CS 319 TERM PROJECT

Section 1 Group 1I

Risk Game Analysis Report

Emin Adem Buran - 21703279

Onur Oruç - 21702381

Ömer Yavuz Öztürk - 21803565

Melike Demirci - 21702346

Yusuf Ziya Özgül - 21703158

Supervisor: Eray Tüzün

Table of Contents

1. Introduction

Risk is a turn based board game. In the game, the purpose is to capture areas owned by other players and owning entire regions. The game can be played with two to six people. In the standard version, the entire world map is divided into 42 regions. When each player comes to his/her turn in the game, the player can make certain moves according to the rules of the game such as attacking or fortifying a region, drafting troops to a region. Also, players can form alliances throughout the game.

In this project, we will make a desktop version of this game developed by Hasbro and there were additional features in the game. We will explain these additional features in detail in the section overview. The game can be played by two or four players on a computer. In the game, players can have money and money can be gained through gold mines. Gold mines will exist in some random regions. Also, troops in the game will have motivation and the motivation of the troops will have effects on the results of wars. The motivation of troops can be increased by organizing events and these events can be organized by using money.

To summarize, our version of the game will be the digital version of the game designed by Hasbro with some additional features and modes that will be explained later in detail. With these additional features and modes, we aim to satisfy the player expectations.

2. Overview

2.1. Gameplay

The Risk game we implemented is a multiplayer game that can be played offline. In this game, each player has regions with troops located in them. The aim of this game is to conquer every region on the map. Each player takes their turn periodically. When it is their turn, players can draft troops, attack enemy regions and fortify troops respectfully. Players can attack a neighbouring region owned by another player during their own turn. Players conquer this region if their attack is successful. The success of their attack depends on how many troops the attacking player has and how many troops defending player has and the motivation level of soldiers and maybe most importantly, chance. During a battle, troops of attacking and defending sides roll dice for each soldier they drive to the battlefield (up to three at a time). Soldiers that get greater numbers defeat enemy soldiers and so decrease the number of enemies. The attackers can attack a region as many times as they wish until

their troops are down to one or they conquer the region by defeating all enemy soldiers. Numbers are of the most importance in a battle so players need to strengthen their critical regions in draft and fortify stages to increase their chances of winning.

2.2. Map

The map of the game is similar to the map of the world and the regions are similar to the countries of the world. In the map regions are colored to show which player they belong to. Colors of the regions can change dynamically depending on the results of attacks. Dice are shown on the map when attacking occurs.

2.3. Players' Initial Items

Players initially have regions, troops, a war hero and some gold. Before the game starts, regions are distributed to players randomly and troops are distributed randomly to regions. Each player has the same amount of troops but distribution of troops to regions is a random process.

2.4. Modes

The game we designed stands out with its varying in-game modes. Activations of modes are independent from each other so the users are able to activate or inactivate whatever mode they want. Modes are named "Pestilence", "Heat and Snow", and "Clash of War Heroes".

2.4.1. Pestilence

In this mode, there is a deadly and extremely viral plague which might emerge from nowhere when the condition is met. The chance of emergence in each region increases according to the number of soldiers in that region because of hygiene problems. How the chance will be calculated is not yet determined due to the balance issues. When a region hosts the plague, each soldier in that region either suffers from the plague, or has a chance to suffer from it next turn. Soldiers that suffer from the plague are unable to fight or move for two turns after which they will have a chance to survive and return back to normal, otherwise they will die. If a soldier that has a chance to suffer from the plague moves to a different region and then suffers the plague in that region, that region will also host the plague. If a region that hosts the plague gets attacked and defeated by the enemy, which means only soldiers with the plague are left in the region, enemies will remove the

plague from the region by burning down the city and then the region will be conquered by them.

2.4.2. Heat and Snow

In this mode, weather conditions get in the way of armies. There will be different seasons and climates to determine the weather condition. Seasons will be the same for the whole map at a time but do change periodically in an order which goes as Spring, Summer, Fall, Winter. They are thought to be changed every two full-turns. On the other hand, climates do not change but they might differ from one region to another. To make things simple, there are three climates which are Hot, Warm and Cold. The combination of climate and season at a time determines the weather condition of a region. According to the weather condition, a disastrous event such as “Drought” or “Frost” might take place in a region. Both of these events make it hard to move soldiers.

2.4.2.1. Drought

When this event occurs in a region, supplies run short and soldiers are unable to be fed properly. Therefore travelling from this region in large groups becomes impossible. Only a limited amount of soldiers can leave this region every turn while the effect is active. To allow more flexible strategies, there might be an option to move more soldiers with the cost of remaining soldiers’ lives. The limit is not yet determined due to the balance issues. Also rather than putting a limit, the amount of usable supplies might be given to the players so they can do its math themselves.

2.4.2.2. Frost

When this event occurs in a region, a biting frost devastates the soldiers who travel. A proportion of soldiers will die travelling if they are moving to or from this region. The proportion is not yet determined due to the balance issues.

2.4.3. Clash of War Heroes

In this mode, powerful and charismatic commanders, who managed to construct strong bonds with their subordinates, come to the battlefield. Each player will have one war hero and they will be able to move their heroes just like their soldiers. During battles, war heroes will increase the motivation level of nearby soldiers. Increased motivation will make that side advantageous at battle but it is not yet determined how exactly. It would be

made in a way that if motivation is high, dice will never give the lowest value (1). Also heroes will be immune to all negative effects such as Frost, Drought or Plague. If a hero is defeated in a battle he will be born again in a random region that belongs to the player and will be available again.

2.5. Gold and Gold Mines

Gold can be used to buy mercenaries which will strengthen one's armies or it can be used to organize Festivals. Festivals increase the motivation level of the troops in the same region. Effects of the increased motivation level is explained under the Clash of War Heroes section (2.4.3). Players will have some gold at the start of the game and they will be able to retrieve more using gold mines.

Gold mines will appear in random regions at random times. At the end of every full-turn, players will gain an amount of gold for every gold mine they have in their regions.

2.6. Troop Cards

There are four types of troop cards which are Infantry, Marines, Artillery, and Mercenary. At the start of their turns, players can trade their troop cards for additional troops. To trade troop cards, players must collect at least three cards of the same kind or one from each kind. At the end of their turns, players will gain cards according to the regions they conquered that turn. If they haven't conquered any region, they gain one card anyway.

3. Functional Requirements

3.1 Game Play

- System should inform Players about the number of troops, climate conditions, existing mines in regions.
- System should manage the number of troops to be given in every turn of players.
- Players can select their regions one by one and draft troops to these regions.
- Players can choose the number of troops to be drafted to their regions.
- Players can select one of their regions to attack neighboring regions in their turn.
- System should adjust the view of selected region's neighbors that are available to attack.
- System should display the dice and attack results on the game map.

- System should update the color of regions on the game map according to the result of an attack.
- System should manage the location of the gold mines and climate changes in the regions.
- Players can buy mercenaries with their gold in their turn.
- Players can move their commanders to a selected region in their turn.
- Players can combine their troop cards in order to get more troops in their next turn.
- System should adjust the view of the game map according to the turn of players.
- System should display and update the number of troops, number of regions and amount of gold each player has.
- System should inform Players when the game has ended and display the scores.

