
# CS319 - Object Oriented Software Engineering

# Project Analysis Report

## RISK

## Group 2-B

Deniz Çalkan - 21703994
Irmak Akyeli- 21803690
Metehan Gürbüz - 21602687
Murat Sevinç - 21702603
Şebnem Uslu- 21802068
Yusuf Alpdemir - 21803035

# 1.Introduction

The goal of our project is to design an online version of the Risk board game. We want to improve the game so that it becomes more user friendly and more fun. In order to do that we will add new features and extend some of the existing features.

Risk is a strategy game in which the goal is to conquer the world by capturing every region on the map. To capture a region, the player attacks another player's region. At this point two players' troops engage in a battle. Depending on the number of troops players roll a dice and depending on the dice the player either defeats the other player or gets defeated [1]. The goal is simple but the game itself is very dynamic. The game depends on both luck and skill of the player but we want our game to focus more on skill rather than luck. Moreover, we will be extending the dice, troop and region components of the game.

# 2.Overview

Originally the game can be played with 2 players. Since, the game will go back and forth between the same 2 players, we thought setting the minimum number of players to 3 will make the game more fun. So, there should be at least 3 and at most 6 players to play the game. The game starts and each player is assigned to a specific color to represent them.

## 2.1 Game Setup

Depending on the number of players, each player gets assets.

- If there are 3 players , each player gets 35 assets.
- If there are 4 players , each player gets 30 assets.
- If there are 4 players , each player gets 25 assets.
- If there are 5 players , each player gets 20 assets.

After distributing the armies each player rolls a die. The player who rolls the highest number starts placing their assets first. Then other players start placing their assets on unoccupied regions in order according to their rolled numbers. Players continue until all regions have been occupied. After all regions are occupied each player adds additional assets to their occupied regions and there is no limit for adding assets to a single region. When there are no armies left, risk cards are shuffled and the first player starts the game [2].

## 2. 2 Game Play

Each turn consists of 3 stages. Overall, the player adds the new soldiers to his/her regions, attacks and moves his/her soldiers at the end of the turn. Each turn starts with new soldiers obtained by the player. The number of the soldiers added are calculated with bonuses and regions that the player holds. During this stage the player can check his/her cards to see whether there is a bonus obtained. After this stage the player now proceeds to attack. Any regions holding 2 or more armies can attack a neighboring region and dice are used to choose the winner. There are advantages that a player can have and these advantages affect the dice

roll, for example if the attacked region is the capital of another player the defense side rolls the dice twice and takes the greatest result. A player can attack as much he/she wants as long as he/she has 2 or more army remaining in one of his/her regions that has at least one enemy neighbor. Once a player cannot attack anymore or chooses not to, the second stage ends and the last stage of a turn starts. During this last stage a player can move his/her existing armies that have at least 2 size to one of his/her other regions. This stage ends when one replacement is done or if it is skipped [2].

### 2.2.1 Getting and Placing New Armies

At the beginning of each turn players receive new armies based on the number of regions they occupied, value of the continents they hold , value of the matched set cards and other bonuses like the number of capitals they hold.

#### A. Regions

Players receive armies according to their occupied regions divided by three. Players can place their armies on their already occupied regions. Players will receive at least 3 armies even if their occupied regions are less than 9.

#### B. Continents

Players receive armies for each continent they hold. In order to hold a continent, players must occupy all the regions on the continent. Each continent has its own bonus points and it activates after the player holds a continent at the beginning of his/her turn.

#### C. Capitals

Each player chooses a capital at the beginning of the game after all the initial soldiers are placed that cannot be changed during the game session. This capital gives 2 bonus points while the player is still controlling the region. If a player attacks and invades another player's capital, it gains the bonus points given by the capital. Therefore, the numbers of capital controls affects the number of soldiers obtained each turn.

### 2.2.2 Risk Cards

At the end of a turn in which the player captures at least one region, the player gains one risk card. The main goal is to collect sets of 3 cards in one of the combinations:

- 3 cards of the same type (Infantry, Cavalry, or Artillery)
- 1 from each type (Infantry, Cavalry, and Artillery)
- Any 2 cards and a wild card

At the beginning of each turn players who have a set of cards that are one the above conditions can trade their set with armies. Each set has its own value in terms of troops.

Wild cards are like joker cards, if a player has 2 artillery cards, it behaves like the third artillery card and provides a set.

If a player has more than five cards at the beginning of the round, the system enforces that user to trade a set since there is always one set when 5 cards exist.

### 2.2.3 Attacking

When a player's turn comes he/she can attack other players' regions. The main goal of attacking is to capture an opponent's region by defeating all the armies that are currently in the region. The defeater of the battle is decided by roll of dice. The players may choose not to attack when their turn comes but can still fortify their armies.
In order to attack the player should satisfy the following conditions:

- The player can only attack a region which is next to or connected to his/her region.
- The player must have at least 2 armies in the region which he/she attacks from.

The player can attack a region till there are no opponent's armies left. Also, the player can attack regions as much as possible and can switch from one region to another during his/her turn.

When the player chooses to attack it is pointed on the map where the player is attacking from and where the player is attacking [2].

#### A.Dice rolls

In our system of dice roll there will be a number generator that will generate a number between 1 and 100. This corresponds to two dice that are used by D&D players, both of them with 10 faces that gives the player a total number of 100 different possibilities. The player that rolls the highest number wins the round and if there is an advantage to one of the players, the player uses the best of two rolls to fight.

#### B.Capturing Regions

When there are no opponent's armies left where the region that the player has attacked, he/she captures the region. In this region, remaining armies from the previous battle are placed. The logic should be keeping as many armies as possible on the front regions since the armies on the back can't help the player during a battle. Also, the player must leave at least one army in the region that he/she has attacked from.

#### C. Ending Attacks

The player can end his/her attacks when desired. If the player has captured at least one region he/she takes a one risk card.

#### D.Eliminating an Opponent

When the player eliminates all armies of another player, the second player is eliminated from the game and the first player gets all the cards that the second player has previously owned.

### 2.2.4 Fortifying Position

While always leaving one army behind, the player may choose to fortify their armies from only one region into only one other region that is his/hers. There are no requirements to fortify armies.

### 2.2.5 Winning

The player who captures all the regions wins the game.

# 3. Functional Requirements

## 3.1 Number of Players

- The game should be available for 3 to 6 players that will be determined by the host.

## 3.2 Gameplay

During each step of the gameplay, players should be able to clearly see all of their options and should be able to easily navigate.

- Each player should be assigned to a specific color that will represent them throughout the game.
- According to the number of players, the corresponding number of assets should be distributed to each player.
- Map should be displayed on the screen.
- Players should be able to select a region on the map.
- All regions should be colored according to the player who occupies the region.
- Players should be able to choose a region to attack.
- Players should be able to roll a dice.
- Players should be able to view their Risk cards.
- Players should be able to end their attacks when desired.
- When a player is eliminated all of his/her risk cards should be given to the player who eliminated him/her.
- When the game ends, players should be able to see the winner.

## 3.3 Help

At any point of the game:

- Players should  be able to get further information about how to play the game.

- Players should be able to get information about the rules of the game.

### 3.4 Quit Game and Resign

- Players should be able to quit the game without corrupting the flow of the game.
- Players should be able to resign and other players should be informed.

### 3.5 Host and Join Options

- Players should be able to host a game or join an existing game.
- Host should be able to select the number of players.
- Host should get a room ID.
- Players who have the room ID should be able to join the game.

## 4. Nonfunctional Requirements

### 4.1 User Interface

One of the first aspects that users notice about a software is GUI. So our Risk game will have a GUI that is aesthetically appealing to users and easy to navigate [3]. We will have a simple yet elegant GUI. For example, there won't be unnecessary buttons or labels on the screen and all buttons will have meaningful and simple titles such as ‘START’ button, ‘QUIT’ button, ‘HELP’ button and so on. Also, ‘HELP’ button will navigate the users throughout the game. The users will be able to press this button at any point of the game to get further information about how to play the game.

### 4.2 Performance

A game with a good performance should have reasonable frame rate and short loading times. Our goal is to keep our game running with 60 FPS if we implement action driven GUI such as attacking regions, rolling dice and placing assets. Also we want to keep our response time less than a second.

### 4.3 Extensibility

Our game will be implemented in a way that it is easy to make new modifications, fix bugs and remove undesired features. In order to do that, an object oriented design will be used where all the classes have single and distinct functionality. Also, we are implementing our game in Java and Java is an object oriented programming language. So, it will be easier for us to make our game extensible.

# 5. System Models

## 5.1 Use Case Model

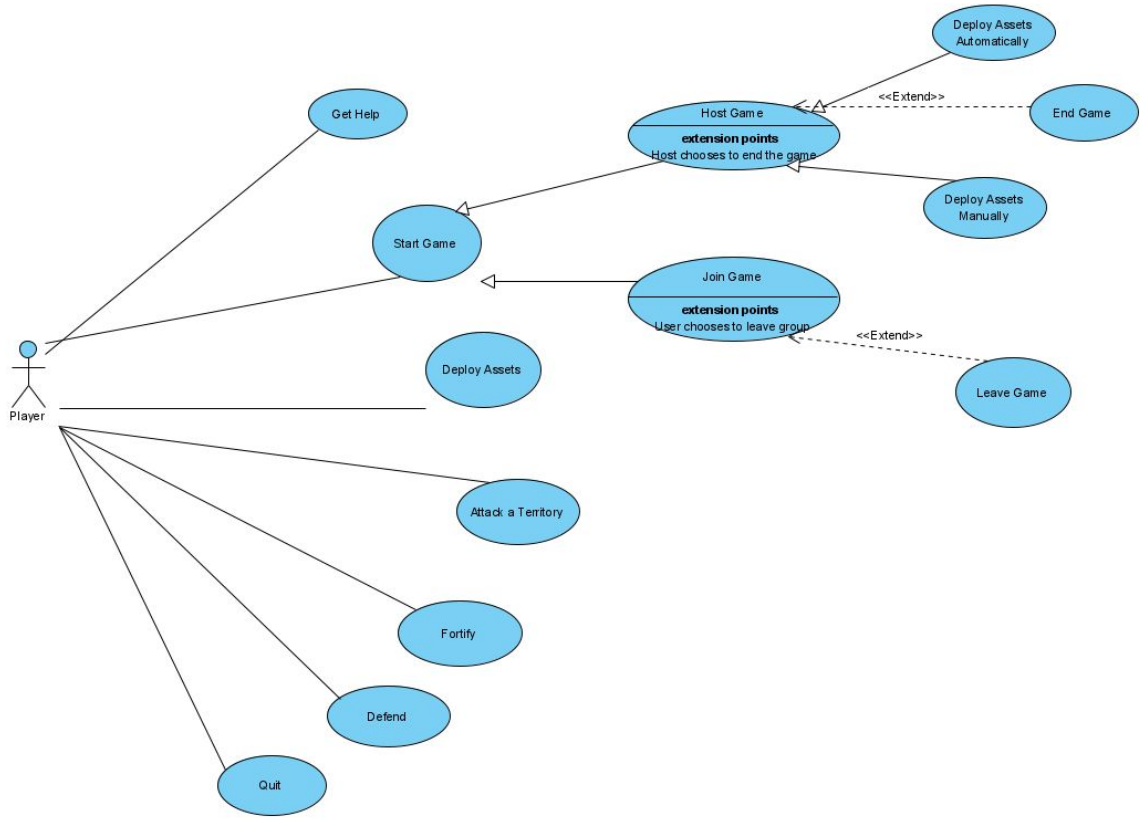


Figure 1: Main Use Case Diagram

| **Use Case Name** | Get Help |
|---|---|
| **Use Case ID** | 1 |
| **Primary System Actor** | Player |
| **Definition** | If the users choose to, they can learn more about the rules of the game. |
| **Precondition** | - |
| **Trigger** | The user clicking on the “Help” button. |

