CS 319 TERM PROJECT

Section 1 Group 1I

Risk Game Analysis - Report Iteration 2

Emin Adem Buran - 21703279

Onur Oruç - 21702381

Ömer Yavuz Öztürk - 21803565

Melike Demirci - 21702346

Yusuf Ziya Özgül - 21703158

Supervisor: Eray Tüzün

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.

2.5. Clash of War Heroes

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.

2.6. 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.

2.7. 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.

2.8. Missions

There are four different missions in the game. When a game starts, each player is assigned to a mission. These missions include conquering all regions, conquering predetermined numbers of regions, eliminating predetermined player(s) from the game and conquering predetermined continents. When a player completes the assigned to him/her, this player wins the game.

3.1 Functional Requirements

3.1.1 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.

3.1.2 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 continue the game that was paused previously.
- System should display the game map after the player chooses to continue the game.

3.1.3 Settings

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

3.1.4 Display Rules

- System should display the rules of the game to the player.
- Players can learn about the rules of the game both before starting the game and during the game.

3.1.5 Load Game

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

3.1.6 Credits

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

3.1.7 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.

3.2 Additional Functional Requirements

- At the start of the game, the system will give missions to all players.
- When a player can complete the mission assigned to her/him, that player can win the game.

4.1. Nonfunctional Requirements

4.1.1 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.

4.1.2 Reliability

- The game should not crash during the 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.

4.1.3 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.

4.1.4 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.

4.2. Additional Nonfunctional Requirements

We do not have additional nonfunctional requirements.

5. System Models

5.1. Use Case Model

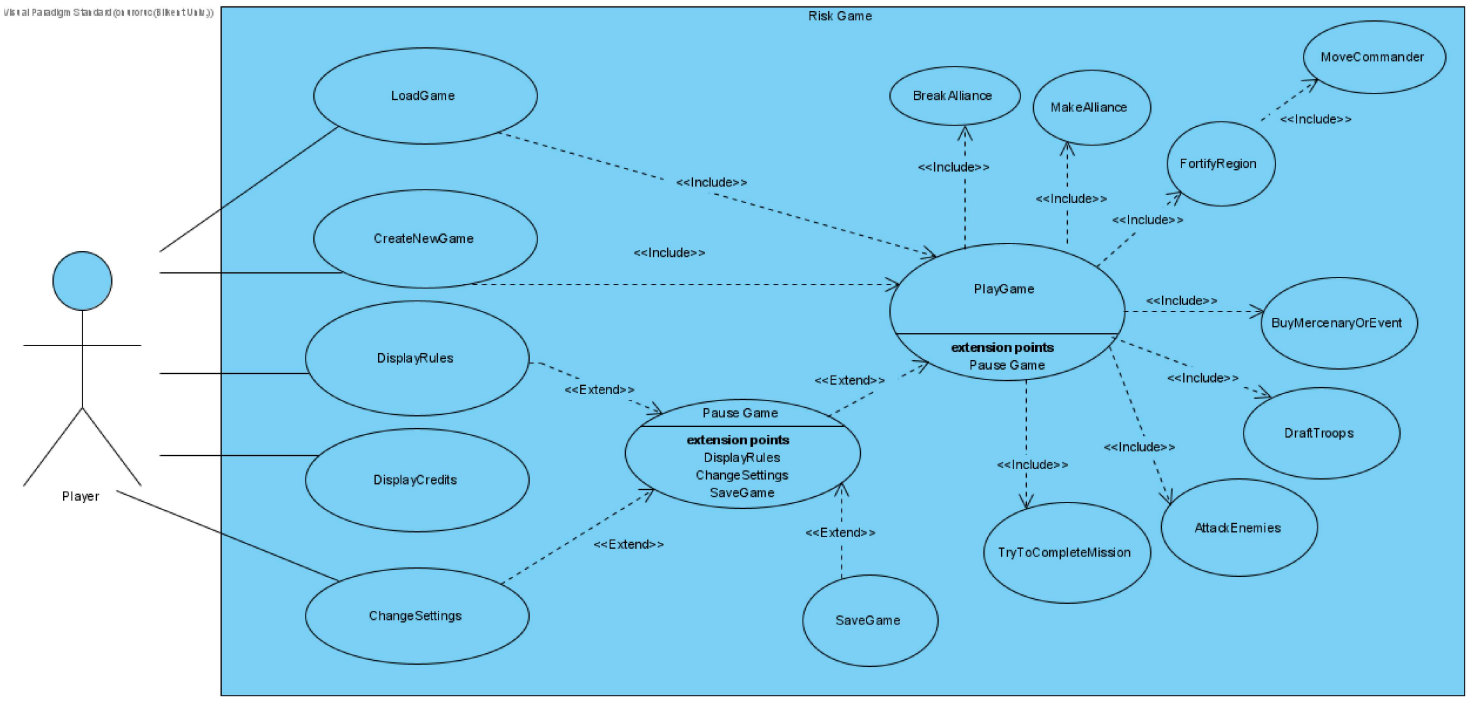

Figure 1: Risk Game use case diagram

Use Case 1: LoadGame

Participating Actor: Player

Entry Condition:

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

Exit Condition:

- The saved game is loaded by Player.

Flow of Events:

- Player displays the list of saved games.
- Player selects a game from the saved games.
- The saved game is fetched from the file in which it was saved previously by Player.

Special/quality Requirements:

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

Use Case 2: CreateNewGame

Participating Actor: Player

Entry Conditions:

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

Exit Condition:

- A new game is created by Player.
- Player is ready to play the game.
- The map initialization and soldier distributions are done.

Flow of Events:

1. Player sees the main menu.
2. Player selects the number of players that will play the game.
3. Player selects the modes of the game.
4. Player is ready to play a new game.

Special / Quality Requirements:

- It should not take more than 5 seconds to direct Player to the game map after determining the number of players and modes of the game.