3.2 Pause Game

- Players can pause the game in their turn.
- System should pause the game and display the pause menu on the screen.
- Players can press the buttons on the menu screen for different purposes such as displaying the rules and changing the settings.
- Players can press the continue game button in order to come back to the game.
- System should display the game map after the player presses the continue game button.

3.3 Settings

- System should display the volume level bar and mute option on the settings menu.
- Players can adjust the music volume by sliding the bar.
- Players can mute the music.

3.4 Display Rules

- System should display the rules of the game to the player.
- Players can learn about the rules of the game by pressing the display rules button from either the main menu or pause menu.

3.5 Load Game

- System should display the games saved before to players.
- Players can continue to saved games by load game option

3.6 Credits

- System should display the contributors of this game.
- Players can learn the ones who contributed this game in the credits part.

3.7 Save Game

- System should display a menu for saving the current game.
- Players can save the current game and they can continue playing it later.

4. Nonfunctional Requirements

4.1. Usability

- The game should be designed in a way that all age groups will be able to play the game easily. The simulations (e.g. dice rolling) for the actions taken in the game will be attractive for children, the design will be easy for old people to understand (e.g. user interface guidelines on color schemes, logos, and fonts will be followed).
- The players should interact with the game by using a mouse.
- When a player wants to play the Risk Game for the first time, he/she should be able to learn the rules of the game from the Rules screen in the Main Menu. Thus, users will be able to join a game without playing a training mode at the beginning of the game.

4.2. Reliability

- The game should not crash more than once during a full game. To handle this, the game will be tested with different test cases, and the reasons behind the crashes will be eliminated.
- The game should restart within 30 seconds after a failure.
- To make the Risk Game function correctly, possible human related errors will be considered when the design is implemented. For instance, when a player clicks the exit button accidentally, the player will be asked whether s/he is sure about the

decision. To give another example, when a player quits the game before saving it, s/he will again be asked whether s/he is aware that the progress in the game will be lost.

- All buttons in the game should not be available to a player in some cases. For example, for players who do not have sufficient money, the button for hiring mercenaries will not be active. Thus, the input errors stemming from players will be reduced.
- When the power is cut, the progress in the game should be saved automatically by the system. When the game is opened again, the game will continue from where it stopped.

4.3. Performance

- The map initialization and random soldier distribution into random regions should be ready in 5 seconds not to lose players' attention.
- The simulations occurring after an action is taken in the game should start to be displayed within a second following the action so that players will not spend time waiting for the simulation to occur.
- The calculations behind the gameplay such as random soldier distribution into random regions, checking whether a player has whole regions in a continent should be implemented to give results in at most 3 seconds.
- The system should support at least 4 players to play a game.

4.4. Supportability

- The game should be implemented in a way that it accepts additional features (e.g. new maps) without making modifications to the existing game. To satisfy this expectation, all classes in the game will be separated under the categories of boundary, entity and controller.

5. System Models

5.1. Use Case Model

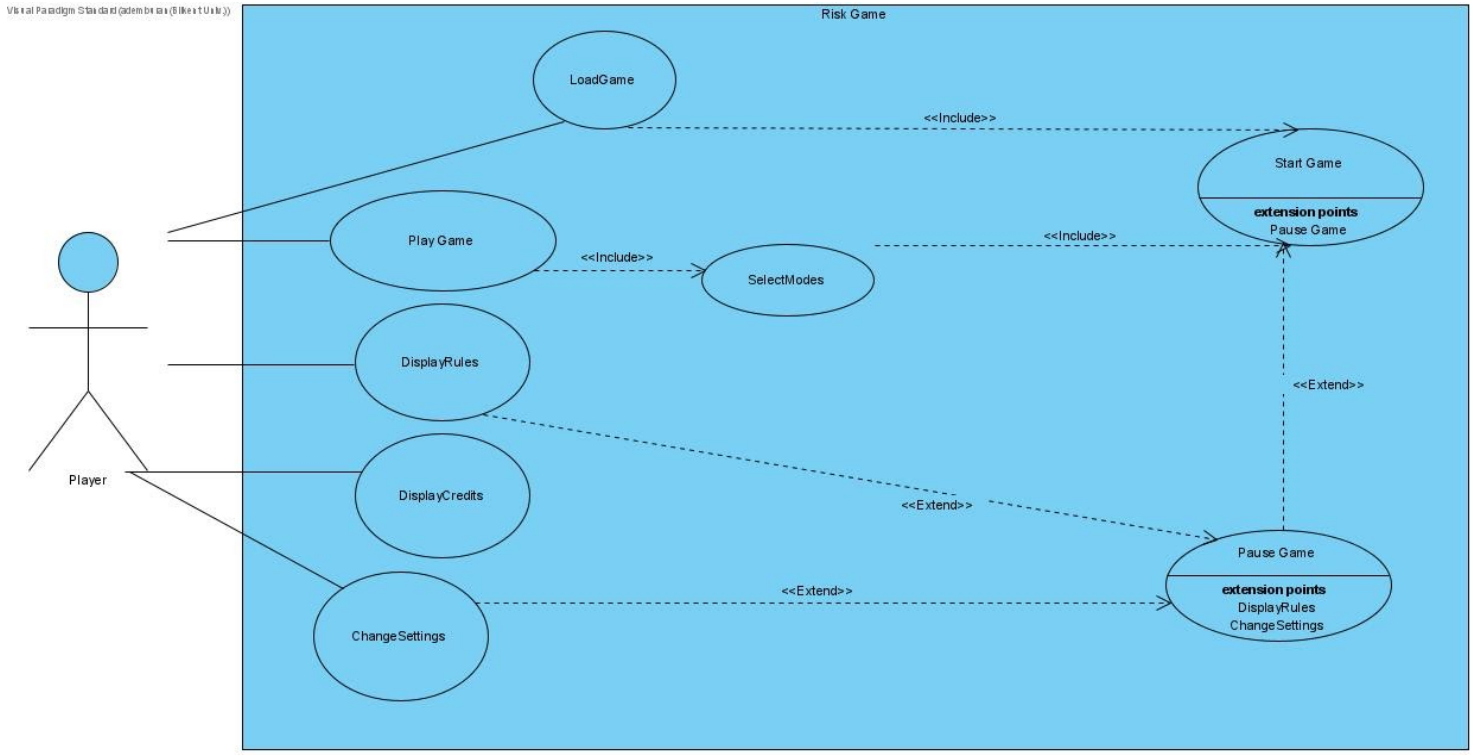

Figure 1: Sketch of the use case diagram of the Risk Game

Use Case 1: PlayGame

Participating Actor: Player

Entry Conditions:

- Player opens the Risk Game.
- Player is on the main menu of the game.

Exit Condition:

- Player was directed to the select modes to start a new game.

Flow of Events:

1. Player sees the main menu.
2. Player clicks on the Play Game button on the main menu.
3. Player is directed to Game Options to set the modes of the game.

Special / Quality Requirements:

- It should not take more than 3 seconds to direct Player to the SelectModes use case.

Use Case 2: LoadGame

Participating Actor: Player

Entry Condition:

- Player wants to play Risk Game with his/her friends.
- Player enters the Risk Game.