| | |
|---|---|
| **Basic Path** | 1. User chooses the “Help” button in the main menu.<br>2. The rules of the game is given in a list in the Help page |
| **Alternative Path** | - |
| **Exceptional Case** | - |
| **Postcondition** | User chooses to exit the help page. |
| **Rules** | - |

| | |
|---|---|
| **Use Case Name** | Start Game |
| **Use Case ID** | 2 |
| **Primary System Actor** | Player |
| **Definition** | The player starts a game and chooses either to join a game or host one. |
| **Precondition** | - |
| **Trigger** | The button named “Play” is clicked to start a game. |
| **Basic Path** | 1. Player chooses to start a new game from the main menu.<br>2. The player chooses if they are to host a game or join one. |
| **Alternative Path** | - |
| **Exceptional Case** | - |
| **Postcondition** | After the choices are made to start the game, the player advances to the Deploy Assets stage. |
| **Rules** | 1. The user must choose one of the two options given which are hosting or joining a game. |
| **Explanation** | - |

| | |
|---|---|
| **Use Case Name** | Host Game |
| **Use Case ID** | 2.1 |
| **Primary System Actor** | Player |

| | |
|---|---|
| **Definition** | The player chooses to host a game by creating a room. |
| **Precondition** | The player must choose to host the game from the Play Game page. |
| **Trigger** | The button named “Host” is clicked to create a room. |
| **Basic Path** | 1. Player chooses to host the game.<br>2. Player (host) enters their username.<br>3. Player (host) chooses manual or automatic deployment.<br>4. Player (host) chooses the number of players in the game. |
| **Alternative Path** | - |
| **Exceptional Case** | The user can choose to quit the game. |
| **Postcondition** | After the game is hosted by the user, the room page will show up. |
| **Rules** | 1. Only the host can choose the number of players in the game.<br>2. Only the host can choose if the deployment of assets is automatically done.<br>3. The host must begin the game.<br>4. The host cannot begin the game if the stated number of players requirement is not fulfilled. |
| **Explanation** | The automatic deployment option lets the computer deploy the initially given assets to be deployed randomly, saving time for players who do not want to spend time on that. After the deployment phase is done automatically, the game continues the same way as the basic path and each player begins with the deploy assets phase. |

| | |
|---|---|
| **Use Case Name** | Deploy Assets Automatically |
| **Use Case ID** | 2.1.1 |
| **Primary System Actor** | Player |
| **Definition** | Before the game starts, the host chooses to deploy assets automatically in the beginning of the game. |
| **Precondition** | The player must choose the “Deploy Assets Automatically” feature in the page. |
| **Trigger** | The switch named “Automatic Play” is switched on to add the feature. |

| | |
|---|---|
| **Basic Path** | 1. Player chooses to deploy assets automatically by clicking on the button.<br>2. The game randomly deploys assets for the players before the deployment stage. |
| **Alternative Path** | - |
| **Exceptional Case** | - |
| **Postcondition** | After the game deploys the assets, it will begin with the deployment phase. |
| **Rules** | 1. Only the host can choose this feature. |
| **Explanation** | - |

| | |
|---|---|
| **Use Case Name** | Deploy Assets Manually |
| **Use Case ID** | 2.1.2 |
| **Primary System Actor** | Player |
| **Definition** | Before the game starts, the host chooses to deploy assets manually in the beginning of the game. |
| **Precondition** | The player must choose the “Deploy Assets Manually” feature in the page. |
| **Trigger** | The switch named “Automatic Play” is switched off to add the feature. |
| **Basic Path** | 1. Player chooses to deploy assets manually by clicking on the button.<br>2. The game does not randomly deploy assets for the players before the deployment stage. |
| **Alternative Path** | - |
| **Exceptional Case** | - |
| **Postcondition** | The game will begin without any deployment done and the players will deploy their assets manually. |
| **Rules** | 1. Only the host can choose this feature. |
| **Explanation** | - |

| **Use Case Name** | Join Game |
|---|---|
| **Use Case ID** | 2.2 |
| **Primary System Actor** | Player |
| **Definition** | The player chooses to join a game by joining a room created by a host. |
| **Precondition** | The player must choose to join a game from the Play Game page. |
| **Trigger** | The button named “Join” is clicked to join a room. |
| **Basic Path** | 1. Player chooses to join the game.<br>2. Player enters the code.<br>3. Player enters their username.<br>4. Player clicks on the “Ready” button on the room page. |
| **Alternative Path** | - |
| **Exceptional Case** | The player may quit the game. |
| **Postcondition** | After the game is started by the host, it will begin with the deployment phase. |
| **Rules** | 1. The player can get out of the room<br>2. The player cannot begin the game<br>3. The player cannot choose the automatic deployment option or number of players in the game. |
| **Explanation** | The ready button referred to on the fourth bullet in the basic path is for the player to let the host know they are ready to play the game but it is not necessary for the game to start. |

| **Use Case Name** | Deploy Assets |
|---|---|
| **Use Case ID** | 3 |
| **Primary System Actor** | Player |
| **Definition** | Each player adds their remaining assets to a region. |
| **Precondition** | The host must have started the game. |
| **Trigger** | The turn of the player begins. |

| | |
|---|---|
| **Basic Path** | 1. Player's turn begins.<br>2. Players add given assets to their owned regions. [A-1][A-2][A-3][A-4] |
| **Alternative Path** | [A-1]: If the game mode does not automatically deploy assets, in the beginning of the game, each player will start deploying assets in empty regions.<br>[A-2]: The player can choose to deploy infantry.<br>[A-3]: The player can choose to deploy cavalry.<br>[A-4]: The player can choose to deploy artillery. |
| **Exceptional Case** | - |
| **Postcondition** | After the troops are deployed, the player advances to the Attack stage. |
| **Rules** | In the alternative case of [A-1]:<br>● The region the player deploys an asset on, will be owned by that player.<br>● The first region owned by the player becomes their capital.<br>● If the empty regions are finished but the player has additional assets to deploy, the player will deploy these assets on their owned region. |
| **Explanation** | In the alternative path, each player will add their assets to empty or their owned regions until the given assets are depleted. After all initially given assets are deployed, the game will carry on to the attack stage. |

| | |
|---|---|
| **Use Case Name** | Attack a Region |
| **Use Case ID** | 4 |
| **Primary System Actor** | Player |
| **Definition** | The player attacks an opponent player's region |
| **Precondition** | - |
| **Trigger** | After choosing one of their regions, the player attacks one of the suitable enemy regions. |
| **Basic Path** | 1. Players assemble an attack configuration by choosing the asset types available in their region.<br>2. Players can attack other regions if they choose to. |
| **Alternative Path** | - |

| | |
|---|---|
| **Exceptional Case** | - |
| **Postcondition** | After the attacks are finished, the player advances to the fortify stage. |
| **Rules** | 1. The attack configuration can include one or many of the assets.<br>2. When attacking, at least one of any deployed assets must remain in the attacker's region.<br>3. The engagement result between the rival assets will be determined based on their relative odds of winning.<br>4. When determining the engagement result, the status of the region will be taken into account (i.e if the defending player's region is the capital, then odds of winning will be higher) |
| **Explanation** | - |

| | |
|---|---|
| **Use Case Name** | Fortify |
| **Use Case ID** | 5 |
| **Primary System Actor** | Player |
| **Definition** | The player relocates their assets to one of their owned regions. |
| **Precondition** | The attack stage must have ended. |
| **Trigger** | From one of their regions to another, the player chooses to relocate the assets. |
| **Basic Path** | 1. In their turn, the player chooses one of their regions in the fortify stage.<br>2. The player chooses the types and numbers of assets to relocate.<br>3. The player chooses the second location the assets will relocate to. |
| **Alternative Path** | - |
| **Exceptional Case** | - |
| **Postcondition** | After the relocations are finished, the player will wait for other players until it is their turn for the deployment stage. |
| **Rules** | 1. When fortifying, at least one of any deployed assets must remain in the player's regions. |
| **Explanation** | - |

| | |
|---|---|
| **Use Case Name** | Defend |
| **Use Case ID** | 6 |
| **Primary System Actor** | Player |
| **Definition** | The player defends their position if another player attacks them. |
| **Precondition** | The turn of this player must have ended. |
| **Trigger** | An owned region is attacked by another player. |
| **Basic Path** | 1. In their turn, a rival player chooses to attack this player.<br>2. The defender will automatically roll a dice. |
| **Alternative Path** | - |
| **Exceptional Case** | - |
| **Postcondition** | After the relocations are finished, the player will wait for other players until it is their turn for the deployment stage. |
| **Rules** | - |
| **Explanation** | The player cannot do anything to change the outcome in this stage. The outcome will be determined automatically. |

| | |
|---|---|
| **Use Case Name** | Quit |
| **Use Case ID** | 6 |
| **Primary System Actor** | Player |
| **Definition** | The player quits the game. |
| **Precondition** | - |
| **Trigger** | Player clicks the quit button. |
| **Basic Path** | 1. On any page, the player clicks the quit button to exit the game. |
| **Alternative Path** | - |

| | |
|---|---|
| **Exceptional Case** | - |
| **Postcondition** | - |
| **Rules** | - |
| **Explanation** | After the player exits the game, they will forfeit from the game if there is an ongoing match. |