Use Case 3: PlayGame

Participating Actor: Player

Entry Condition:

- Players are ready to play the game.
- A saved game is loaded.
- A new game is created.

Exit Condition:

- Game ends.

Flow of Events:

1. Players are given a mission which makes Player winner when completed.
2. Turn is given to the first Player.
3. This use case includes BuyMercenaryOrEvent use case to enable Player to buy mercenary(ies) or organize an event(s) in the regions with money.
4. This use case includes DraftTroops use case to allow Player to place troops in the regions that Player selects.
5. This use case includes AttackEnemies use case to allow Player to attack another Player(s).

6. This use case includes FortifyRegion use case to enable Player to move troops between the regions his/her own, and allow Player to move the commander independently of other troops.
7. Turn is given to the next Player and this use case continues with the third step again until the game ends.

Special/quality Requirements:

- The region and troop distribution should be done within 30 seconds.
- At any point during the flow of events, this use case can include the BreakAlliance and the MakeAlliance use cases. These use cases are initiated when Player wants to break alliance with his/her alliance(s) or make a new alliance(s).
- At any point during the flow of events, this use case can include the TryToCompleteMission use case to enable Player to keep track of the mission that is given to him/her to win the game.

Use Case 4: 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 sees the contributors of the game.
2. After all credits have been displayed, Player is directed to the main menu.

Special/quality Requirements:

- This screen should be cinematic.

Use Case 5: DisplayRules

Participating Actor: Player

Entry Condition:

- Player opens the game and sees the main menu.
- Player starts to play the Risk game.
- This use case extends the PauseGame use case. It is initiated when Player wants to see the game rules during an ongoing game.

Exit Condition:

- Player has information about how to play the game.

Flow of Events:

1. Player sees the game rules.
2. Player learns the rules of the game.

Special/quality Requirements:

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

Use Case 6: 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:

- The game settings have been changed by Player.

Flow of Events:

1. Player adjusts the music volume or mute the music.
2. 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 an ongoing game).

Special/quality Requirements:

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

Use Case 7: BreakAliance

Participating Actor: Player

Entry Condition:

- Player is in turn.
- Player wants to break alliance with his/her alliance(s).

Exit Condition:

- Player breaks alliance with his/her ally.

Flow of Events:

1. Player sees his/her allies.
2. Player chooses an ally that Player wants to break alliance with.

3. Alliance between Player and his/her alliance is broken and goes to the first step of the flow of events if s/he intends to break alliance with another alliance.

Special/quality Requirements:

- The system should allow Player to cancel the break alliance decision.

Use Case 8: MakeAliance

Participating Actor: Player

Entry Condition:

- Player is in turn.
- Player's request to make an alliance is sent to the corresponding Player.

Exit Condition:

- Player makes an alliance with other Player(s).

Flow of Events:

1. Player sees the player list.
2. Player chooses another Player to make an alliance
3. Alliance between Player and his/her alliance is broken and goes to the first step of the flow of events if s/he intends to break alliance with another alliance.
4. Player reviews if any alliance request has arrived at him/her by other Player(s). If a request has arrived, Player either refuses or accepts to be in alliance with.

Special/quality Requirements:

- The system should allow Player to cancel the decision about making an alliance.

Use Case 9: FortifyRegion

Participating Actor: Player

Entry Condition:

- Player is in turn.
- Player enters the fortify stage of the turn.

Exit Condition:

- Player fortifies a region(s) or
- Player skips fortify stage.

Flow of Events:

1. Player either chooses to not perform a fortify operation and gives his/her turn to the next Player. If Player chooses the former, this use case ends here.

Otherwise, it continues with the second step.

2. Player is asked to select one of his/her regions from which s/he will move troops.
3. Player chooses one of his/her regions which has more than one troops.
4. Player then selects another region of his/her own to which troops will be moved. This region has to be connected with the first region (i.e., these two regions have to be neighbors or all the regions connecting these two regions should be owned by Player).
5. Player chooses the number of troops that will be moved to the other region.
6. If Heat and Snow mode is active and the selected regions suffer from drought or frost, some of the troops will die. Remaining troops will be moved to the selected region.
7. This use case includes the MoveCommander use case. The MoveCommander use case is initiated when Player wants to move the commander independently of the troops.
8. Player is directed to the first step.

Special/quality Requirements:

- System should not allow Player to select a region that has one troop.

Use Case 10: MoveCommander

Participating Actor: Player

Entry Condition:

- Player is in turn.
- Player is in the fortify stage.

Exit Condition:

- Player moved commander or
- Player skips this stage without moving the commander.

Flow of Events:

1. Player is asked to select one of his/her regions from which s/he will move the commander.
2. Player can skip this stage without moving the commanders s/he has. If Player chooses to move the commander, then Player is asked to choose another region to move the commander. This region has to be connected with the first region (i.e., these two regions have to be neighbors or all the regions connecting these two regions should be owned by Player).

3. The commander is moved to the selected region.

Special/quality Requirements:

- The system should not allow Player to move the commander more than once in a single turn.

Use Case 11: BuyMercenaryOrEvent

Participating Actor: Player

Entry Condition:

- Player is at the start of the turn.
- Player is in the buy stage.

Exit Condition:

- Player has a mercenary(ies) and/or organizes an event(s) in the corresponding regions.
- Motivation level of the regions in which an event was performed increases.
- Player proceeds to the draft stage without buying mercenary(ies) and organizing events.

Flow of Events:

1. Player is asked to buy a mercenary(ies) or organize an event(s) with money.
2. Player can directly skip this stage and go to the draft stage.
3. If Player chooses to buy mercenary(ies), Player chooses the number of mercenary(ies) to be bought with his/her money.
4. If Player does not want to organize an event, nothing more needs to be done. Otherwise, Player decides to organize an event(s), and selects a region(s) in which the event(s) will occur.

Special/quality Requirements:

- System should not allow Player to mercenary(ies) requiring more money than Player has.