Exit Condition:

- The game Player selected is ready to be loaded.

Flow of Events:

- Player clicks the Load Game button.
- Player sees saved games.
- Player selects a saved game.
- Player clicks the Load button.
- The saved game is loaded by the system.

Special/quality Requirements:

- It should not take more than 5 seconds for a saved game to be loaded.

Use Case 3: SelectModes

Participating Actor: Player

Entry Condition:

- Player chooses the play game option.

Exit Condition:

- Player has selected the game modes.

Flow of Events:

1. Player marks the modes s/he wants in the game.
2. Player clicks the Start Game button or the Back button.
3. If the Start Game button is clicked, the game starts. If the Back button is clicked, Player is directed to the main menu.

Special/quality Requirements:

- Selected modes are initialized within 5 seconds.

Use Case 4: StartGame

Participating Actor: Player

Entry Condition:

- All players are ready to start the game.
- The map initialization and soldier distributions are done.

Exit Condition:

- Player won the game.
- Player quit the game.

Flow of Events:

1. Player clicks the Start Game Button.
2. Map initialization and solder distribution are done.
3. Player in turn hires a mercenary and organizes events according to his/her need.
4. Player drafts his/her troops and commander to her/his regions.
5. Each player attacks neighboring areas by using his/her soldiers.
6. Player fortifies his/her regions or his/her allies' regions by using troops.
7. The turn of the player changes.
8. The other players in turns do steps 3-7 until the step 9 is satisfied.
9. Player who does not have any region leaves the game.
10. The game ends when a player has all regions in the game.

Special/quality Requirements:

- The simulations created after the actions taken by Player in the game should not take more than a second to be displayed.
- The game should restart within 30 seconds after a crash.

Use Case 5: DisplayCredits

Participating Actor: Player

Entry Condition:

- Player opens the game and sees the main menu.

Exit Condition:

- Player has information about the game developers.

Flow of Events:

1. Player clicks the Credits button on the main menu.
2. Player sees the contributors of the game.
3. After all credits have been displayed, Player is directed to the main menu.

Special/quality Requirements:

- Since there are no buttons in DisplayCredits, no crashes that force players to take action should happen.

Use Case 6: DisplayRules

Participating Actor: Player

Entry Condition:

- Player opens the game and sees the main menu.
- Player starts to play the Risk game.

Exit Condition:

- Player has information about how to play the game.

Flow of Events:

1. Player clicks the Rules button on the main menu or on the game screen.
2. Player sees the game rules.
3. Player clicks the X button on the top right.
4. Player leaves the Rules page.

Special/quality Requirements:

- Player should be able to understand the game rules in DisplayRules without any need to look at external resources.

Use Case 7: ChangeSettings

Participating Actor: Player

Entry Condition:

- Player opens the game and sees the main menu or Player starts to play a Risk game.

Exit Condition:

- Player changed the game settings.

Flow of Events:

1. Player clicks the Settings button on the main menu or on the game screen.
2. Player adjusts the music volume or mute the music.
3. Player clicks the X button on the top right.
4. Player is directed to the main menu (if Player has not started a game yet) or the game screen (if Player changed the settings during a game).

Special/quality Requirements:

- Changes in the game settings should be adjusted without resulting in any crash in the game.

5.2. Dynamic Models

5.2.1. Sequence Diagram

5.2.1.1. Starting a New Game

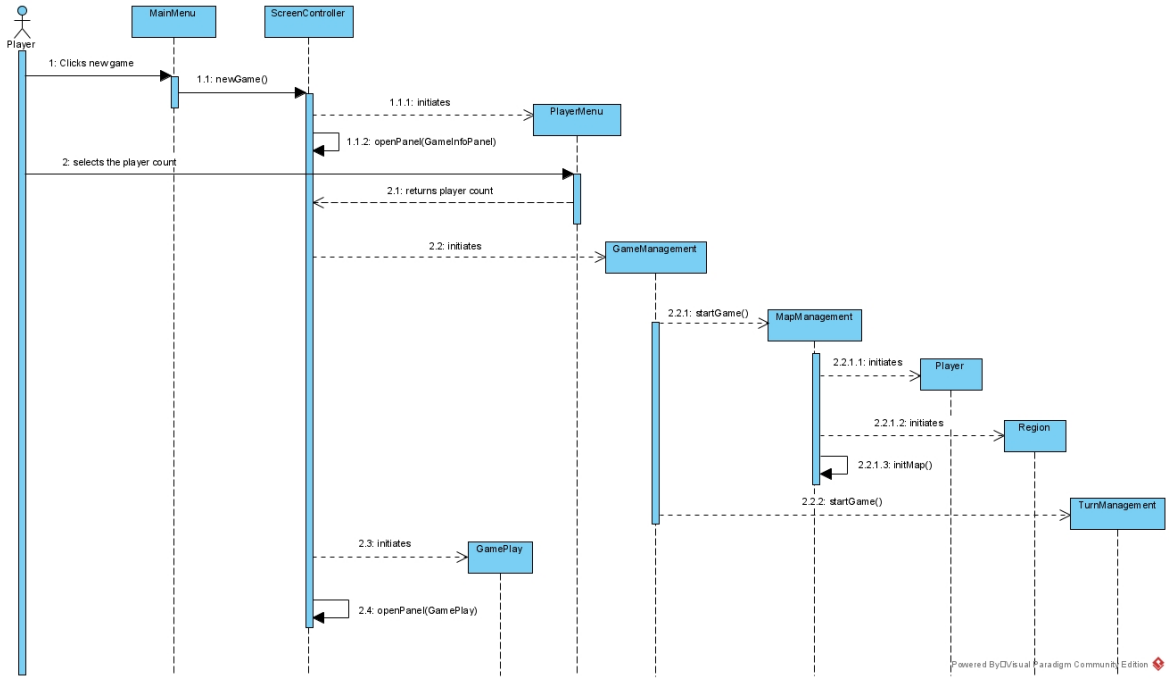

Figure 2: Sequence diagram illustrating the actions to start a new game

Scenario: The player clicks the new game button on the GamePlay panel. Clicks the New Game button. After that, ScreenController initiates PlayerMenu class which is another panel to get game information such as player count, nicknames, game mode and map. ScreenController draws this panel on the screen with updateGUI() method and waits for the user input. The player should provide the game information through the PlayerMenu panel. Then, ScreenController initiates the GameManagament object with provided information. Later on, GameManagement initiates MapManagement and TurnManagement classes with startGame(). MapManagement, which is initiated by the GameManagement, initiates the players and regions then distributes regions to players.