## 5.2 Dynamic Models

### 5.2.1 Activity Diagram

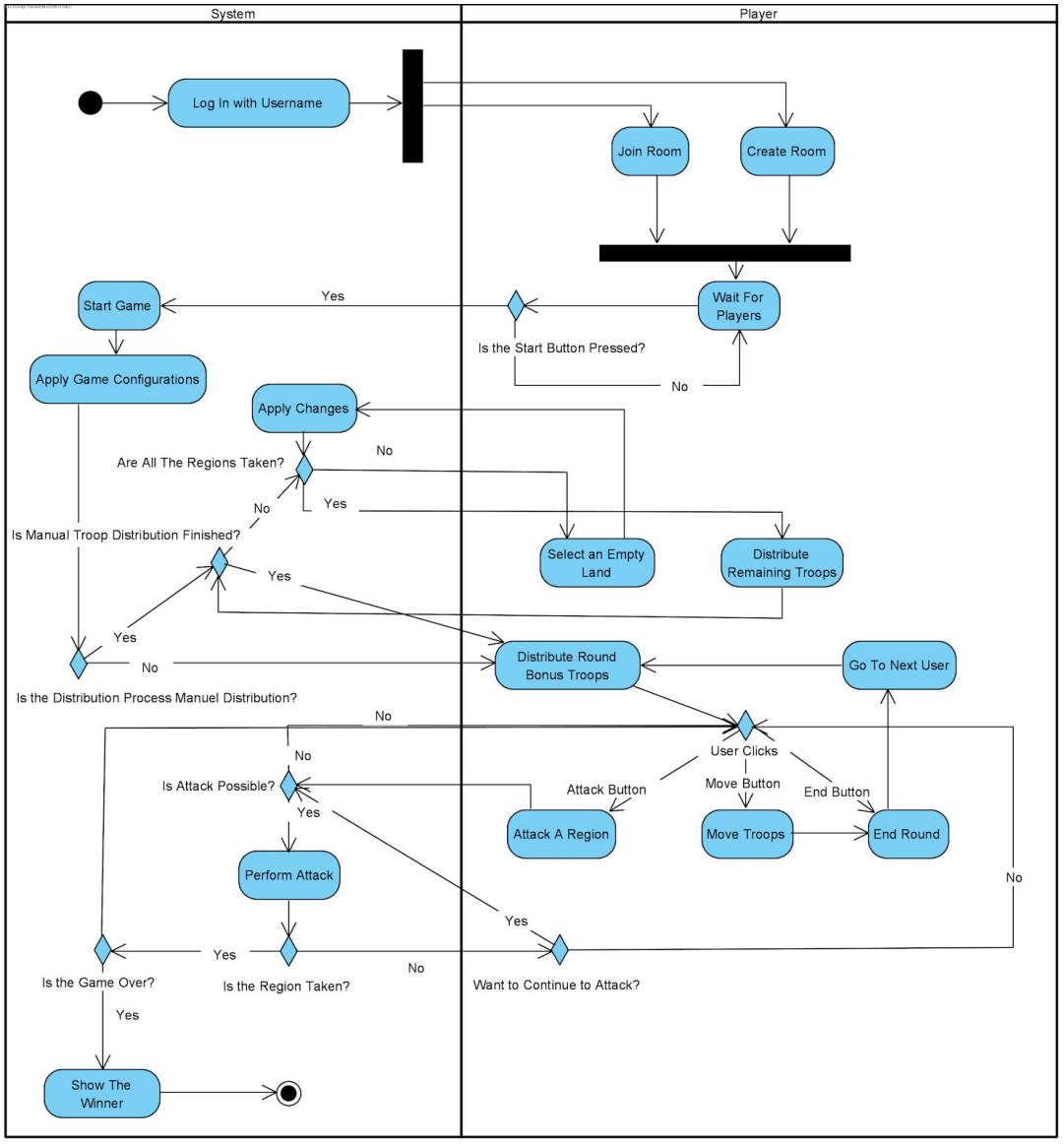


Figure 2: Activity Diagram

Since our Risk game will be an online game, the game starts with a dialog which asks the username to the player. After a user logs in the main page of the game is shown.
In the main page, users can either join a room or create a room. After game configurations are done, the game starts with the troop distribution phase which is the first phase of every game. After all players distribute their all the troops, second and the main phase of the game begins. Every user gets a certain number of troops in the beginning of each round. After the distribution of new troops, the player has three options: attacking, moving troops or ending the round. The player can attack multiple times if possible. If a player gets a region, they receive a risk card and the system checks whether the game is over or not. A player can either end the turn directly or the round can be ended after the troops fortification.

### 5.2.2 State Diagrams

#### A. States for Attack

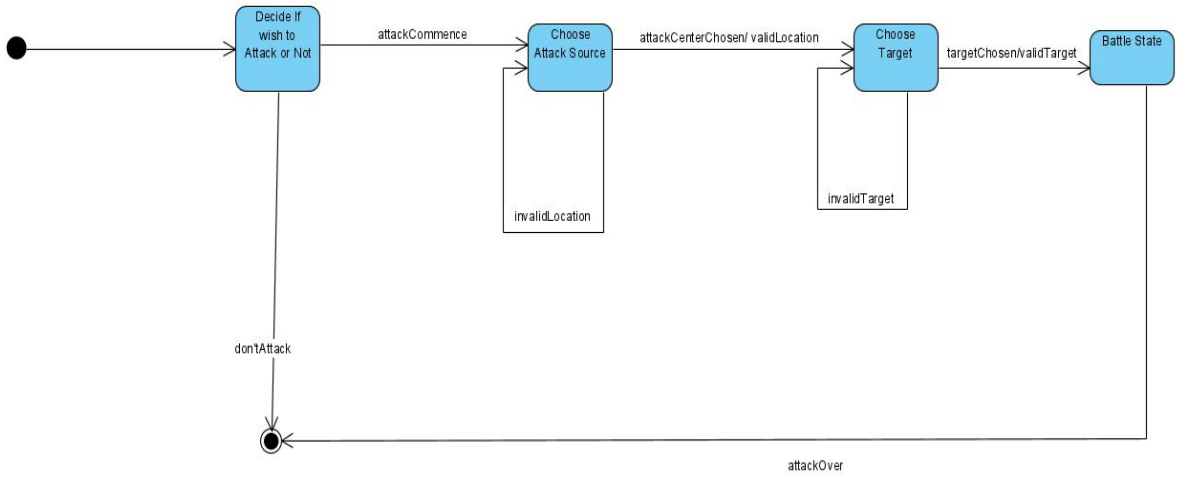


Figure 3: State Diagram for attack

#### a.Initial State

Attack phase begins.

#### b. Attack Decision State

User attacks or not.

#### c. Choosing Attack Source State

Choose the location where the attack is launched from.

#### d.Choosing Target State

Choose the location of the attacked place.

### e.Battle State

The battle takes place between the attacker and the attacked, goes to the final state after the battle is over.

### f.Final State

Game has ended.

## B. States For the Player

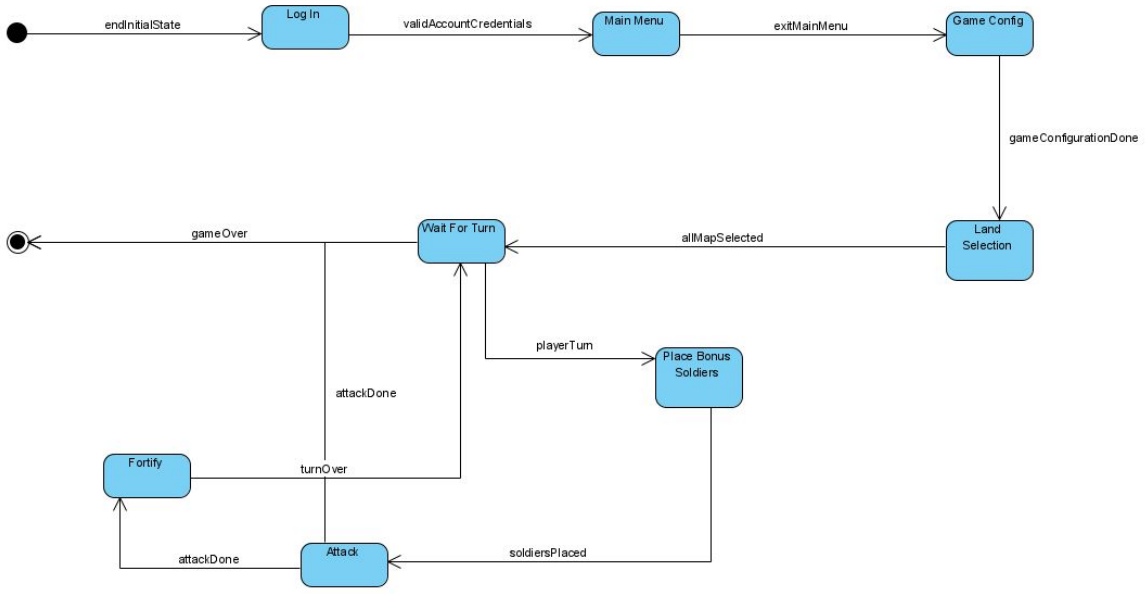


Figure 4: State Diagram for the Player Object

### a. Initial State

Player is at the opening screen of the game.

### b. Login State

Players login. If the credentials are correct, advance to the next state.

### c. Main Menu State

This is the main menu that players can use to launch a game config screen.

### d. Game Configuration State

This is the menu where the player will launch a game by either hosting or joining. Advance to the next state when the game is launched.

### e. Land Selection State

As the game is launched, the player selects the land when the selection turn is theirs.

### f. Waiting For Turn State

After the land selection state the main game starts. In the main game this is the state where the player will mostly spend their time. State ends when it's the player's turn or if a different player has all the land.

### g. Placing Bonus Soldiers State

When it's the players turn to play, firstly they place the bonus soldiers that they get at the start of every round. This is the state when that happens. State is over when all the soldiers are placed.