Use Case 12: DraftTroops

Participating Actor: Player

Entry Condition:

- Player is in turn.
- Player is in the draft stage.

Exit Condition:

- Troops are placed into the selected regions by Player.

Flow of Events:

1. Player chooses a region to place the troops (mercenary(ies) from the buy stage, bonus troops if Player combined some of troop cards, default number of troops that are given at the beginning of the turn, and the extra troops if Player owns a continent(s)).
2. Player chooses the number of troops to be drafted to the selected region.
3. If there are remaining troops, Player performs the first step of this use case again.

Special/quality Requirements:

- Player should be able to cancel a draft operation.

Use Case 13: AttackEnemies

Participating Actor: Player

Entry Condition:

- Player is in turn.
- Player is in the attack stage.
- Player has at least two troops in the region which player wants to start an attack.

Exit Condition:

- Player performed attack(s) or
- Player directly skips this stage.

Flow of Events:

1. Player can skip this use case and go to fortify state. If Player decides to skip, the use case ends here. Otherwise, it continues with the second step.
2. Player chooses one of his/her own regions from which the attack comes from. This region should have more than one troops.
3. Player chooses a neighboring region owned by another Player.
4. Player chooses the number of troops which will be used for the attack.
5. If Player chooses to attack with one troop, Player throws one dice. If Player chooses to attack with two troops, Player throws two dice. If Player chooses to attack with three troops, Player throws three dice.
6. Attack is applied and whether Player wins or loses are calculated.
7. If Player fails in the attack and decides to perform another attack to the same region, Player performs the fourth step again. If Player decides to attack another region from

another region of his/her own, Player is directed to the second step of AttackEnemies use case. If Player does not want to perform any more attacks, the use case ends here.

Special/quality Requirements:

- The system should allow Player to attack as many as Player wants as long as the regions from which the attack will happen has more than one troops.

Use Case 14: TryToCompleteMission

Participating Actor: Player

Entry Condition:

- Player starts the game.

Exit Condition:

- Player completes the mission given at the beginning of the game.

Flow of Events:

1. Player sees the mission assigned to her/him at the beginning of the game.
2. Player plays the game according to the mission assigned to her/him.
3. If a player completes the mission assigned to him/her, the game is over.

Special/quality Requirements:

- Player should be able to see the corresponding mission at any time during the turn.

Use Case 15: SaveGame

Participating Actor: Player

Entry Condition:

- This use case extends the PauseGame. It is initiated when Player is in the game and opens the Pause Menu.

Exit Condition:

- Current game is saved for future uses.

Flow of Events:

1. Player goes to the pause menu.
2. If Player decides to save the game, Player selects the save game option.
3. The game is saved for the future.

Special/quality Requirements:

- The system should not crash while a game is being saved.

Use Case 16: PauseGame

Participating Actor: Player

Entry Condition:

- This use case extends the StartGame use case. It is initiated when Player wants to save the game, wants to change settings, or wants to display rules.

Exit Condition:

- Player is ready to continue the game, OR
- Player quits the game by selecting the quit game option.

Flow of Events:

1. Player sees the Pause Menu.
2. Player saves the game, or displays rules, or changes sound, or quits the game.

Special/quality Requirements:

- Player should be able to return to the game after exiting the Pause Menu.

5.2. Dynamic Models

5.2.1. Sequence Diagram

5.2.1.1 Draft Troop

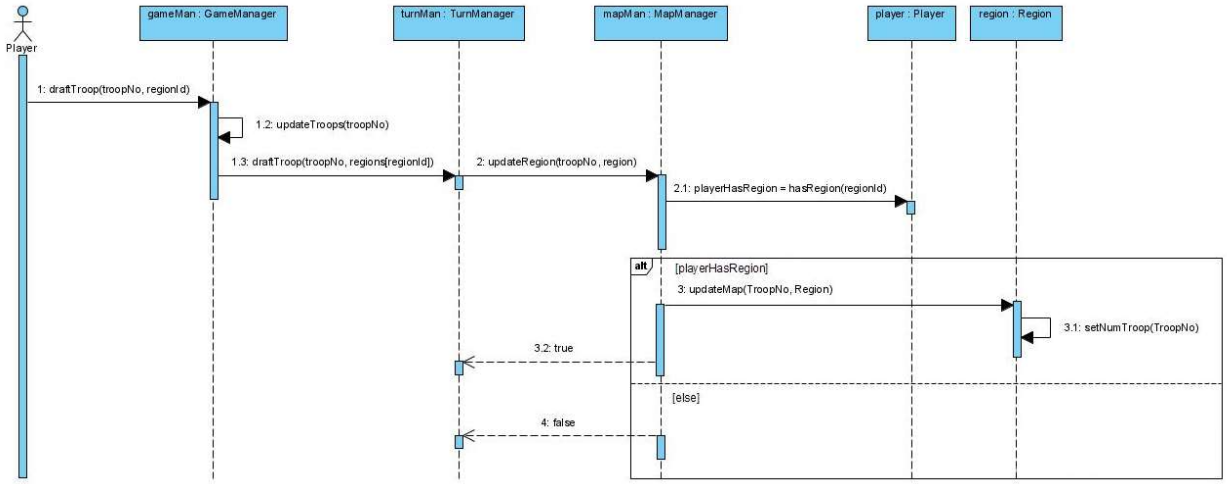

Figure 2: Sequence diagram illustrating the actions to drafting troops

Scenario: The player presses on a region on the game map and selects the number of troops to be drafted. User interface class passes the region id and number of troops to the GameManager which is a class that connects the gui and game logic. User interface sequences are not added to the diagram for simplicity. GameManager class updates remaining troop numbers for draft according to the number of troops decided from the player. Then passes the troop number and region to the TurnManager class. TurnManager also passes these to the MapManager class which checks if the given region belongs to the current player. If the player has the region number of troops is updated accordingly and updateRegions method returns true; else no change is done and method returns false.