5.2.1.2. Draft Troop

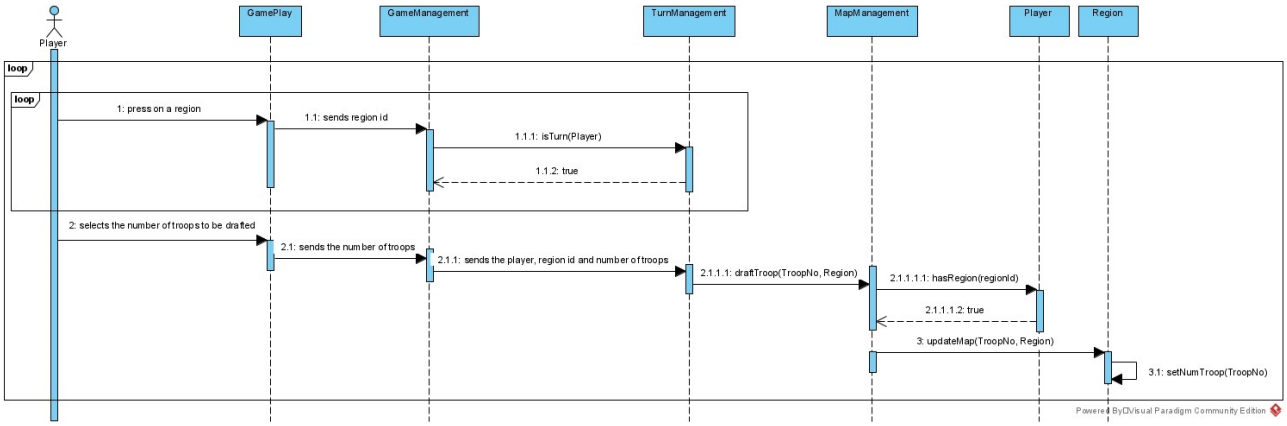

Figure 3: Sequence diagram illustrating the player actions to draft soldiers

Scenario: The player presses on a region in the GamePlay panel. The GamePlay panel sends the id of the pressed region. After that, GameManagement checks if it is the turn of the player. Player can choose different regions more than once. Same sequence occurs for every region selection. After selecting a region, player selects the number of troops to be drafted. GamePlay sends this information to the GameManagement. Thereafter, GameManagement sends the player, region id and number of troops to be drafted to the TurnManagement. TurnManagement drafts the troop according to the given information with the help of MapManagement. MapManagement performs the draft by checking if the player has a given region. If player owns the region, MapManagement updates the number of troops in the region.

5.2.2. Activity Diagram

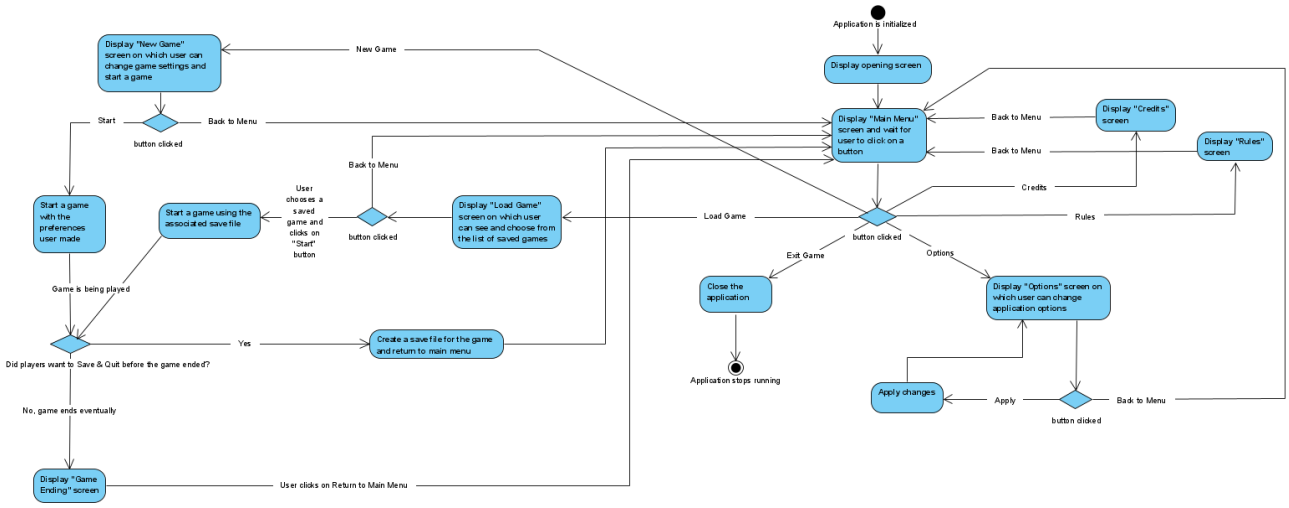

Figure 4: Activity diagram for Game Menu

This diagram illustrates the flow of activities in our application. User is led to the main menu on which he will see a total of six buttons. If user decides to click on the New Game button, he will be able to create and play a game. User is also able to save and quit before the game ends, to play it later by using the Load Game option in the main menu. If the game ends or user saves and quits the game, user is led back to the main menu. On the main menu, user can also decide to view and change the application options by using the Options button. Changes will be applied only if user clicks on the Apply button. Other than these, user can choose to view game rules or credits by using associated buttons on the main menu. Finally, user can use the Exit Game button to close the application.

