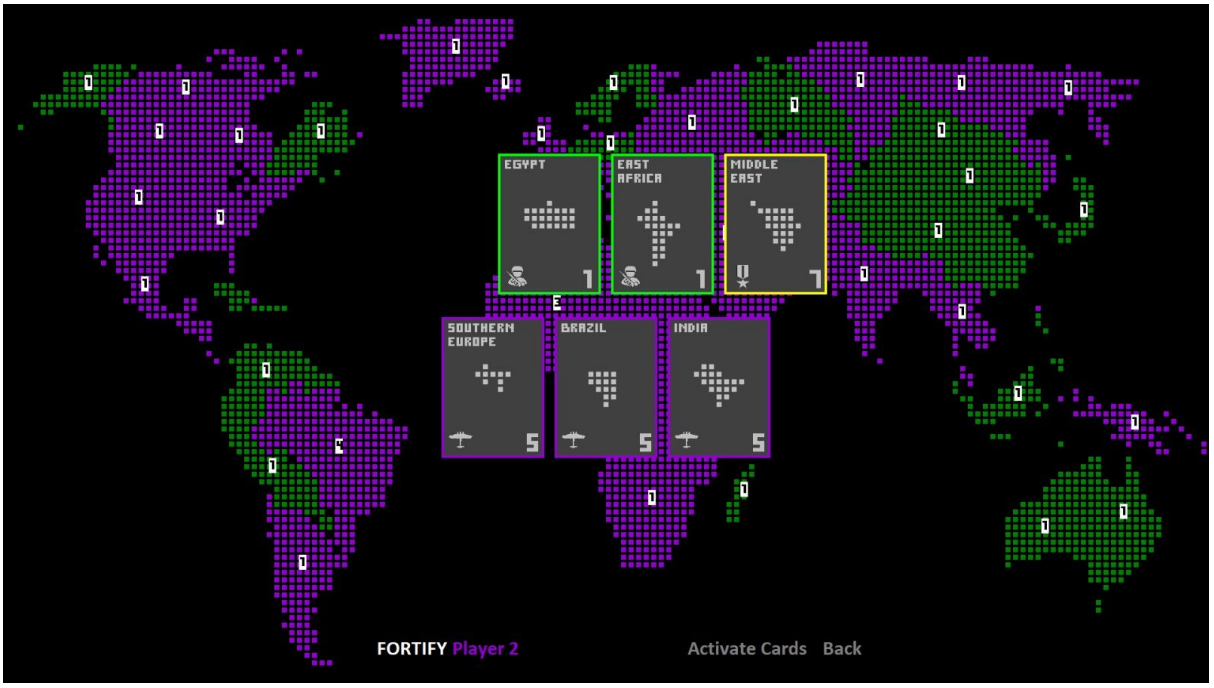
- To learn how to play game users can check "How to Play" option the menu.

- To see credits, players can "Credits" to learn who developed the game.