### h. Attacking State

This is the state when the player attacks if they choose to. Complete the attack or skip the attack to advance to the fortify state. If the player acquired the whole map after the attacks advance to the main menü.

### i. Fortifying State

This is the state where the player reinforces their troops, shuffles them for tactical purposes if they choose to. Advance to the next state when the player is pleased with their troops' locations. After this state, the player again advances to "Wait For Turn" State.

### j. Final State

Game has ended.

## 5.3 Object and Class Model

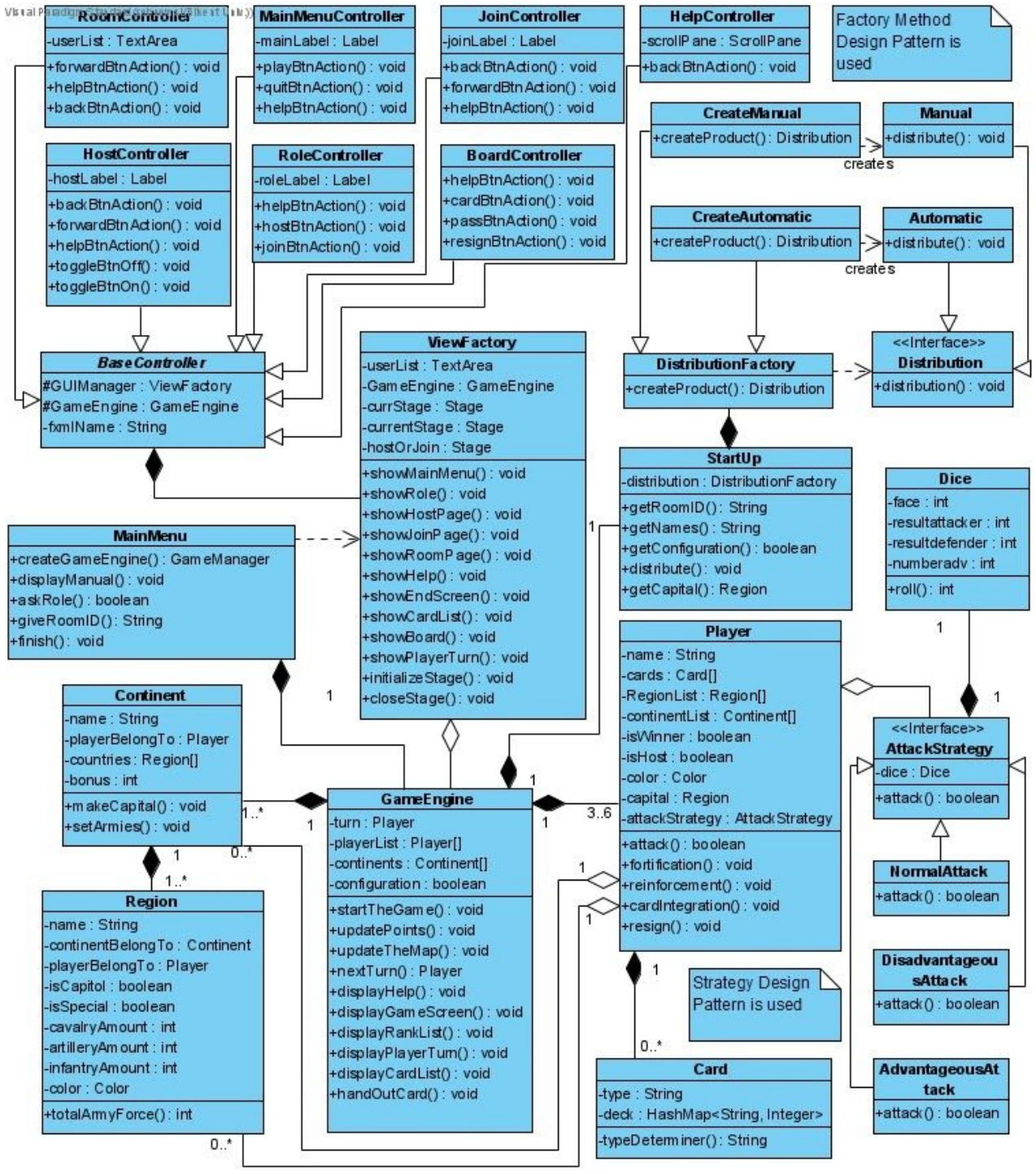


Figure 5: Class Diagram

### 5.3.1 ViewFactory Class

ViewFactory class has been established due to the necessity that emerged for the interaction between the players. ViewFactory provides the user interface by calling the relevant method. It is called by MainMenu and GameEngine.

### 5.3.2 MainMenu Class

MainMenu class handles the beginning of the game. It takes the input from the user for the options: Manuel, Play, and Exit. In order to offer the user this selection in a proper interface, it calls ViewFactory. If the user selects to play, it calls GameEngine.

### 5.3.3 StartUp Class

StartUp class was designed for the beginning functions of the game. If the player selects the “Play” option, GameEngine will call StartUp to determine player names, configuration status and the initialization of the game.

### 5.3.4 GameEngine Class

GameEngine class is the main controller of the game. It is created when the MainMenu calls it after the player selects the “Play” option. Other attributes of the game such as Player, Continent, Card, and StartUp are attached to it and created if GameEngine calls them. Furthermore, GameManager calls ViewFactory so that the changes upon the game can be reflected to the player through the user interface that ViewFactory provides.

### 5.3.5 Card Class

Card class represents a Card that will be drawn within the game. Cards can provide cavalry, infantry... etc. Thus, a card should have a property that indicates the type. The type of the card should be random so that the player cannot count the cards to determine what will come. typeDeterminer() provides a random card to be created. Because this method is only used when the card is created which is in the constructor, the method will stay private.

### 5.3.6 Player Class

Player class will be created when the Game Manager calls it. Thus, the existence of the Player is dependent on the GameEngine. If there is no game to play, there will be no player. The player has multiple properties as well as methods. Players will have the attributes of a name, a Region list to track down the conquered countries, the continent list so that at the next round, the player can receive a bonus for conquering an entire continent, a boolean to determine whether the exact player is the winner one or not. Lastly, the rank of the player.

### 5.3.7 Region Class

The creation of the Region class is dependent on the existence of the Continent class. If only if the Continent class is ever created, a list of Region objects will be created. Region class has 8 attributes: the name of the Region, the owner of the Region, the continent that the Region is within, is it a capital or is it a special city like Rome, and lastly, the numbers of three different kinds of soldiers deployed in the Region. There is only one method that this class contains which calculates the total army force according to the amount of the different types of soldiers.

### 5.3.8 Continent Class

The creation of the Continent class is dependent on the existence of the GameEngine class. If only if GameEngine class is ever created, a list of Continent objects will be created. Continent class has 3 attributes: the name of the continent, the owner of the continent, and the list of countries that the continent includes within. This class works as the board of the game which is the map.

### 5.3.9 Dice Class

The existence of the Dice class is dependent on the AttackStrategy interface, since if there is no attacking there is no need for the dice. Dice class takes two army and an advantage indicator and returns the army lost from a side after roll function.

### 5.3.10 RoomController

RoomController class has been established to provide functions to buttons of the Room window. RoomController manages the attributes and actions related to the window that it controls. It extends the abstract class BaseController and uses its properties to call the view functions.

### 5.3.11 RoleController

RoleController class has been established to provide functions to buttons of the Role Choosing window. RoleController manages the attributes and actions related to the window that it controls. It extends the abstract class BaseController and uses its properties to call the view functions.

### 5.3.12 JoinController

The JoinController class has been established to provide functions to buttons of the Join window. JoinController manages the attributes and actions related to the window that it controls. It extends the abstract class BaseController and uses its properties to call the view functions.

### 5.3.13 HostController

HostController class has been established to provide functions to buttons of the Host window. HostController manages the attributes and actions related to the window that it controls. It extends the abstract class BaseController and uses its properties to call the view functions.

### 5.3.14 MainMenuController

MainMenuController class has been established to provide functions to buttons of the Main Menu window. MainMenuController manages the attributes and actions related to the window that it controls. It extends the abstract class BaseController and uses its properties to call the view functions.

### 5.3.15 HelpController

HelpController class has been established to provide functions to buttons of the Help window. HelpController manages the attributes and actions related to the window that it controls. It extends the abstract class BaseController and uses its properties to call the view functions.

### 5.3.16 BoardController

BoardController class has been established to provide functions to buttons of the Game window. BoardController manages the attributes and actions related to the window that it controls. It extends the abstract class BaseController and uses its properties to call the view functions.

### 5.3.17 BaseController

BoardController class has been established to provide functions to buttons of the Game window. BoardController manages the attributes and actions related to the window that it controls. It extends the abstract class BaseController and uses its properties to call the view functions.

### 5.3.18 CreateAutomatic

CreateAutomatic class was designed for the automatically soldier distribution options. It calls and creates an instance of Automatic class and returns the object of it to be used in the StartUp class.

### 5.3.19 CreateManual

CreateManual class was designed for the manually soldier distribution options. It calls and creates an instance of Manual class and returns the object of it to be used in the StartUp class.

### 5.3.20 DistributionFactory

The DistributionFactory class was designed for the soldier distribution options. Classes that extend this class creates the related class option which determines the distribution method functionality which differs as automatically or manually, and returns the created class instance.

### 5.3.21 Distribution (Interface)

Distribution interface was designed for the soldier distribution options. Classes that implement this interface determine the distribution method functionality which differs as automatically or manually. The interface usage gives clean and less complicated codes for us.

### 5.3.22 Automatic

Automatic class was designed for the automatic soldier distribution option. If the host selects the “On” on the toggle switch, StartUp will call DistributionFactory to create the

Automatic Class instance which will result in the distribute() method at the StartUp class to distribute soldiers automatically.

### 5.3.23 Manual

The Manual class was designed for the manual soldier distribution option. If the host selects the “Off” on the toggle switch, StartUp will call DistributionFactory to create the Manual Class instance which will result in the distribute() method at the StartUp class to distribute soldiers manually.

### 5.3.24 AttackStrategy (Interface)

The AttackStrategy interface was designed to be able to use strategy design pattern in our attacking component since there are different types of attacks. Payer class is the class that uses this strategy.

### 5.3.25 NormalAttack

Normal attack class was created to perform attack strategy when there is no advantage for either side (attacker and attacked).

### 5.3.26 DisadvantageousAttack

Disadvantageous attack class was created to perform attack strategy when there is an advantage for the attacked side.