5.2.1.2. Attack

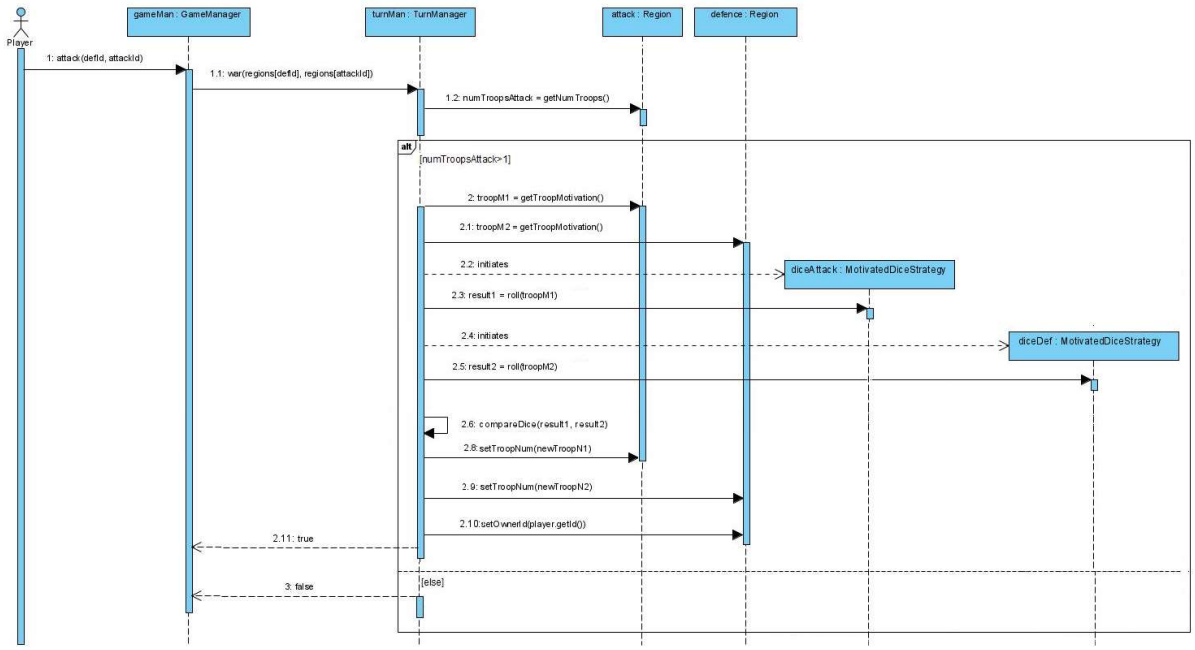

Figure 3: Sequence diagram illustrating the player actions to attack to a region

Scenario: The player presses on one of her/his regions and another region to be attacked. User interface class passes the region ids to the GameManager. GameManager class passes the regions for the given ids to the TurnManager class. TurnManager class first checks if the number of troops of the attacking region is more than one. If there is only one troop in the region, war method returns false and does nothing. If there is more than one troop, troop motivations of the two regions

acquired and two different dice are constructed by using these motivations. Two dice are rolled; one for attacking region the other for defending region. Results are compared in the method compareDice. Thus, according to the compareDice method new troop numbers for the regions are decided. If there are not enough troops left in the defending region, the attacking player becomes the new owner of the region by using setOwnerId method. Lastly, war method returns true.