5.2.3. State Diagram

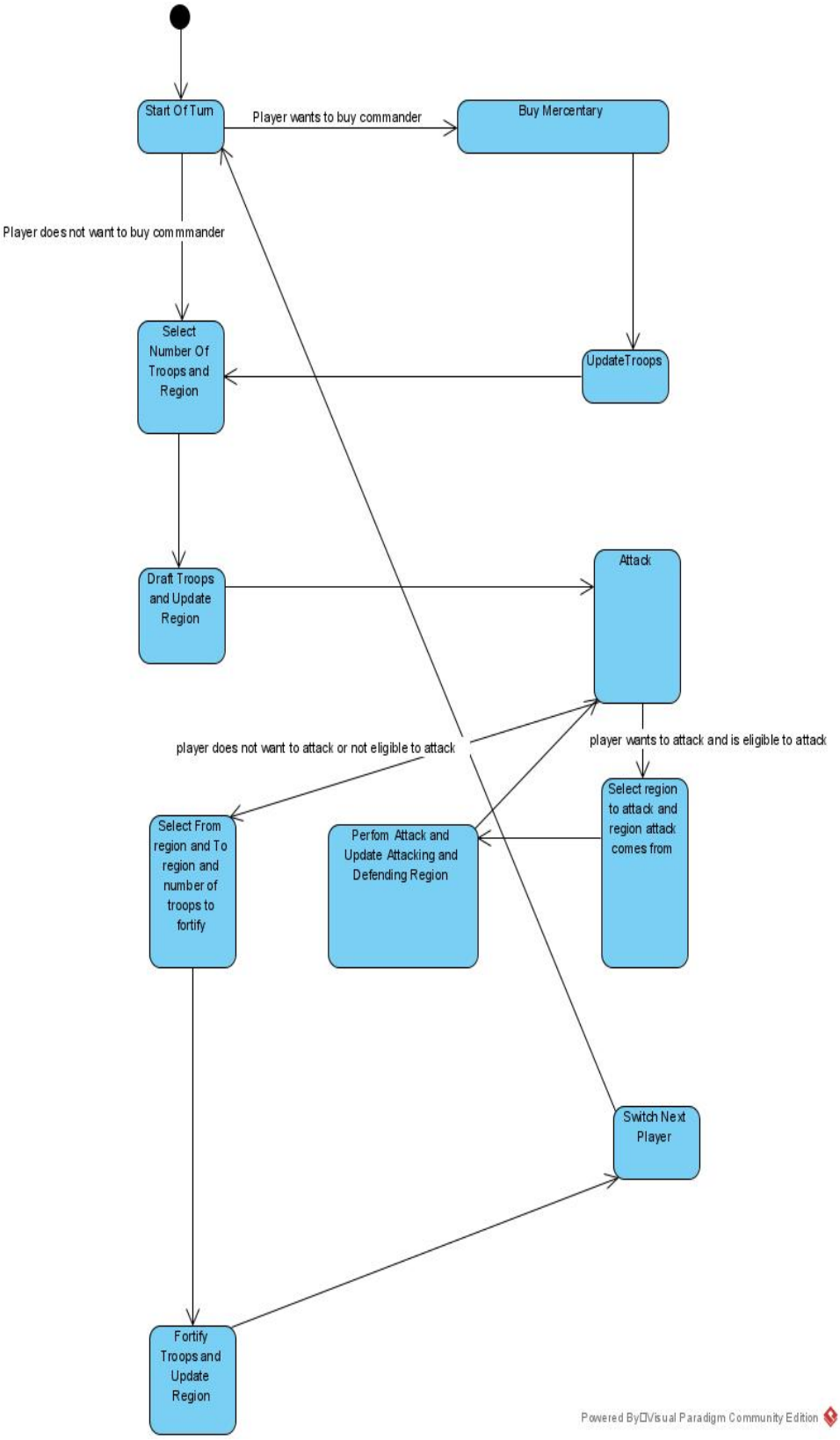

Figure 5: State diagram for turn of a player

The state diagram given above shows how the turn of one player processes. In the beginning, player selects whether to buy a commander(s) or not. Player has a chance to not buy the commander. Then, player draft troops to a region he or she is owning. After drafting is completed, player goes to the attack stage. Player can do nothing and can directly go to the fortify stage. If player wants to attack, player selects the region to attack and selects the region to select the troops used for attacking. As long as player is eligible to attack, player can attack. There is no constraint on how many times attacking. It only depends on whether player can attack or not. The last stage is the fortify stage. Player can fortify soldier(s) from one region to another region. Then, turn of the player ends and turn switches to the next player.

5.3. Object and Class Model

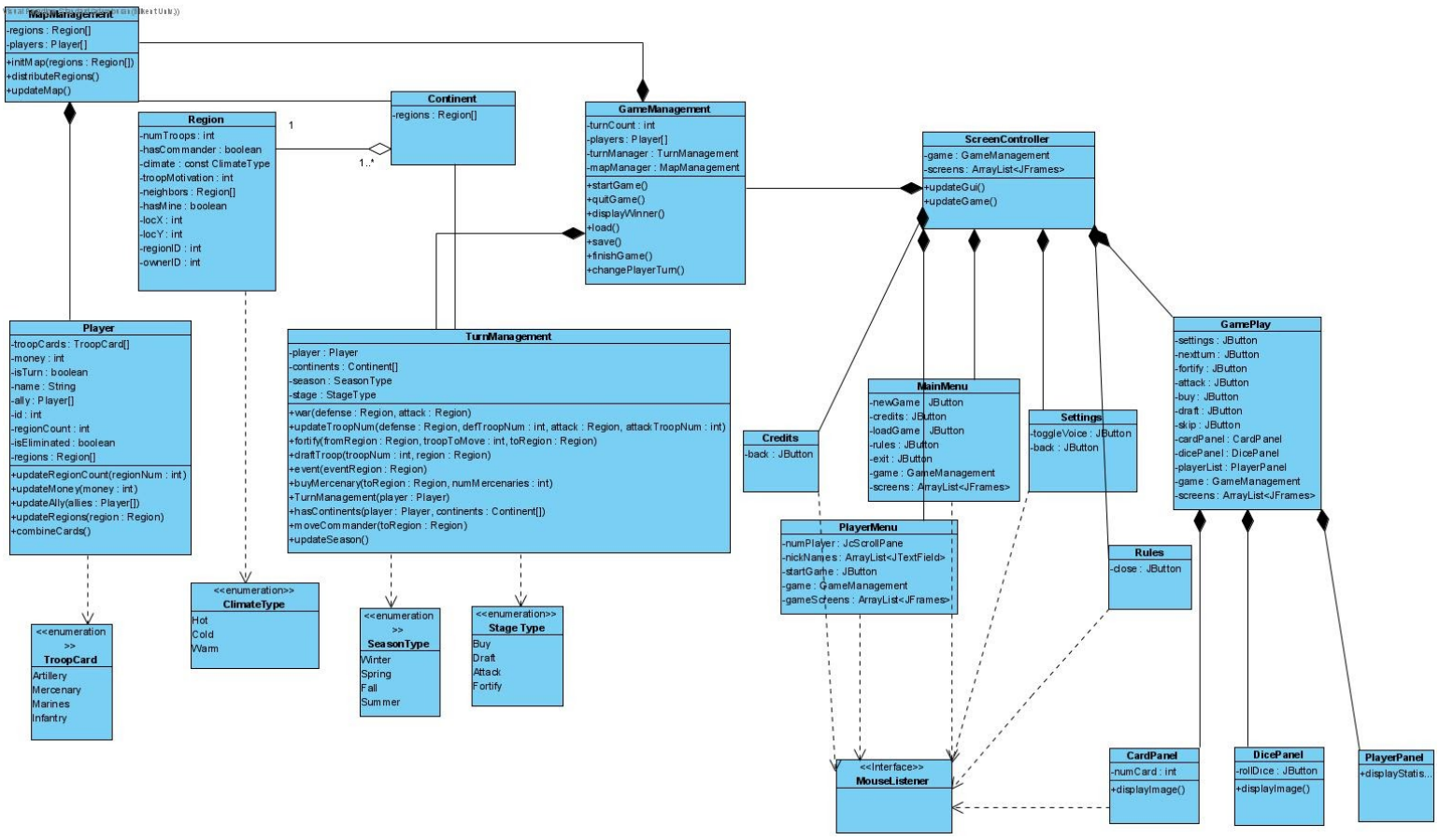

Figure 6: The sketch of the object and class models

❖ **Controller Class**

ScreenController: This class controls the entire Risk Game. Thanks to this class, boundary classes and entity classes interact with each other. This class initializes a GameManagement object and necessary boundary objects. Then, ScreenController updates the GameManagement objects according to the request coming from users through Boundary Classes.

❖ **Boundary Classes**

MainMenu: This class will provide a main menu screen view for the Risk Game. According to the player options, this class will update the GameManagement object. Players can click the New Game button, or Load Game button, or Exit Game button, or Credits button, or Setting button, or How to Play button on this main menu view.

Credits: This class will provide a credits screen view for the Risk Game. Players can click the Back button to return the main menu view.

PlayerMenu: This class will provide a game initialization screen view for the Risk Game. Players will select game modes, their avatars and their names in this view. Then the player will click the Start Game button to start a Risk Game.

Settings: This class will provide an audio setting screen view for the Risk Game. Players can click the Back button to return the main menu.

Rules: This class will provide a how to play screen view for the Risk Game. Players can click the Back button to return the main menu.