### 5.3.27 AdvantageousAttack

Advantageous attack class was created to perform attack strategy when there is an advantage for the attacker side.

## 5.4 Sequence Diagram

### 5.4.1 Sequence Diagram for the Start of the Game

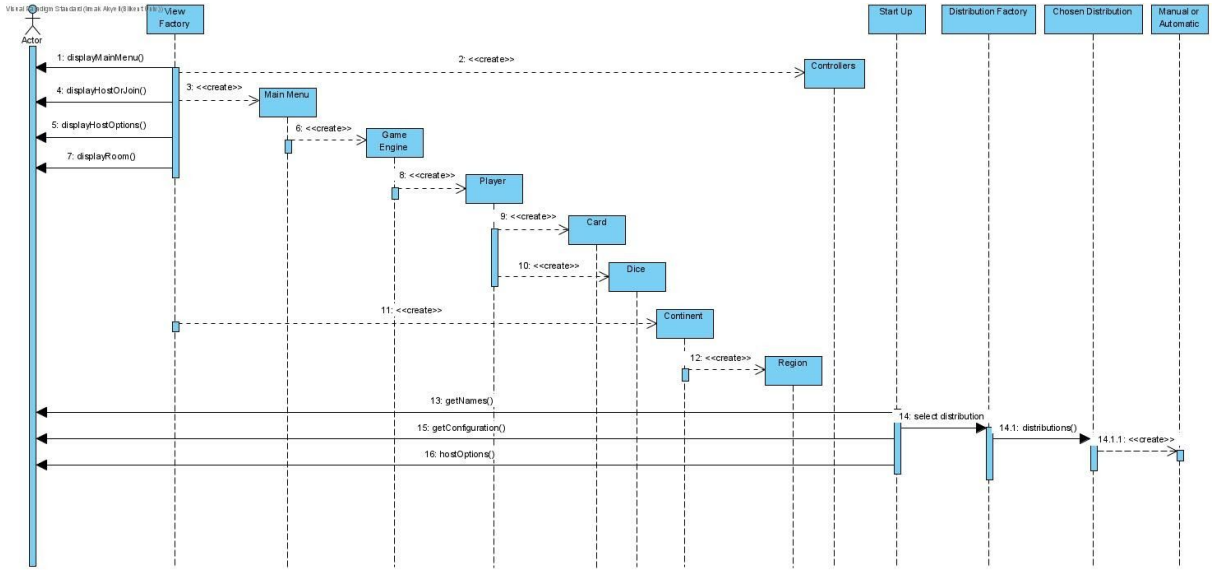


Figure 6: Sequence Diagram of the Start

In this sequence diagram we see the start of the game. The actor who is the player, chooses if he/she wants to be the host or join an already existing game. After that if the actor is the host he decides the number of players and starts the game. When the game is started a series of object creations occur. After all the object creations the game is ready to start and a special game part that lets the actors distribute their army and choose their initial lands occur that is controlled by the Distribution Factory. As there are two options of the distribution, a special pattern is used here. This part is not involved in the main game play as it is not in the main loop of the game and the mechanics are different. Therefore, it is a special state and is considered separately. It can be seen by the sequence diagram that there is no main Class that calls all the others to create a more simple game control. This simplifies the control of the operations and will enable simpler coding.

## 5.4.2 Sequence Diagram for the Game Play

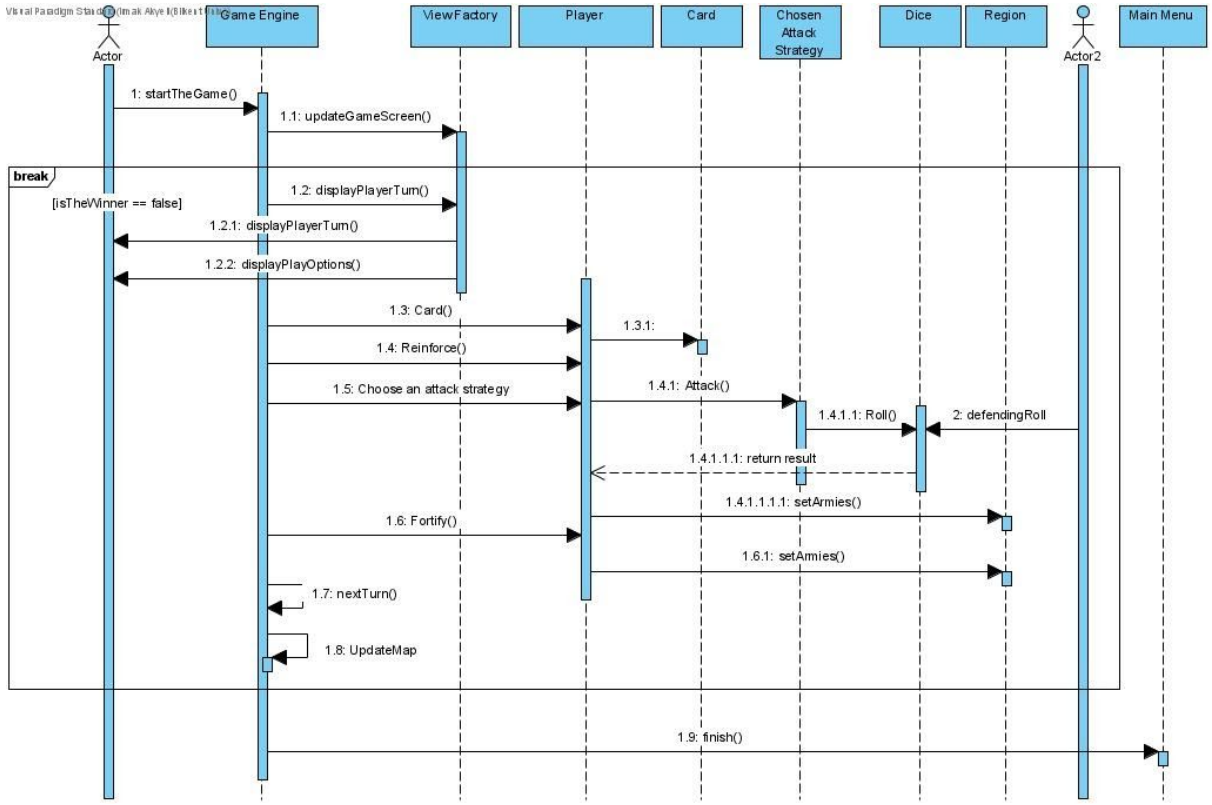


Figure 7: Sequence Diagram for Game Play

In this sequence diagram we see the actual game play. It starts after the game is started by the actor and the screen is updated. There is a main loop for the gameplay that controls the turns. The turn starts when the View Factory shows the actor which is the player of the game (but not to be confused with the player class and therefore the players are addressed as actors) who's turn it is. After that the play options are shown to the actor and depending on the actor, the game manager can check the cards, reinforce, attack or fortify during the actors turn. Attack can have several different types like Advantageous, Disadvantages and normal, therefore we see that there is a decision made to use the specific attack class. After each turn it is checked if the player has won and the variable isWinner updated. As long as a player is not the winner it remains false and therefore the game continues. Once all the actions of the actor are done, the map is updated and the turn is passed to the next player. As long as there is no winner the loop continues and the game carries on. When there is a winner the game exists in the loop and the main menu is called with the finish method.