5.2.1.3 Fortify

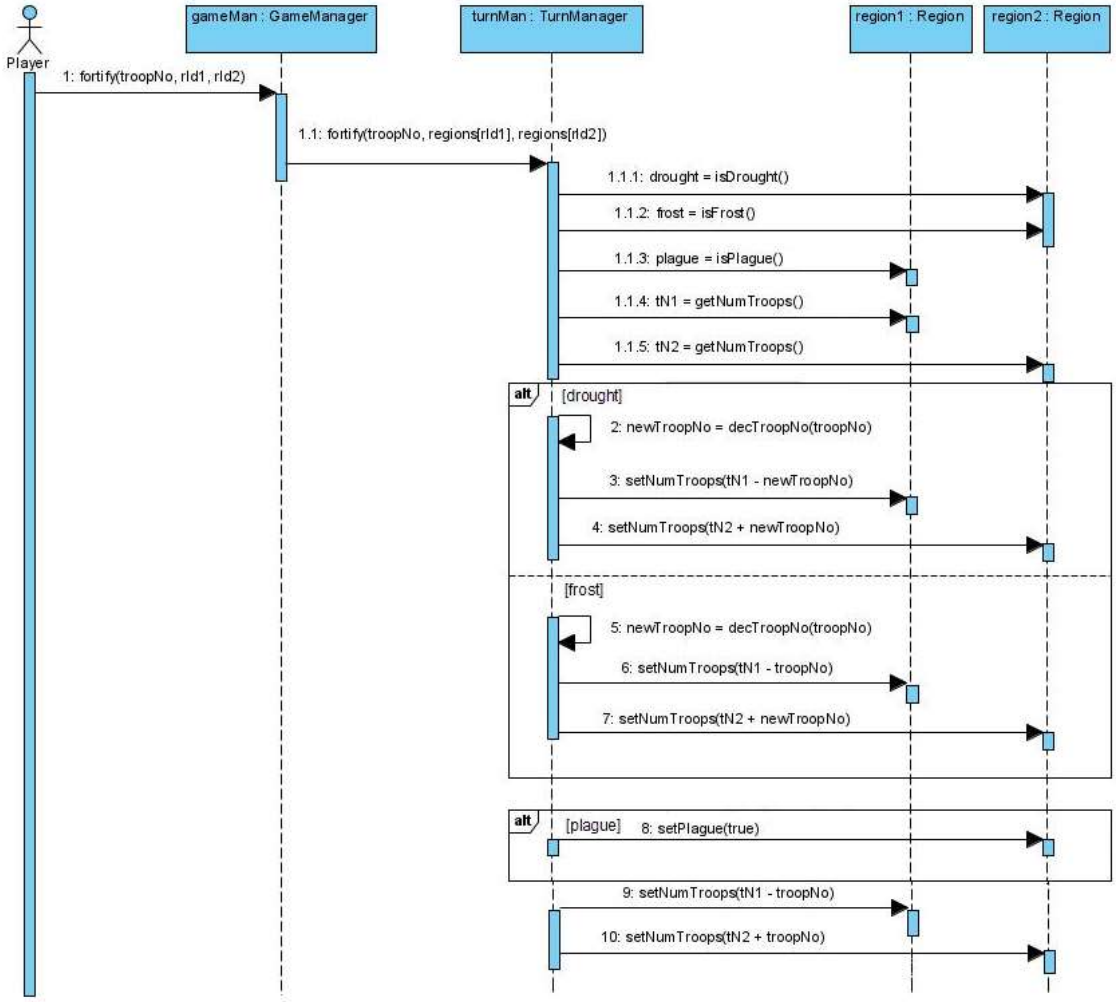

Figure 4: Sequence diagram illustrating the player actions to fortify troops

Scenario: The player presses on two of her/his regions; the one is giving troops and the other region is receiving these troops. Also the player selects the number of troops. User interface class passes the region ids and number of troops to the GameManager. The GameManager class passes the number of troops and the regions for the given ids to the TurnManager class. In the TurnManager class's fortify method, drought and frost information of the troop receiving region is acquired by using getter methods. Also the plague information about the other region is acquired. If there is a drought or frost in the receiving region, the number of troops to be fortified is decreased by using the decTroopNum method. The difference is when there is a drought there is no loss of troops , only the number is limited. On the other hand when there is a frost, some troops are lost. Moreover, by using the plague information of the first region, the second region's plague property is set, which means if there is a plague in the first region it is spreaded to the other region as a result of fortification. Lastly, if the troop receiving region is in the normal weather conditions, troops are fortified without any change in the amount.

5.2.1.4 Organize Event

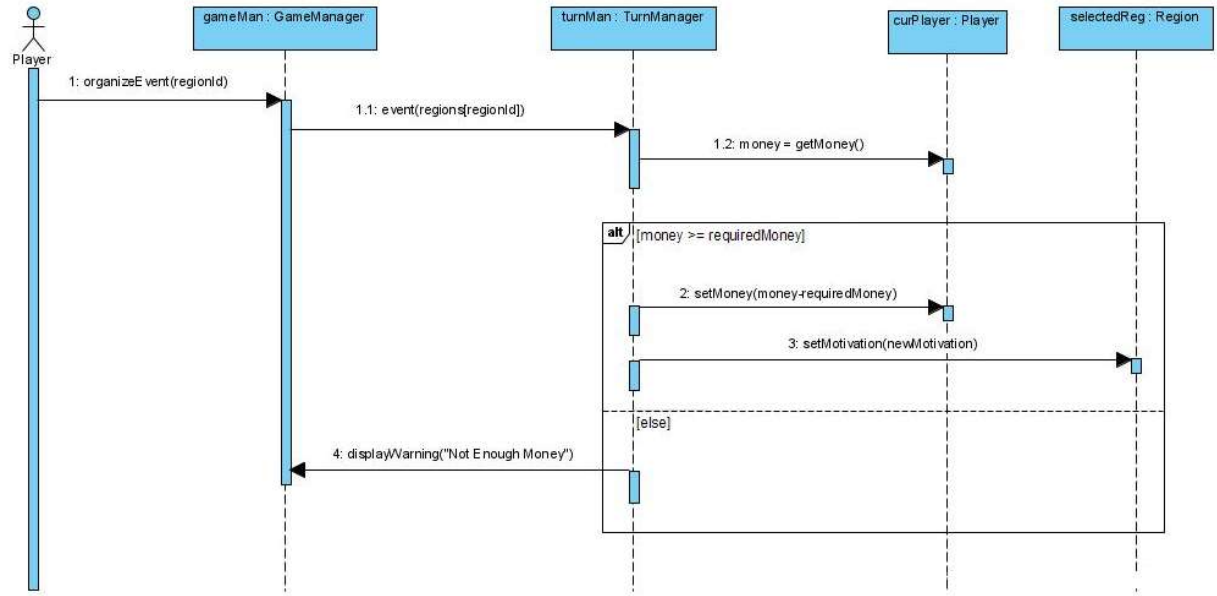

Figure 5: Sequence diagram illustrating the player actions to organize event

Scenario: The player presses on one of her/his regions in which the event will be organized. User interface class passes the region id to the GameManager. The GameManager class passes the region for the given id to the TurnManager class. In the TurnManager class's event method, first money of the current player is acquired by using the getter method. If this money is more than or equal to the cost of the event, money of the player is decreased according to the cost and motivation of the region is increased and method returns true. If there is not enough money, the event method returns false.