GamePlay: This class will provide a main game play screen view for the Risk Game. Players will play the game on this screen. From this screen, players can save their game, return to the main menu. Also, Game Play class will have some panels such as Card Panel,

Dice Panel, Player Panel. These panels will be set visible when a player wants to roll dice, look at the leadership table and their cards.

CardPanel: This class will provide a card panel view for users. This panel can be opened and closed when players want it on the Game Play screen.

DicePanel: This class will provide a dice panel view for users. This panel can be opened when players want to attach some region.

PlayerPanel: This class will provide a leadership table for users. This panel shows the situation of each player in the Risk Game.

❖ **Entity Classes**

Player: A player who plays the Risk Game will be defined by this class. Player class will hold the troop cards, money, name, allies, number of regions of a player.

MapManagement: The distribution of the regions to the players will be provided by this class when the start game is clicked.

Region: Regions in the Risk Game will be defined by this class. This class will hold the number of troops in the region, whether a region has a commander, the climate of the region, the motivation of soldiers in the region, whether a region has gold mines and the neighbours of a region.

Continent: This class will be used to store the regions that form a continent.

TurnManagement: When a player is in turn, all the moves that the user can make will be completed in this class. Thus, this class will hold a user which is in turn, the stage of users and the season of the game. In this class, players can attack or fortify a region, draft troops to a region, buy mercenaries and move their commanders.

GameManagement: This class will be responsible for the starting a new game, loading a saved game, saving game, changing the turn of players, displaying the winners and finishing the game.

5.4. User Interface - Navigational Paths and Screen Mock-ups

5.4.1. Navigational Paths

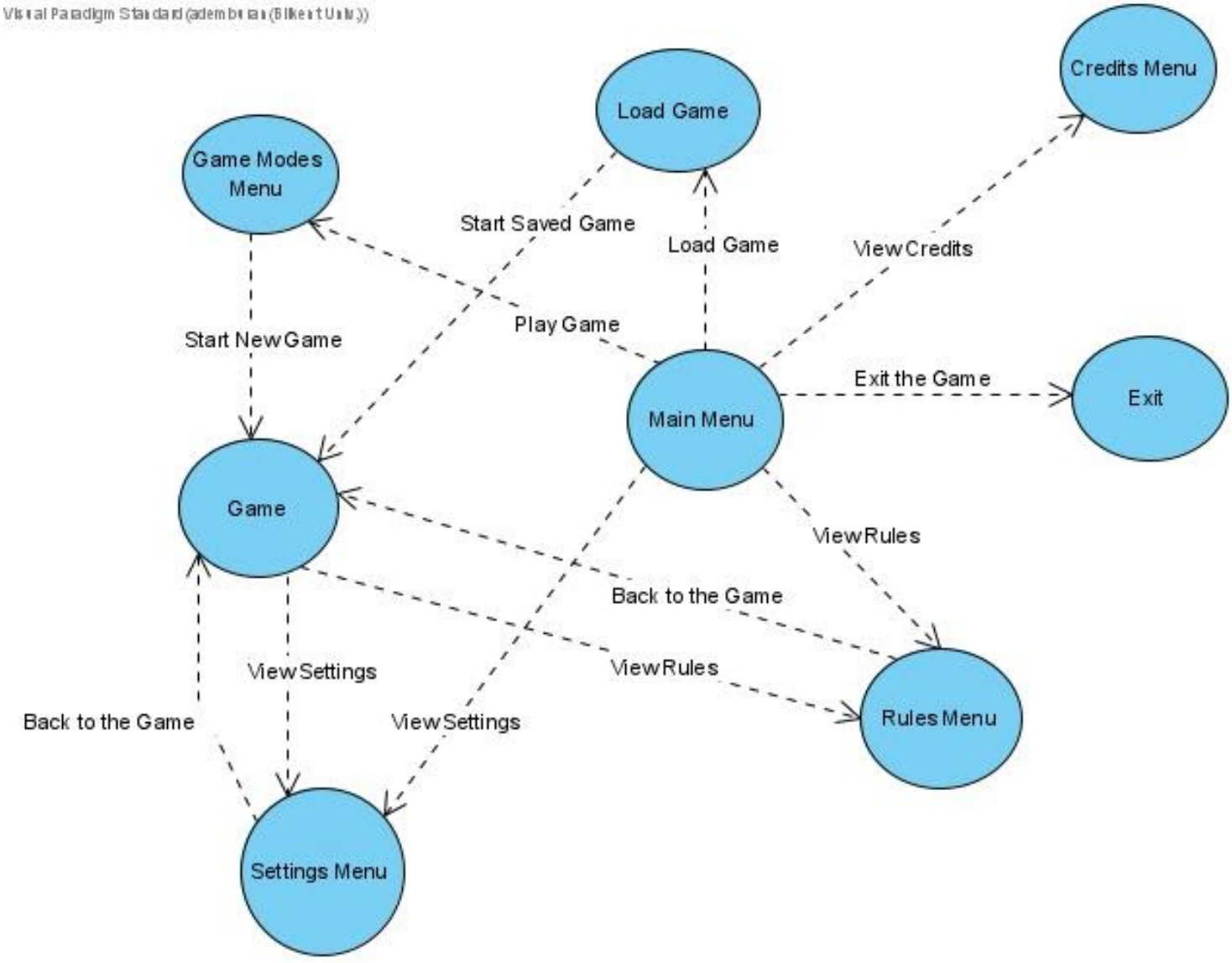

Figure 7: Sketch of the navigational path

5.4.2. Screen Mock-Ups

5.4.2.1. Main Menu

Figure 8: Main menu screen displayed when the game is opened

Main Menu is the screen consisting of 6 buttons that players see when they open the game. From the menu, players can create a new game, load a saved game, change options, read the game rules, see credits, and quit the game.

5.4.2.2. Game Settings

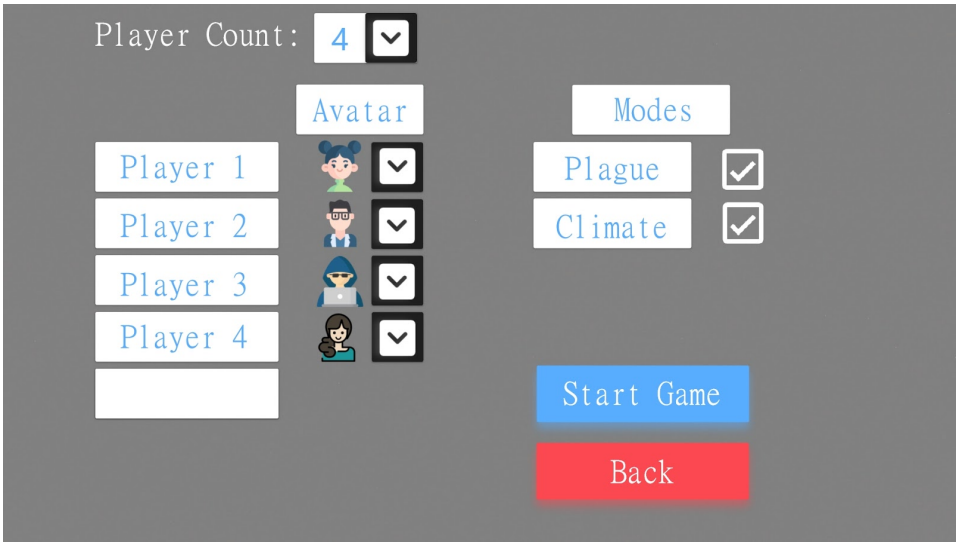

Figure 9: Game Settings is the screen in which players choose avatars and adjust the game modes

In the Game Settings menu, the player count is set. Each player chooses their avatar. Game modes are chosen. After setting up the game options, the player either starts the game by pressing the Start Game button or goes back to the main menu by pressing the Back button.