## 5.5 User Interface

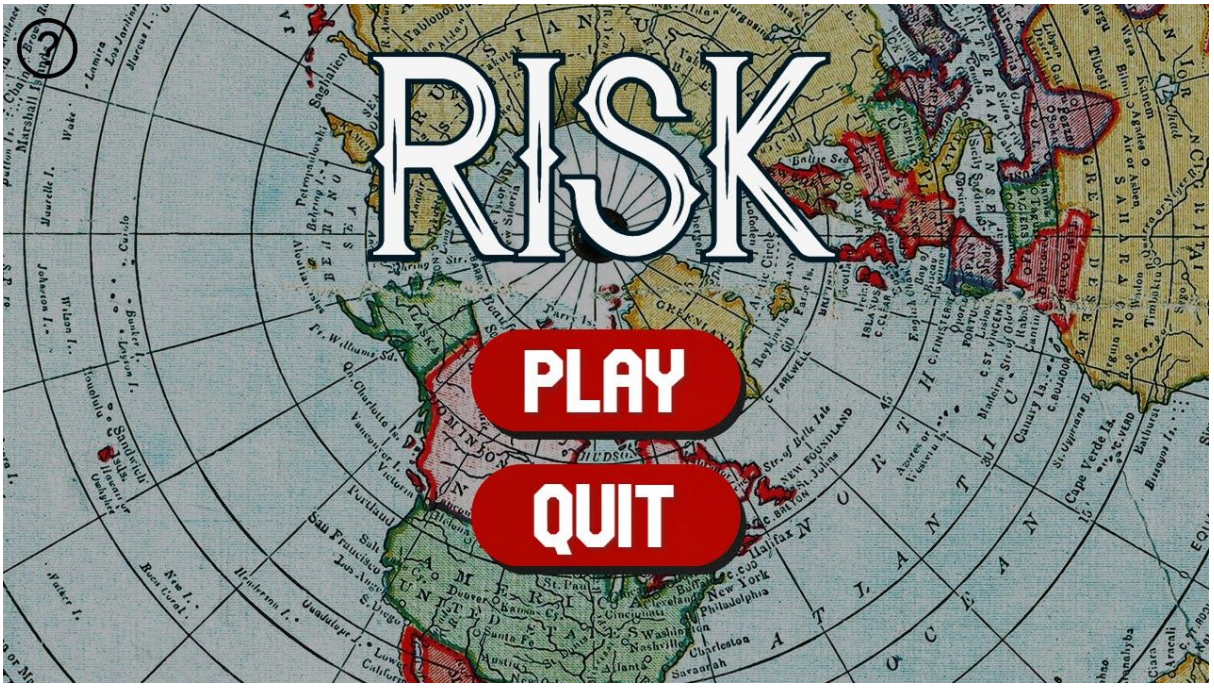


Figure 8: UI of the Main Menu Screen

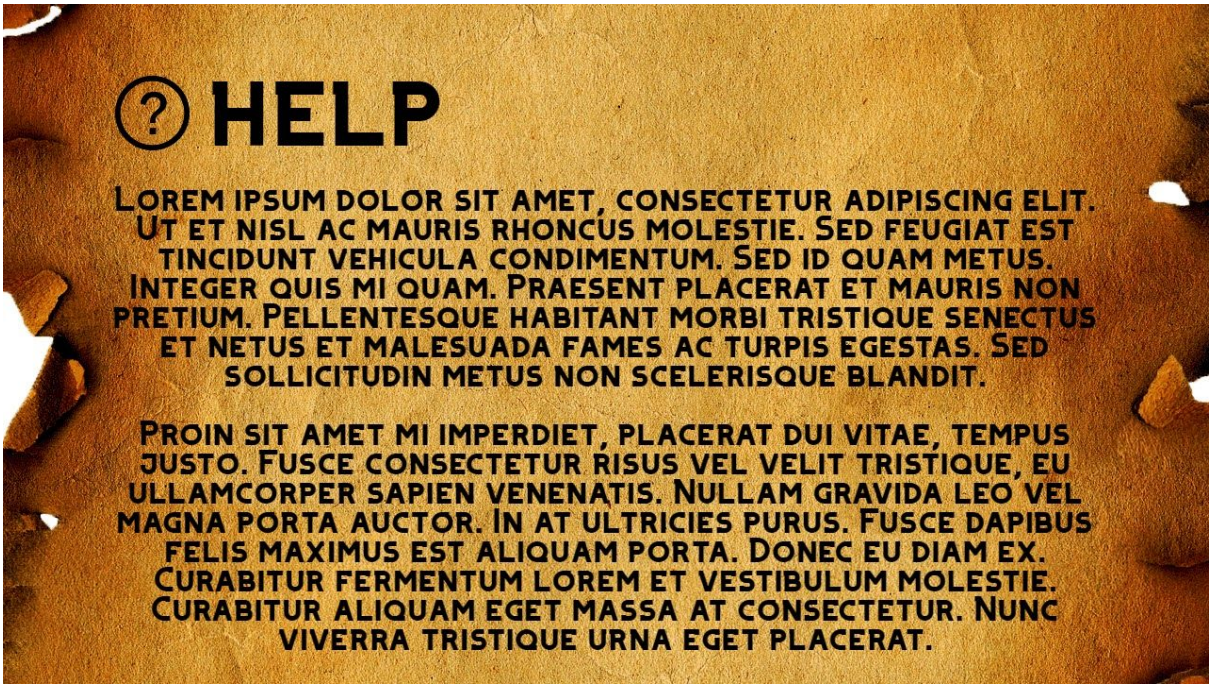


Figure 9: UI of the Help Screen


Figure 10: UI of the Host and Join Options Screen

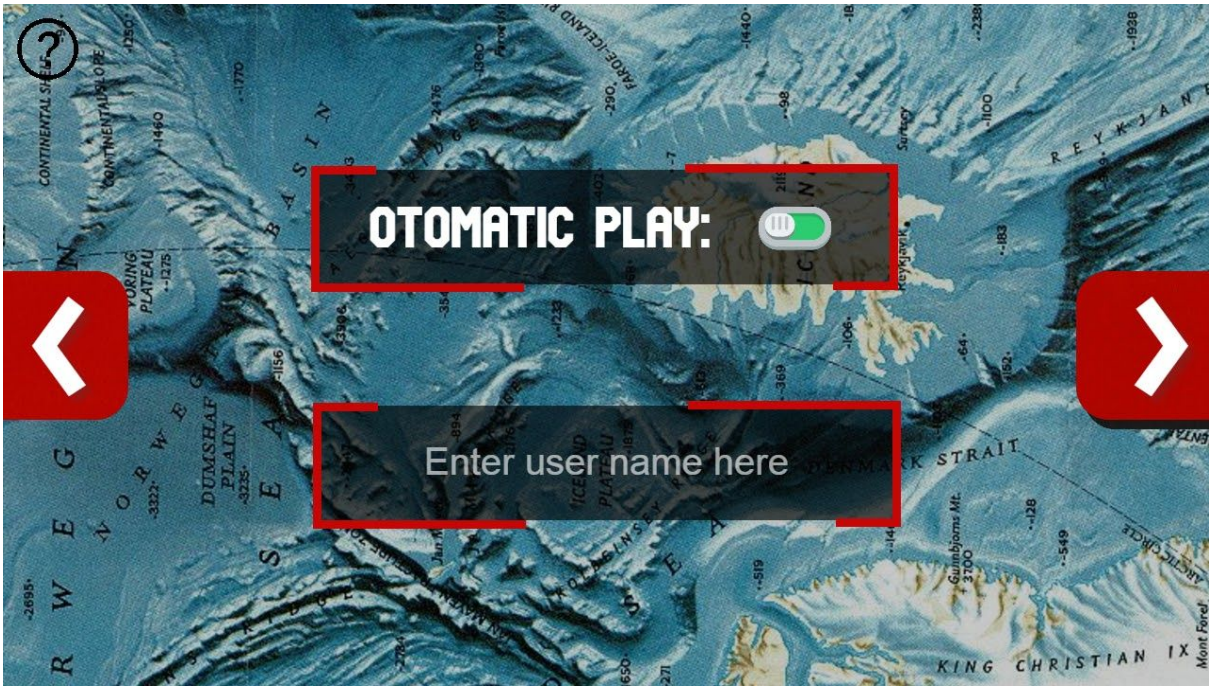


Figure 11: UI of the User Configurations Screen for Host


Figure 12: UI of the User Configurations Screen for Guests

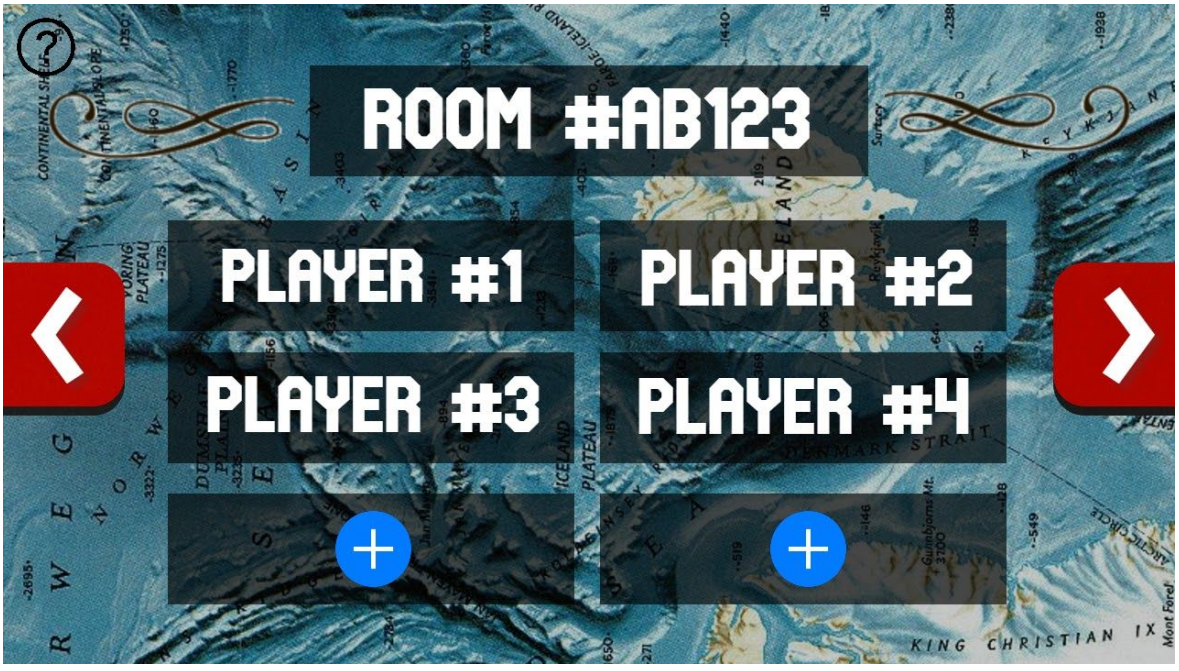


Figure 13: UI of the Game Room Screen


Figure 14: UI of the Player's Reinforcement and Game Setup Screen

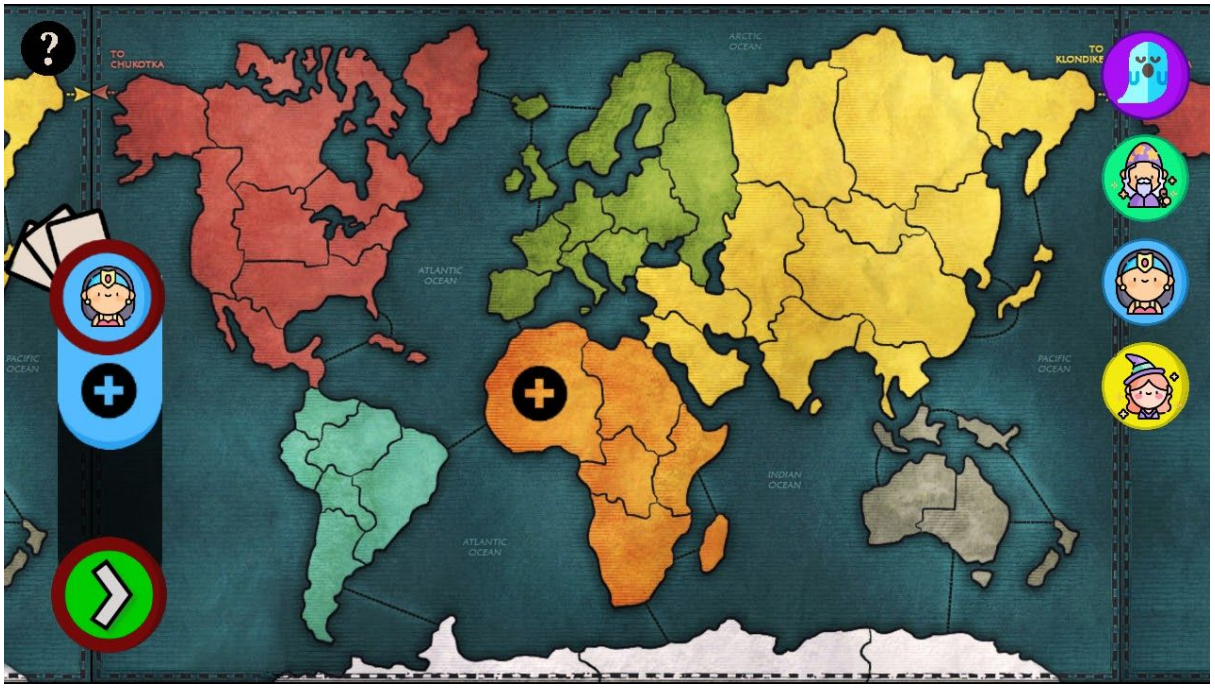


Figure 15: UI of the Player's Reinforcement and Game Setup Screen


Figure 16: UI of the Player's Reinforcement and Game Setup Screen