- For learning more about our implementation, check our second iteration reports

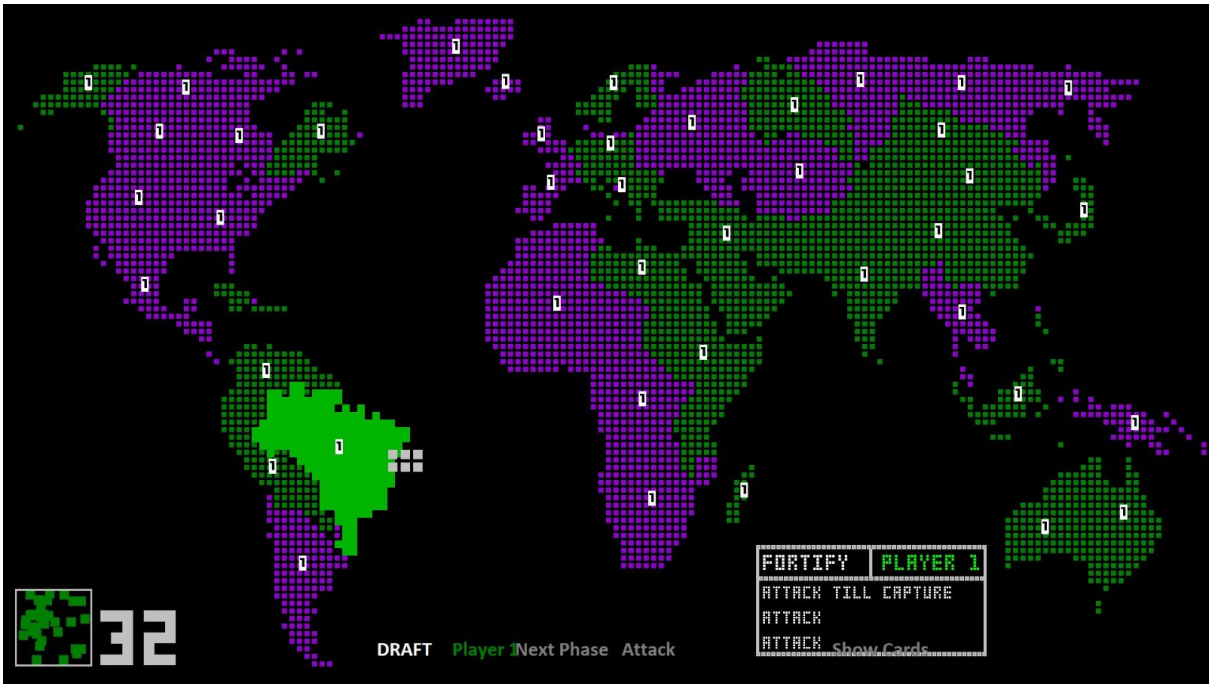
5.2 User's Manual

The installation manual is given in the above section, this section is the in-game manual.



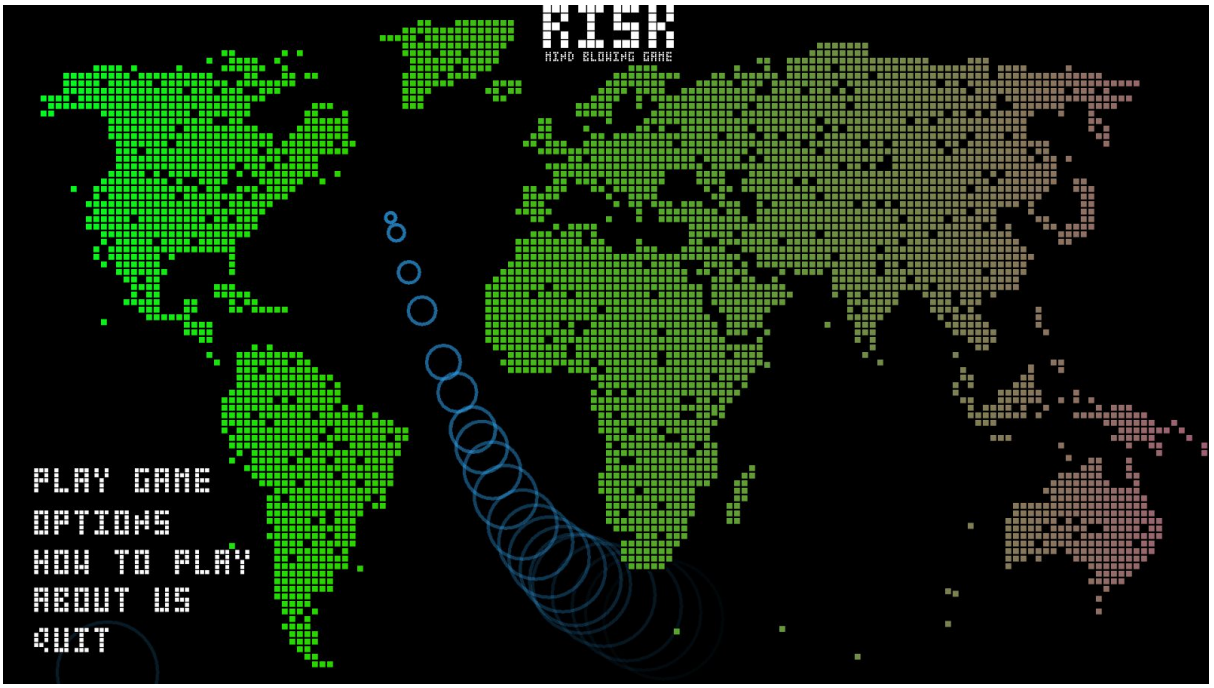
5.2.1 How to Play and Proceed to the game

In this section, users will be informed on how to play the game. Nothing related to hints will be given there. The information will be only about how to move units to the possible areas, explaining the cards and basically what is supposed to do when the game starts. Players can proceed in the game by playing their turns. In their turns, players are allowed to place new troops as the way they like into their belonging territories. They may attack another territory. When they win a battle, they will be rewarded with the card of the territory they have recently captured. And by the combinations of the cards, they may reinforce their territories with new soldier units. When a player captures all the territories in the world map the game ends.