5.2.1.5 Buy Mercenary

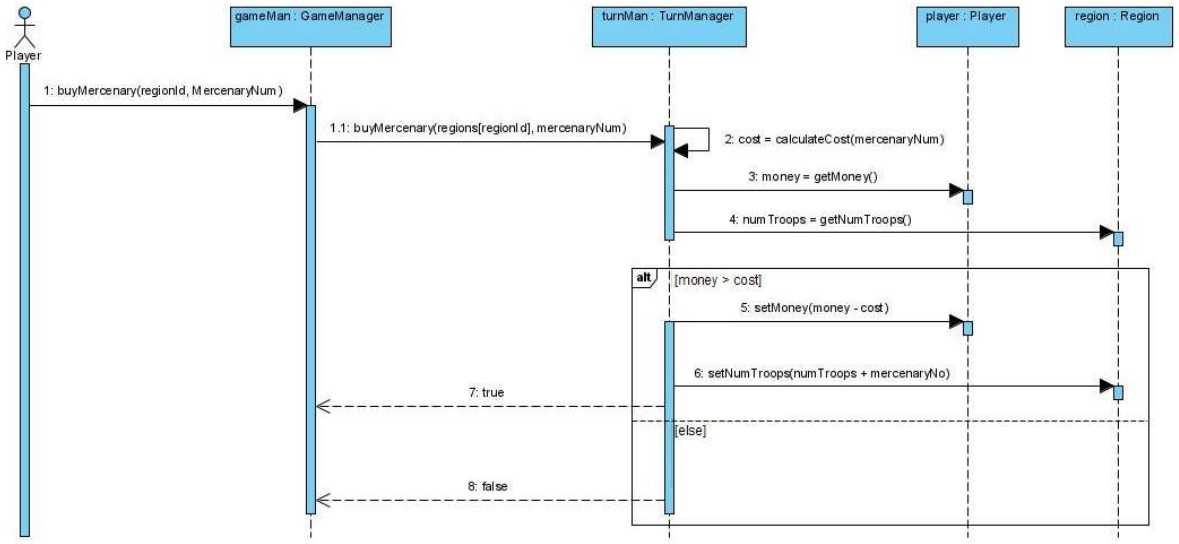

Figure 6: Sequence diagram illustrating the player actions to buy mercenary

Scenario: The player presses on one of her/his regions in which the mercenary will be placed. User interface class passes the region id to the GameManager. The GameManager class passes the region for the given id to the TurnManager class. In the TurnManager class's buyMercenary method, first money of the current player and number of troops in the region is acquired by using the getter methods. If this money is more than or equal to the cost of the mercenary, money of the player is decreased according to the cost and number of troops updated and method returns true. If there is not enough money, the buyMercenary method returns false.

5.2.2 Activity Diagrams

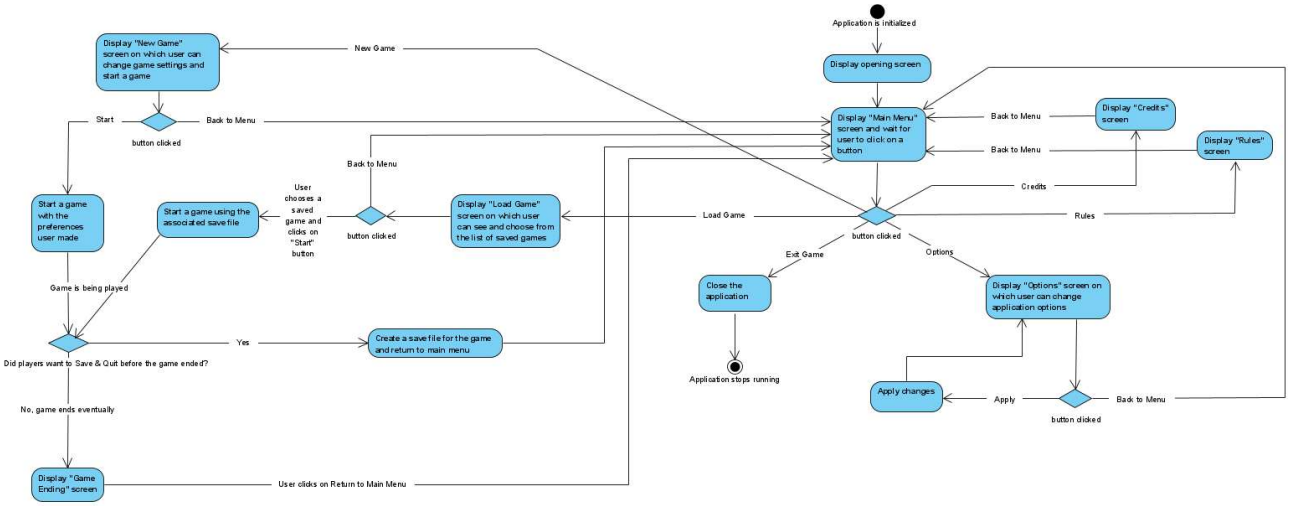

Figure 7: Activity diagram for Game Menu

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 a 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 the user saves and quits the game, the 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 the 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.

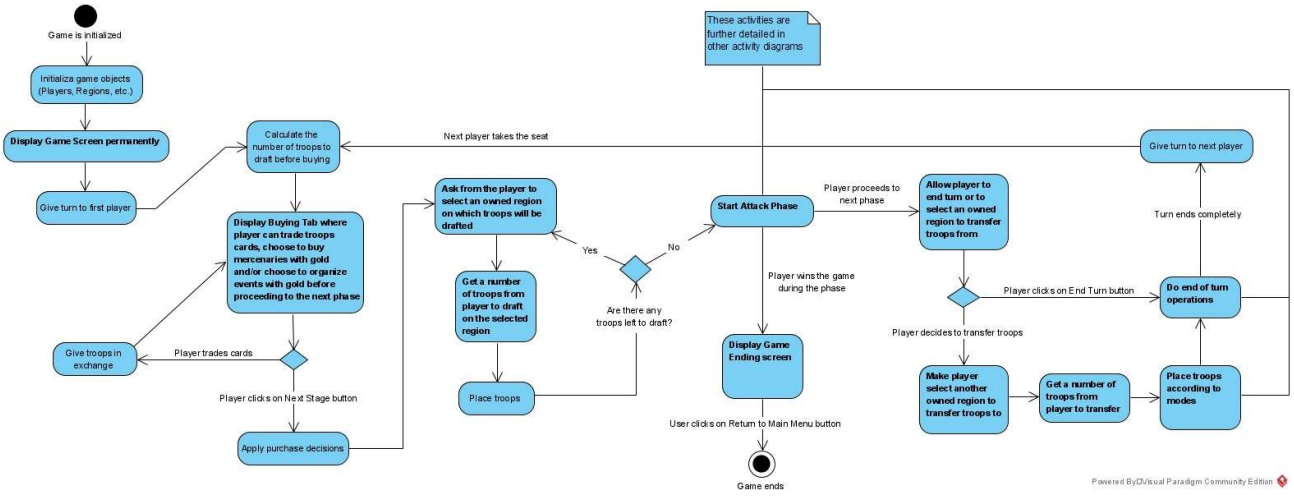

Figure 8: Activity Diagram for Gameplay

This diagram covers the whole gameplay. After the game begins, game objects will be initialized and Game Screen will be displayed until the game is won after which Game Ending screen is displayed. While the game is going on, players will take turns one by one and they will complete the phases of their turns which are Buy, Draft, Attack, and Fortify. System will arrange the game objects according to player inputs during these phases. Three of the activities in this diagram (which are connected to the note) are explained in more detail in the following diagrams. Also in our Risk game, a player can win only in Attack Phase.