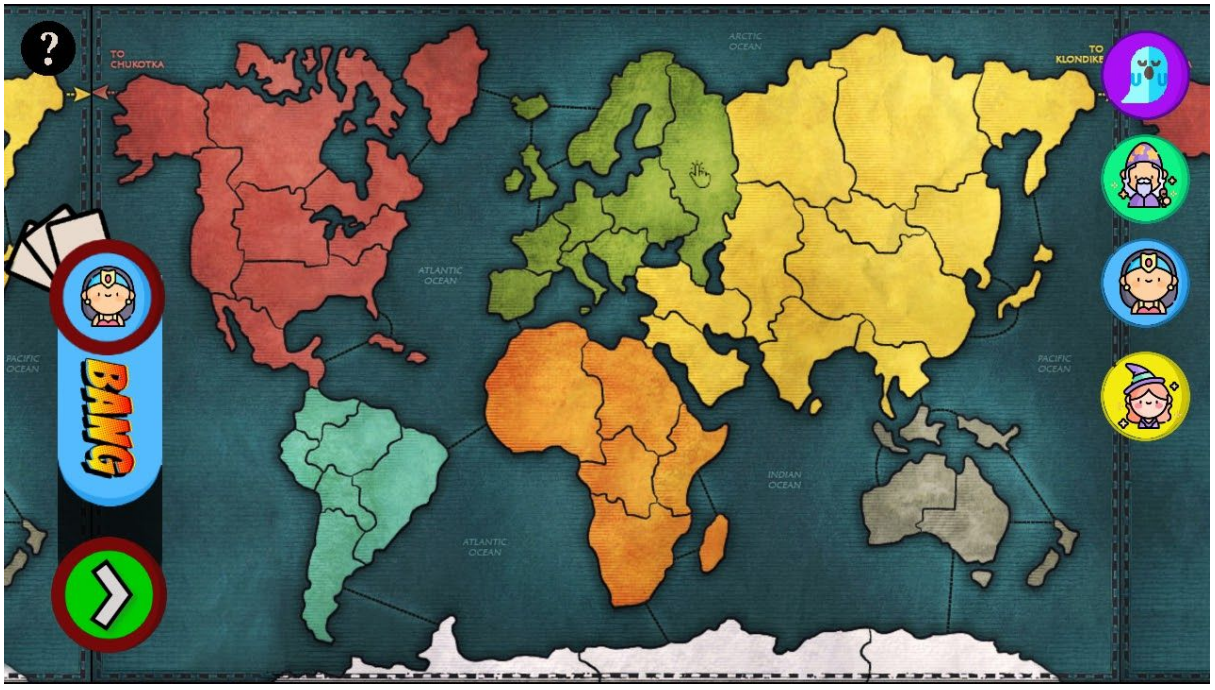


Figure 17: UI of the Player's Attack Screen | Selecting the attacker country

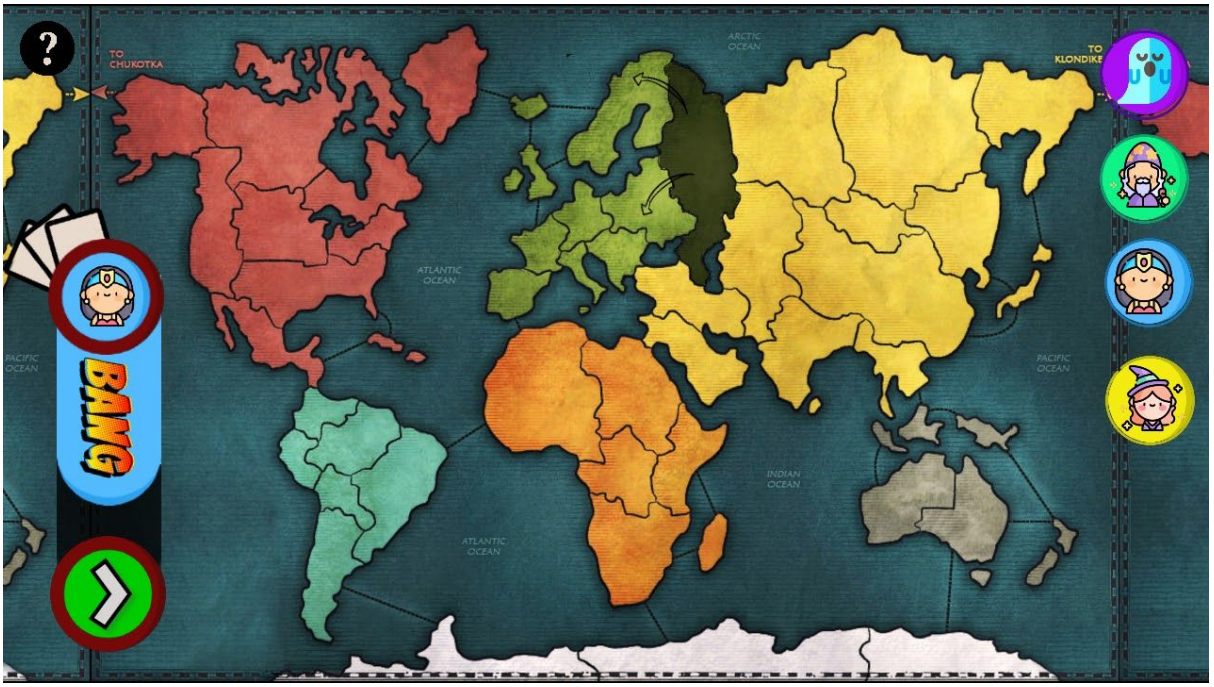


Figure 18: UI of the Player's Attack Screen | Attacker country is selected

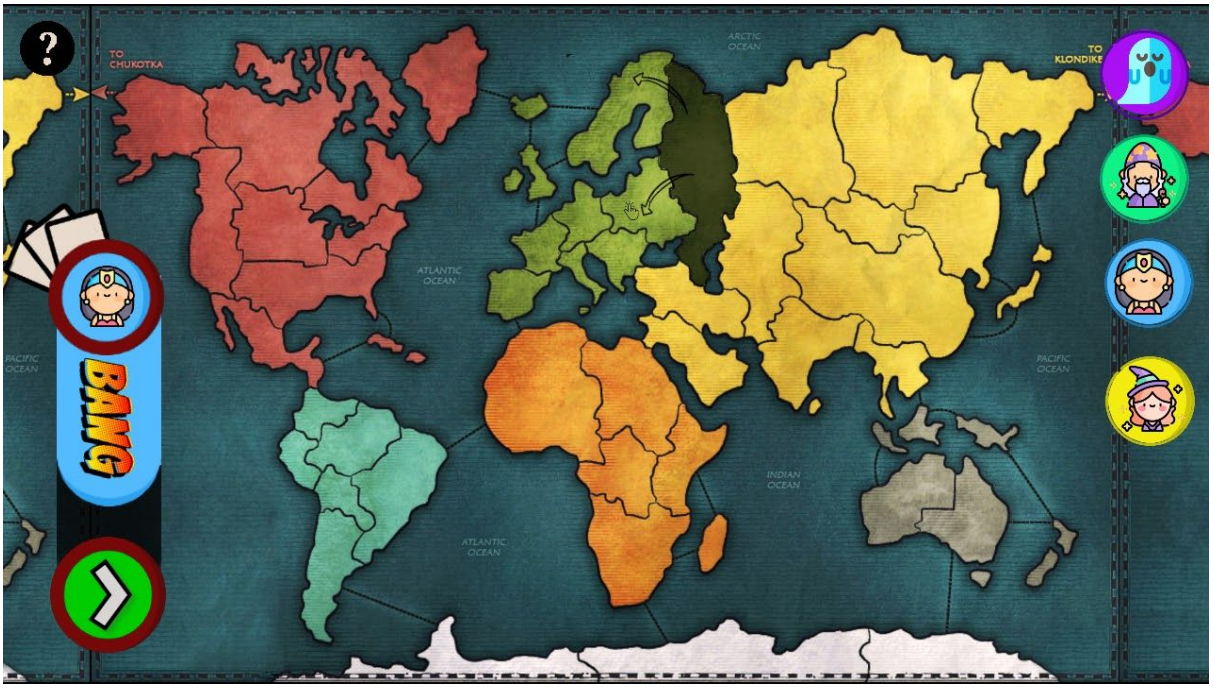


Figure 19: UI of the Player's Attack Screen | Attacked country selected

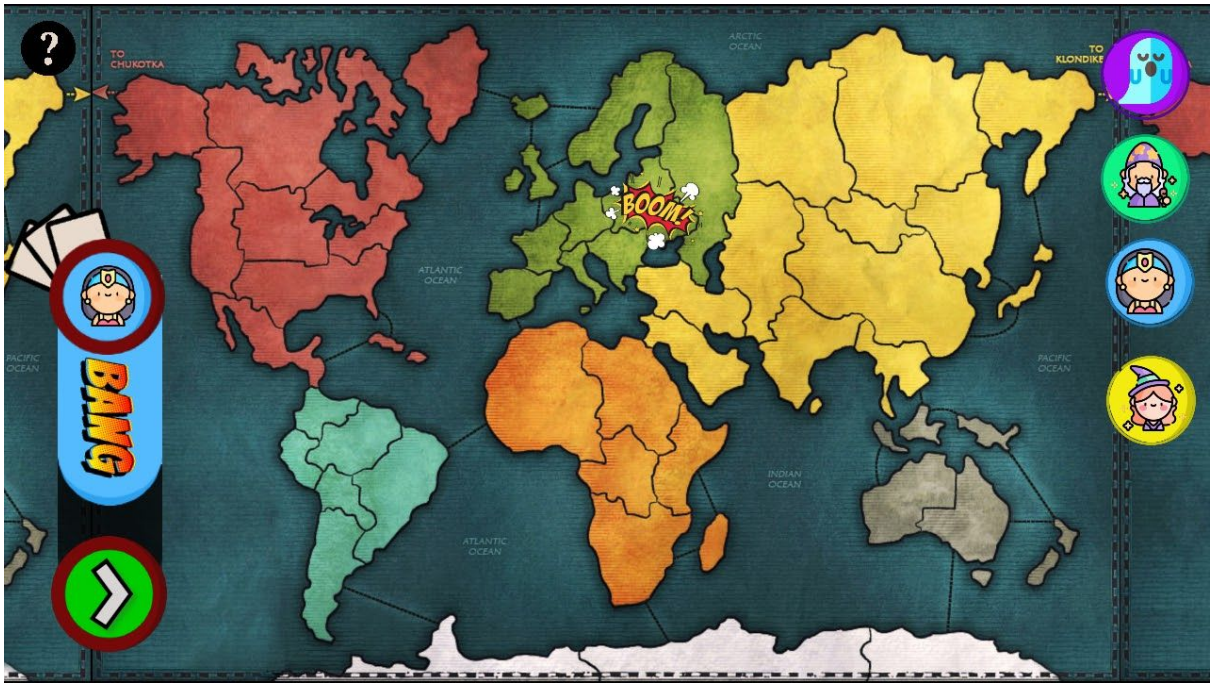


Figure 20: UI of the Player's Attack Screen | Attack

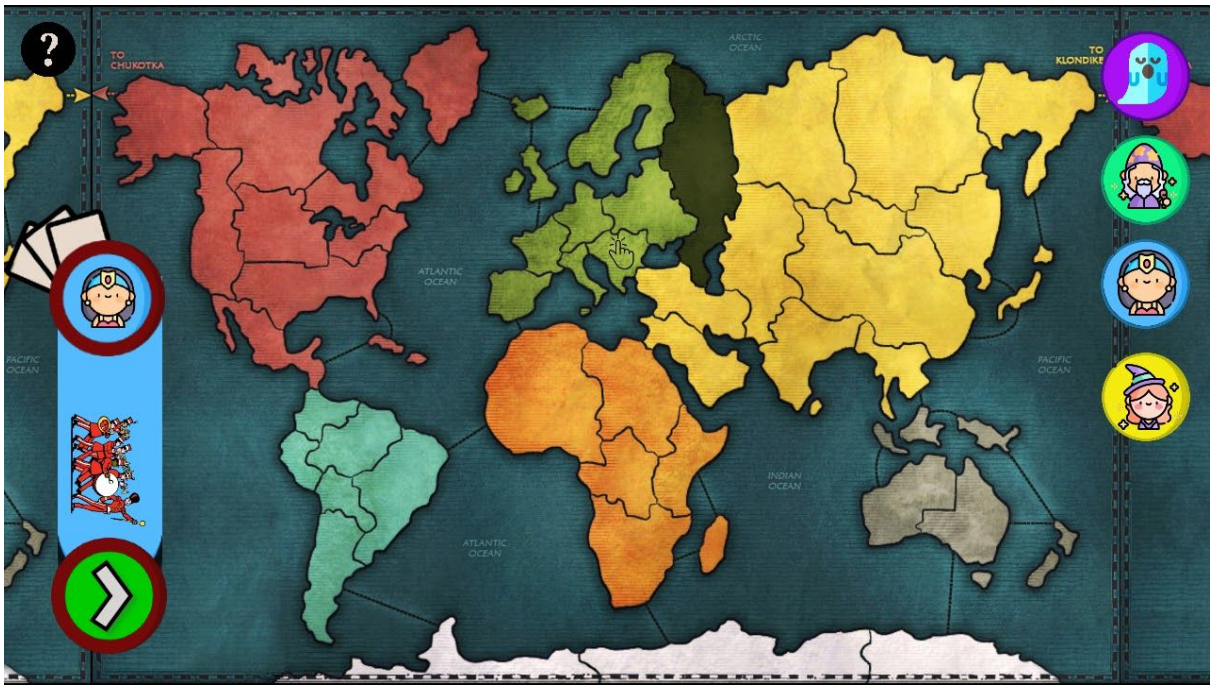


Figure 21: UI of the Player's Fortify Screen | Fortified country selecting

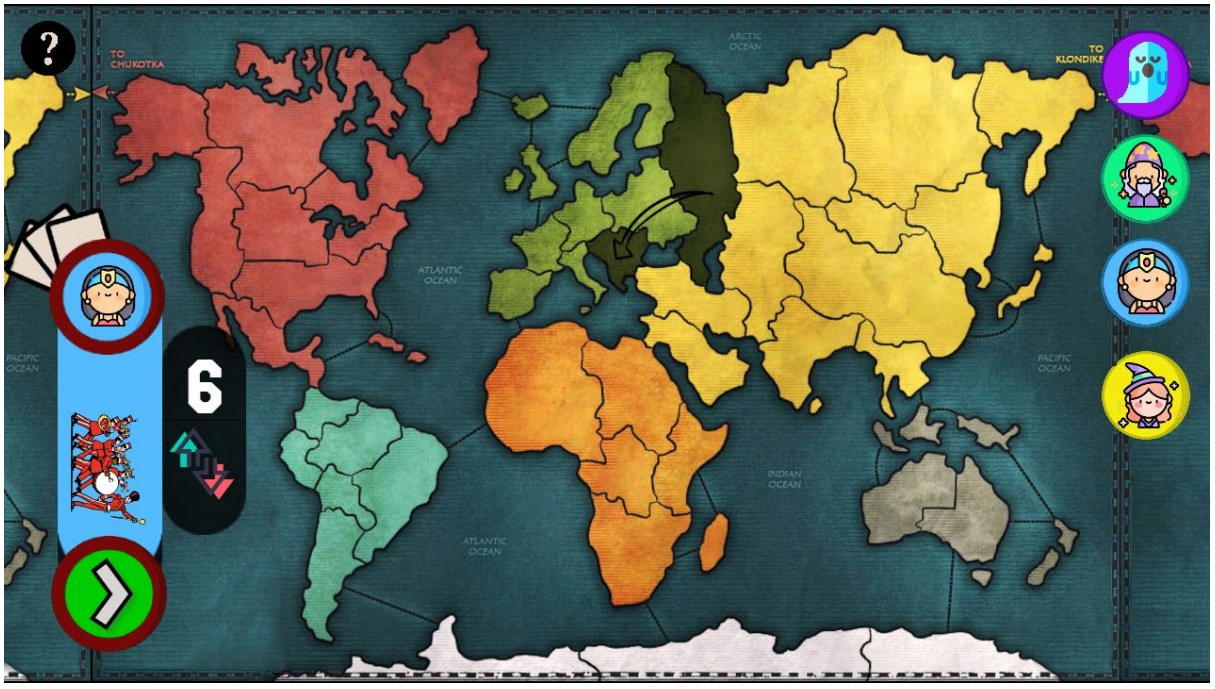


Figure 22: UI of the Player's Fortify Screen | Fortified soldier amount selecting