5.4.2.3. Credits

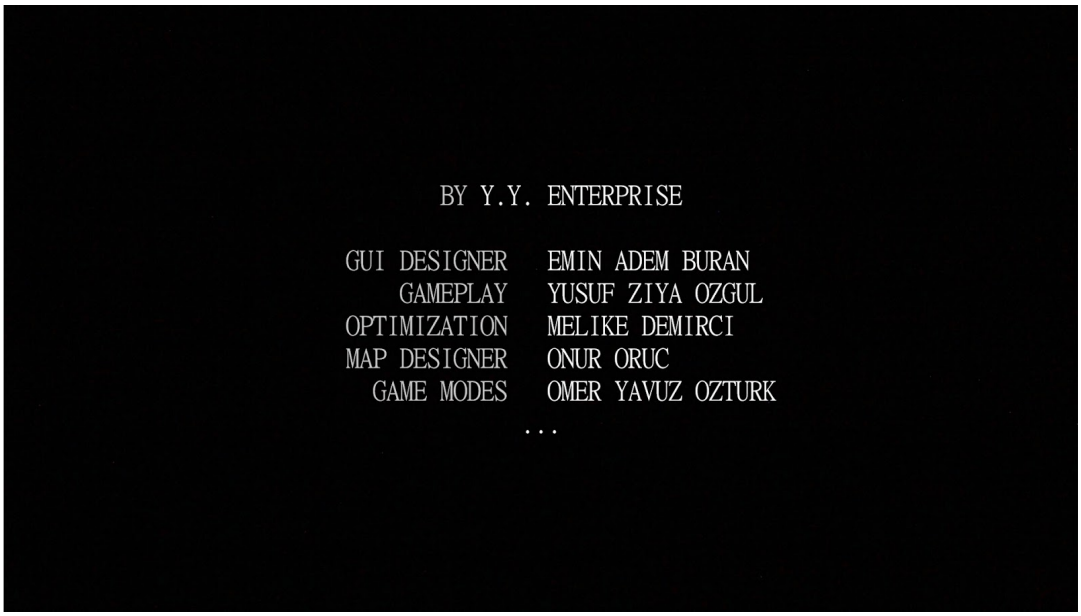

Figure 10: Credits page displaying the game contributors

In the Credits screen, players see which developer is responsible from which part of the game.

5.4.2.4. Settings

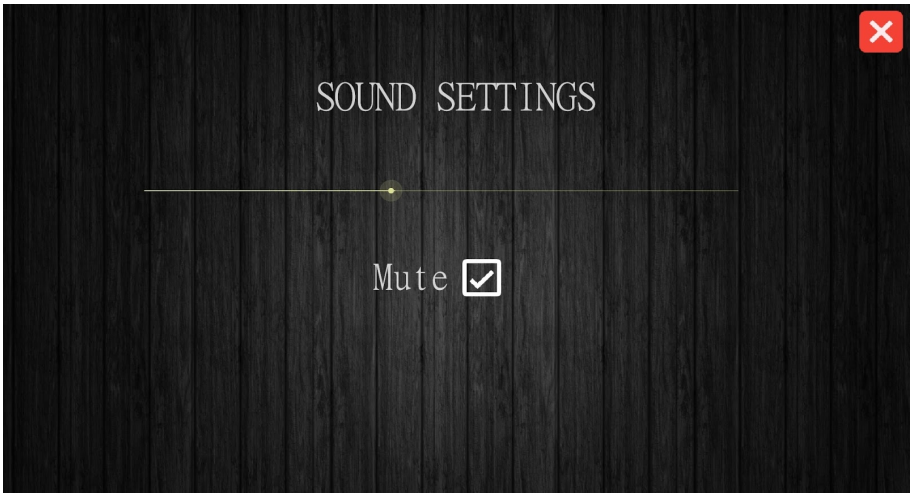

Figure 11: Sound Setting screen navigating the music volume in the game

In the Settings, players can adjust the music volume by sliding the bar or they can mute the music.

5.4.2.5. Rules

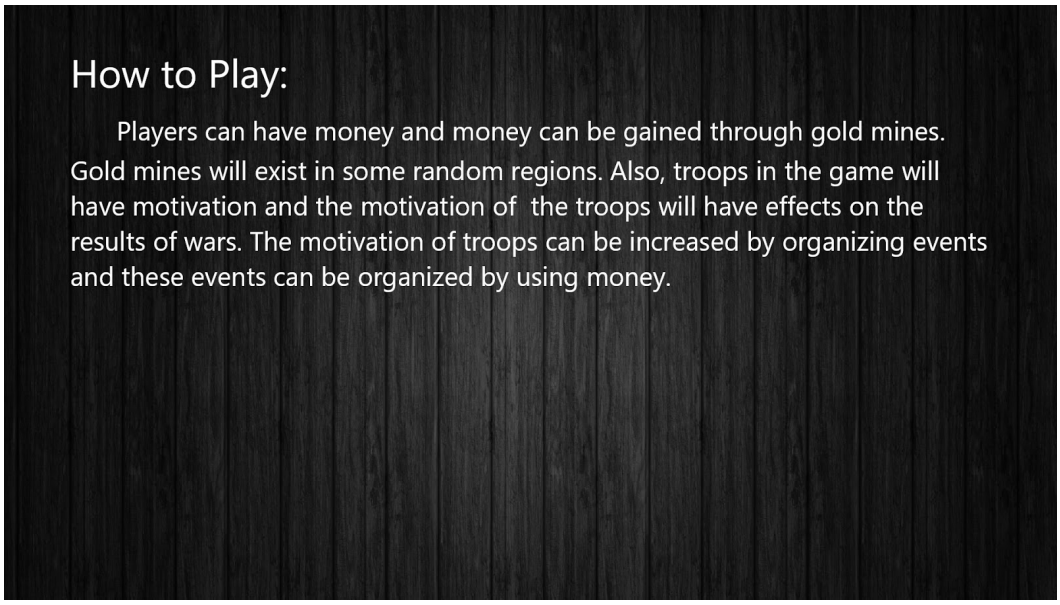

Figure 12: Rules screen displaying the game rules

In the Rules, players can read the game rules.