5.2.2 Start a new game

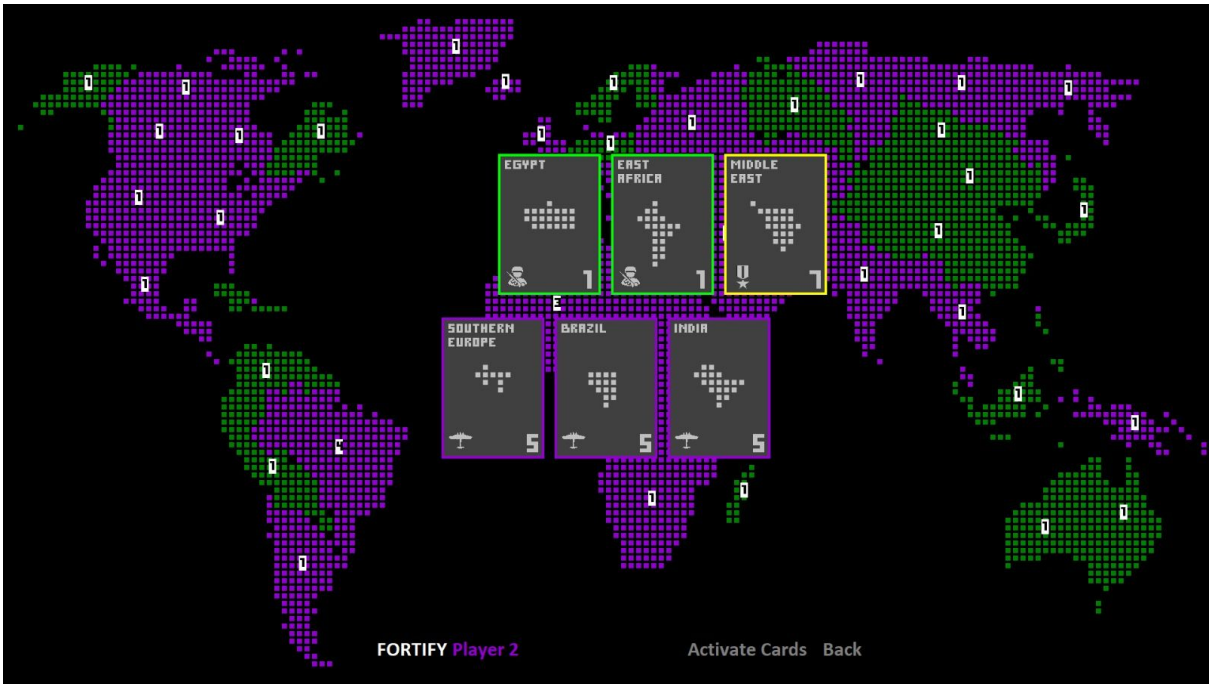
In order to start the new game, a user should firstly click on “Play Game” button. After that, the player should choose either singleplayer or multiplayer. In solo option player will play against the AI and on the multiplayer option players will play a turn-based game. After the choosing option, a new game will start.