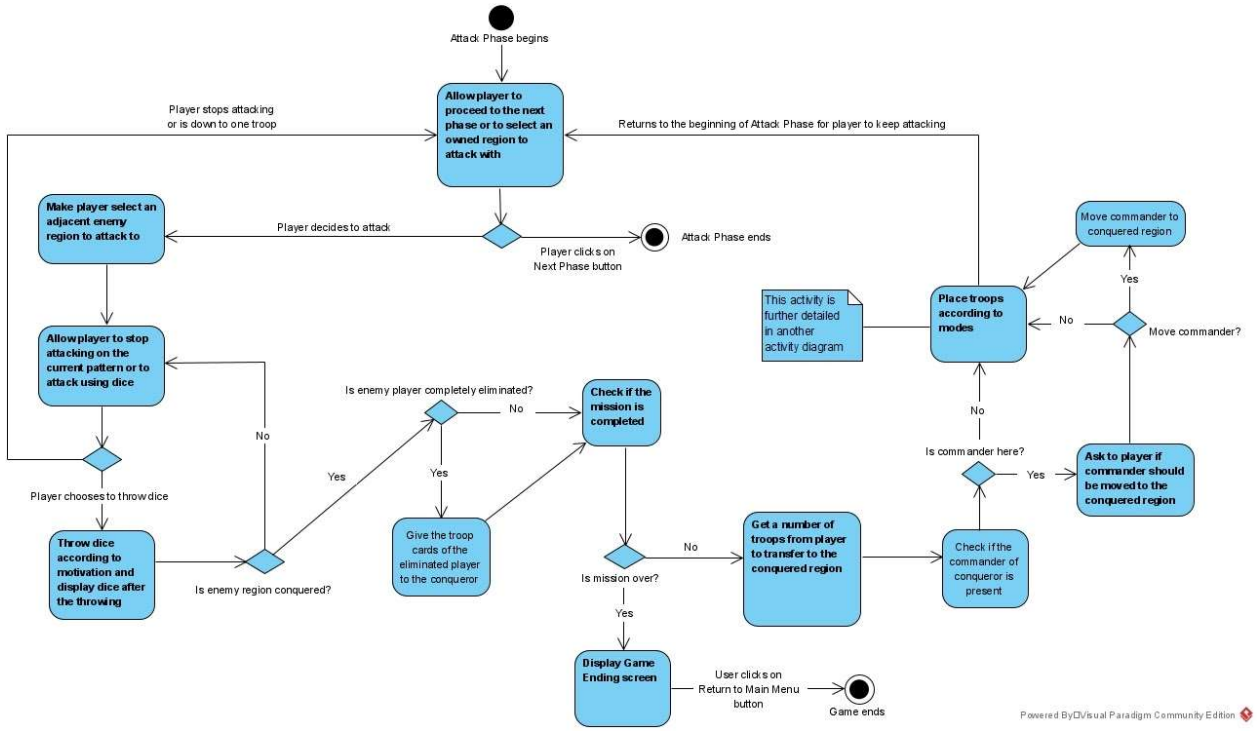

Figure 9: Activity Diagram for Attack Phase

This diagram covers the Attack Phase of the gameplay. Players will be able to attack other players' regions in this phase. The phase includes two main loops. First loop is that the player can attack somewhere and when the attacking to that somewhere is done, he/she can move on to attacking elsewhere. Second loop is that the player can keep attacking on the same pattern using dice, which can be thrown as long as there are enough troops on both sides. After any region is conquered, the win condition of the player will be checked and if it is satisfied, the game ends. Otherwise the player proceeds to the next phase when he/she is done attacking. Diagram is drawn separately in order to simplify the main gameplay diagram.

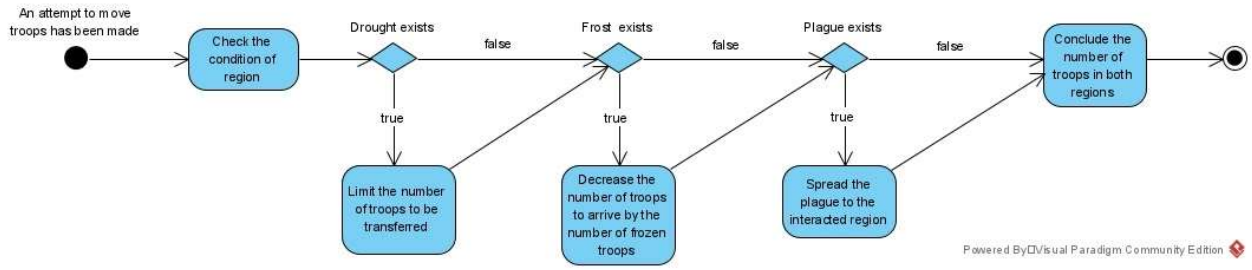

Figure 10: Activity Diagram for Placing Troops According to Modes

This diagram covers the movement of troops in Attack and Fortify phases. In our Risk game, a region can experience three negative events which are Drought, Frost and Plague. While these events occur in a region, the movement of troops is affected. It is drawn separately in order to simplify the main gameplay diagram and because this operation is repeated.