5.4.2.6. Game Screen

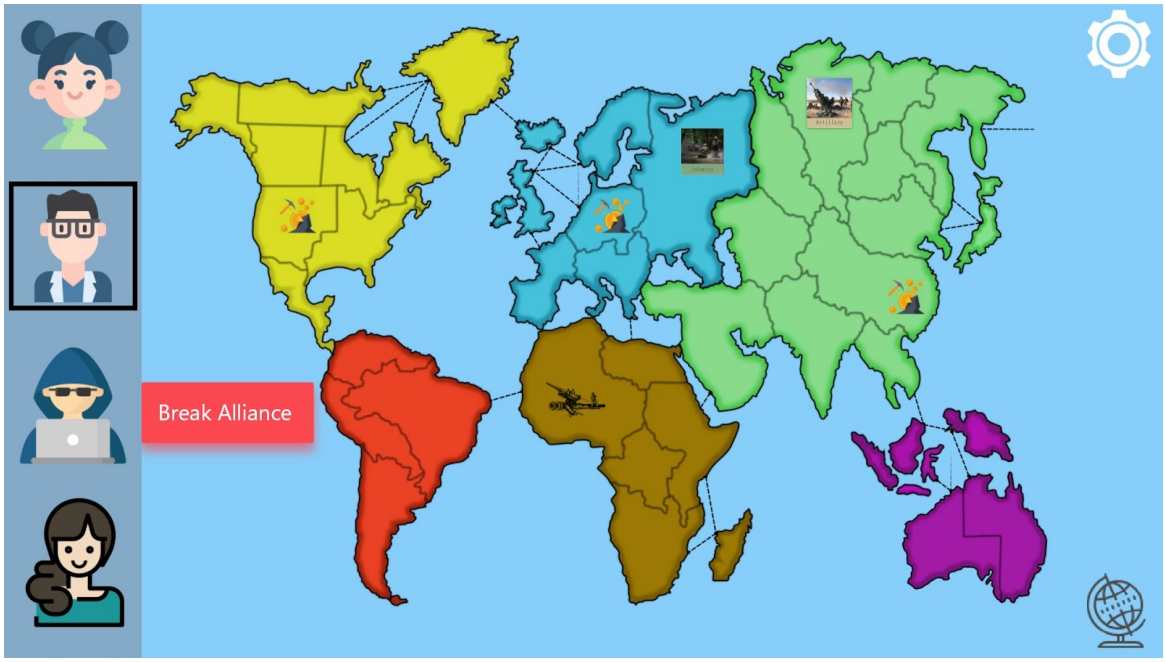

Figure 13: Game Screen displayed when the game has started

This screen is what players see when the game has started. The gold mines, a missile launcher, and troop cards that randomly placed in the map when the game has started is visible on the map. The player whose turn is active is marked with a bold rectangle. The player can break alliance if s/he has one with another player, or make alliance with a player by right clicking the related player's avatar. An example for a Break Alliance button is given in the figure. By clicking the settings icon on the top left, players can change sound settings, pause the game, save the game or quit the game.

5.4.2.7. Player Profile

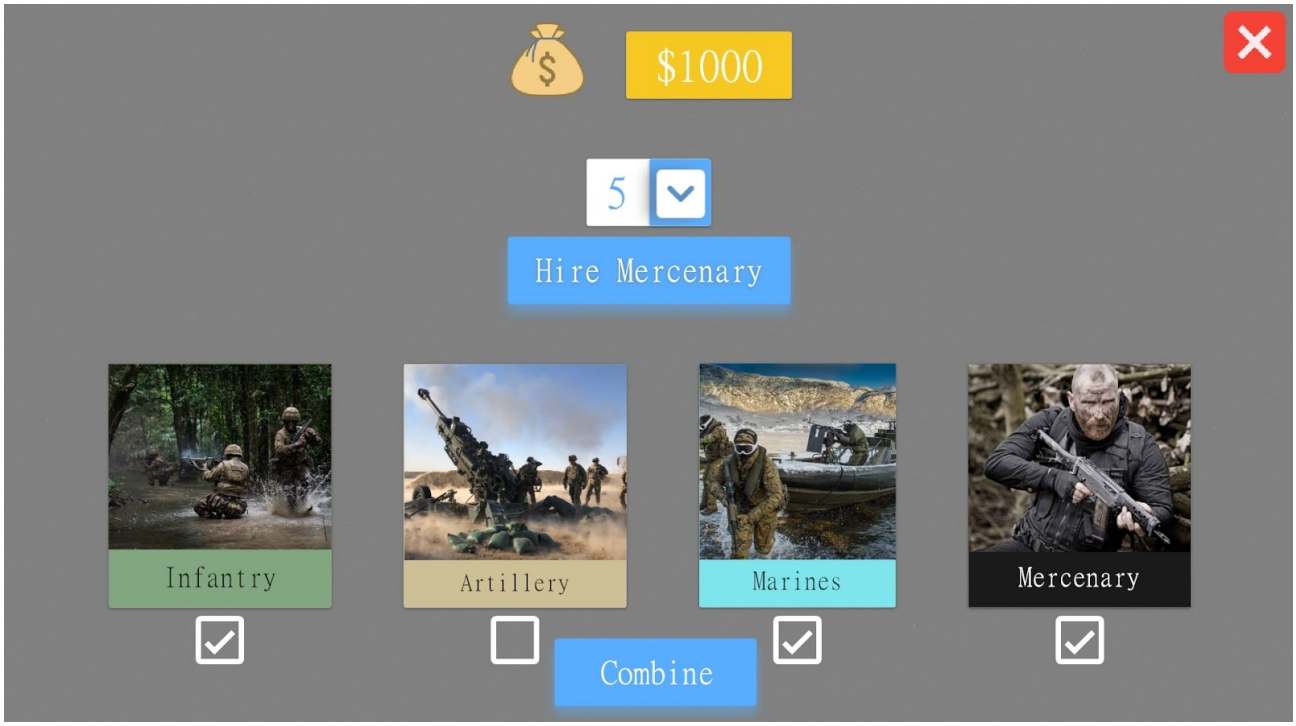

Figure 14: Player profile screen [1], [2], [3], [4]

This is the screen that the player whose turn is active sees when s/he clicks his/her avatar. The player can see his/her budget, and accordingly specify the number of mercenaries s/he will hire to place in regions at the beginning of his/her turn. S/he can combine the troop cards (if possible) to receive more soldiers in the current turn.

6. Glossary & References

[1] "Infantry: AT the Heart of the Action," *The British Army*. [Online]. Available: https://www.army.mod.uk/who-we-are/corps-regiments-and-units/infantry/. [Accessed: Nov. 1, 2020].

[2] T. South, "Return of fires: How the Army is getting back to its big guns as it prepares for the near-peer fight," *ArmyTimes*, Aug. 27, 2018. [Online]. Available: https://www.armytimes.com/news/your-army/2018/08/27/return-of-fires-how-the-army-is-getting-back-to-its-big-guns-as-it-prepares-for-the-near-peer-fight/. [Accessed: Nov. 1, 2020].

[3] "Role of the Royal Marines," *Royal Navy*. [Online]. Available: https://www.royalnavy.mod.uk/our-organisation/the-fighting-arms/royal-marines. [Accessed: Nov. 1, 2020].

[4] J. A. Tures, "Commentary: Ignoring mercenary threat is risky," *myjournalcourier*, July 20, 2020. [Online]. Available: https://www.myjournalcourier.com/opinion/article/Commentary-Ignoring-mercenary-threat-is-risky-15419674.php. [Accessed: Nov. 1, 2020].

* UI components have been designed using the Lunacy desktop application. Available: https://icons8.com/lunacy.