RISK
MIND BLOWING GAME
PLAY GAME
OPTIONS
HOW TO PLAY
ABOUT US
QUIT


SELECT GAME MODE
NUMBER OF PLAYER
4
SINGLEPLAYER
MULTIPLAYER
NUMBER OF HUMAN
BACK

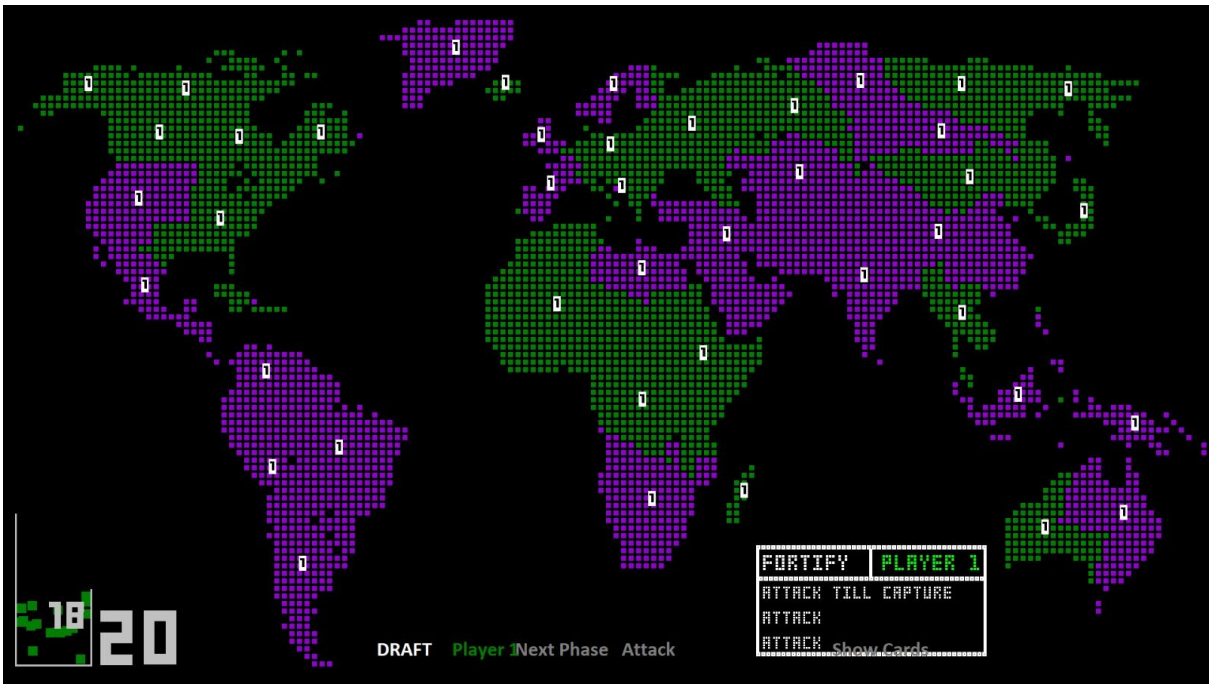
5.2.3 Card Mechanics

There are four distinct power of cards respect to weakest to strongest; Soldier, Tank, Plane, and Madal. Every Card shows a territory on them with the soldier they provide and a number that represents the power of the card. When you combine three same unit type cards such as Soldier, Soldier, Soldier, you may exchange those three cards with a Soldier to reinforce your troops. If you do the same process with three Plane Cards, you will get a Plane as a reinforcement. One exception is that if you have one Soldier, one Tank, and one Plane you will receive a Tank as a reward. But Medal is only be gained with three Medals.



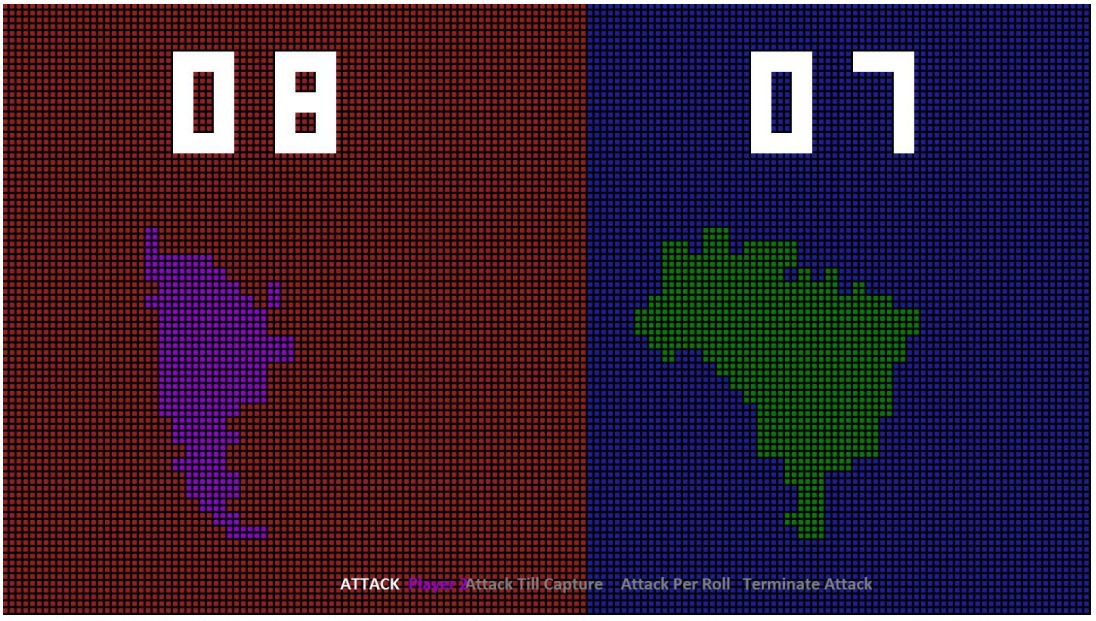
5.2.4 Add Unit

A player can click the box which is placed at the left bottom corner that shows the number of units in it, by clicking on it can get next to the curser, can place the desired territory as the desired number of units.