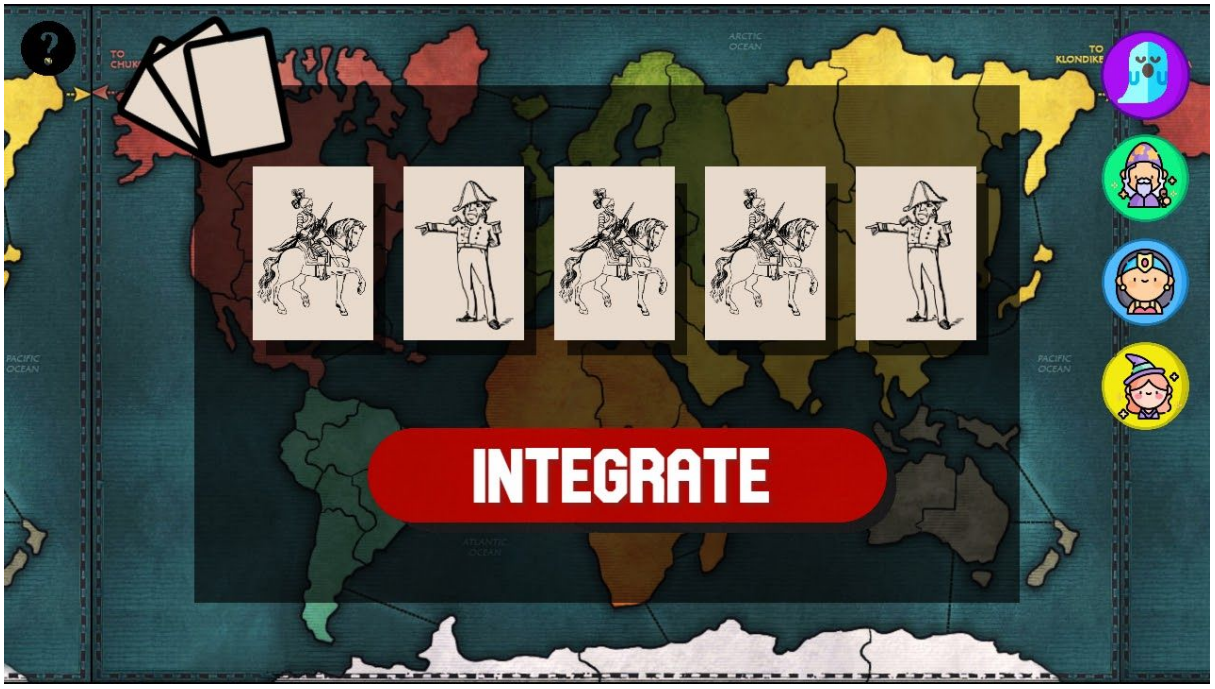


Figure 23: UI of the Player's Risk Cards Screen


Figure 24: UI of the Victory Screen Where the Winner is Shown

# Glossary and References

[1] Robinson, G., n.d. *The Strategy Of Risk*. [online] Web.mit.edu. Available at: <https://web.mit.edu/sp.268/www/risk.pdf> [Accessed 26 October 2020].

[2] Hasbro.com. 1993. *Risk: The World Conquest Game*. [online] Available at: <https://www.hasbro.com/common/instruct/risk.pdf> [Accessed 26 October 2020].

[3] Wallen, J., 2010. *10 Things That Make Software User-Friendly*. [online] TechRepublic. Available at: <https://www.techrepublic.com/blog/10-things/10-things-that-make-software-user-friendly/> [Accessed 7 December 2020].