5.2.5 Attack a Territory

When it is your turn, you may choose the territory you want to attack if you want to attack another player. The moment you want to end the attack you can stop by clicking the “back”. Every time you attack you and your opponent roll dice but if you want to pass the attack phase, you may use the “Attack Till Capture” option which appears at the attacking animation.



5.2.6 Move Unit

A player can move his/her units from one of his/her territory to another neighbour one. Each time you left click on your territory, you load the units and then add them to neighbour territory by left clicking on that territory

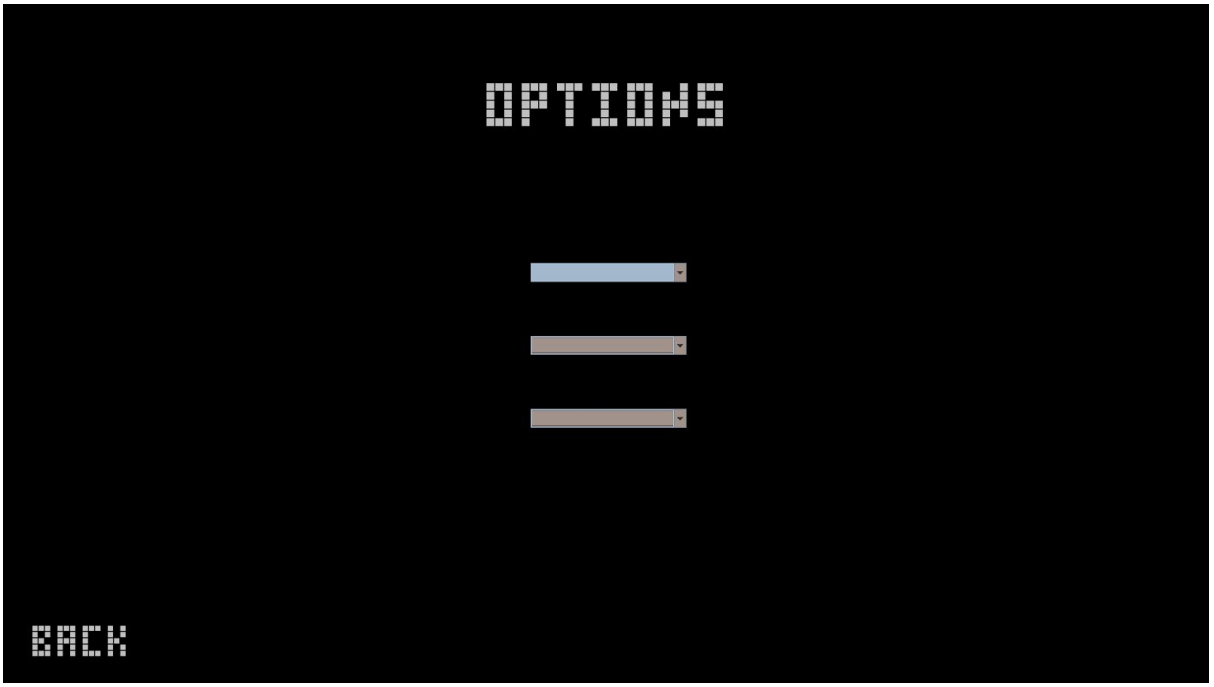
5.2.7 Pause the game

In order to pause the running game, the user can click on the “escape” button on the keyboard. In that case, the game will stop and another screen with the in-game menu will appear.

5.2.8 Resume to the paused game

After pausing the game, the in-game menu will appear. In order to resume to the game, users just need to click on the “escape” button on the keyboard. In that case, the game will start from where it was paused.

5.2.9 Go To Settings



User may go to the settings menu from the main menu or in-game menu. User may change the game's difficulty level, change music and change volume.

5.2.10 Exit from the game

To exit from the game, the player should firstly pause the game. After that, he should click on the "quit" button on the in-game menu in order to leave the game.

5.2.11 About Us

User may go to credits in the main menu to see the information about the producers.

6. Conclusion

In a general perspective, we think we did a great job. We tried to stick to “Risk of Object-Oriented Design” as much possible.Since we have only implemented functions before this course, analysing and designing before the implementation was a new experience which is important. During these processes, we have observed the advantages of the analysis and the design by living the trade-offs in the first hand. Working on a project with a planned design makes the process more faster and precise. Generally, we saw that when we have the design, implementation is nothing but coding, meaning it is rather simple. As a conclusion, we have tried our best to learn from this project and complete our tasks accordingly.