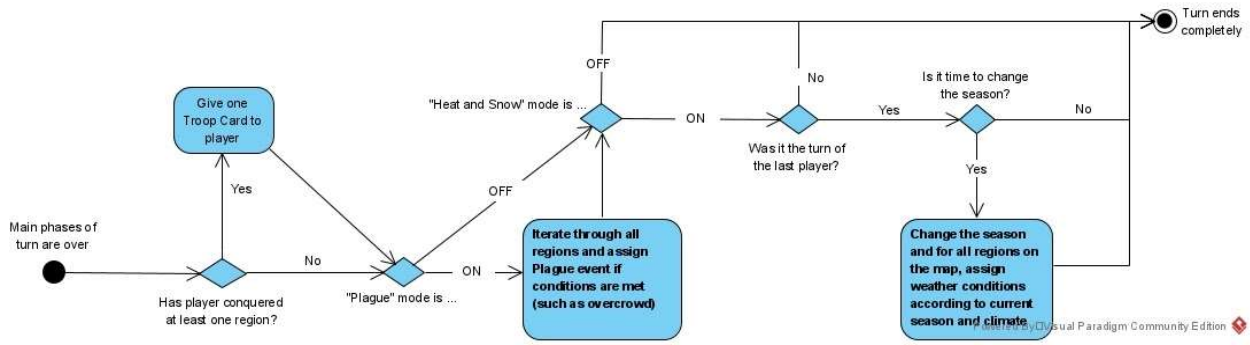

Figure 11: Activity Diagram for End of Turn Operations

This diagram covers the operations which will be done after every turn of any player. Giving a Troop Card action will be done in every game but executions of the other two actions depend on the modes or on the turn the game is on. Diagram is drawn separately in order to simplify the main gameplay diagram.

5.2.3. State Diagrams

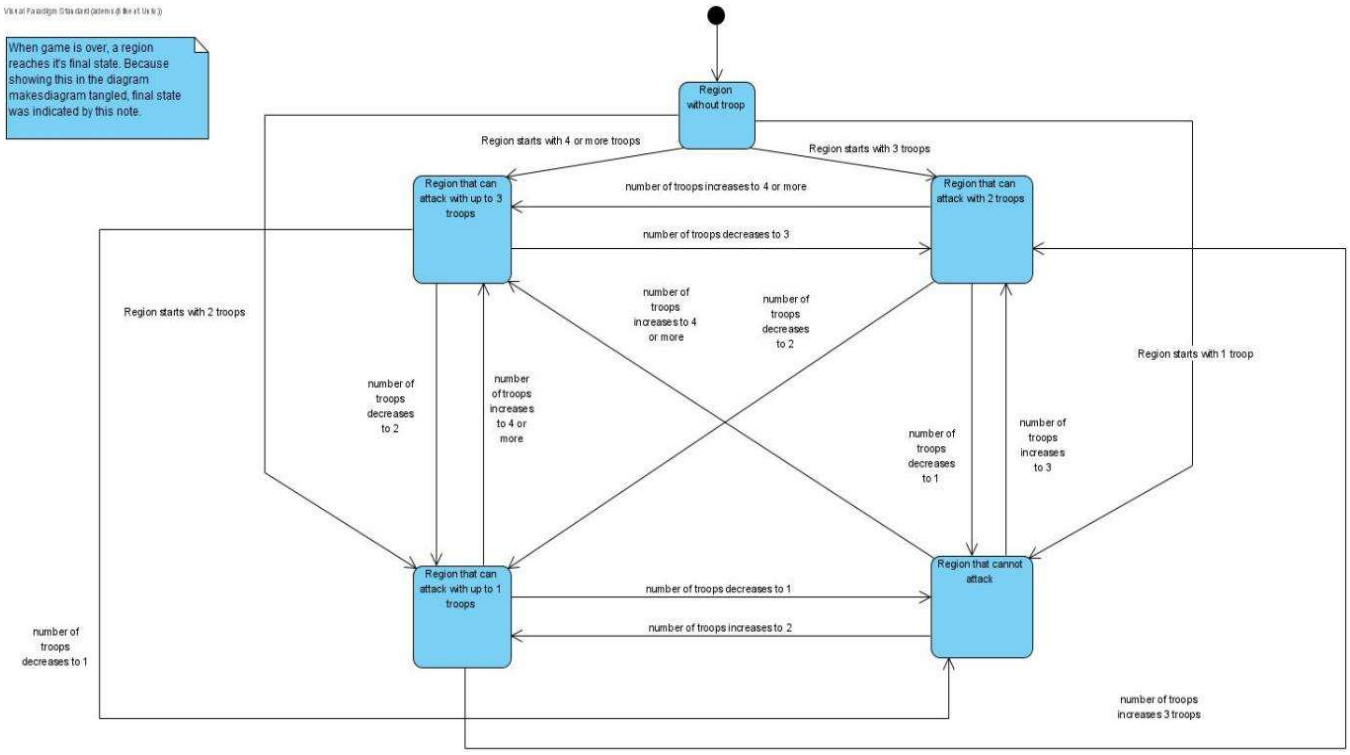

Figure 12: State diagram for attacking ability of a region.

The state diagram given above shows how the eligibility of any region for attacking changes throughout the whole game. If the number of troops in any region is smaller than 2, it means there is no opportunity to start an attack from this region to the neighbouring regions. At the beginning of the game every region is in “Region without Troop”. Then, the system adds troops to regions randomly. If a player fortifies the troop(s), the number of troops can increase in a region or if a player attacks another region, the number of troops may decrease considering the result of attacking. A player can attack using at most 3 troops for one attack, if the region has more than 3 troops. If a region has exactly 3 troops, at most 2 troops can be used to attack. If a region has exactly 2 troops, exactly 1 troop can be used to attack. Therefore, there are five states and passing between two states occurs according to changes in the number of troops in the region.

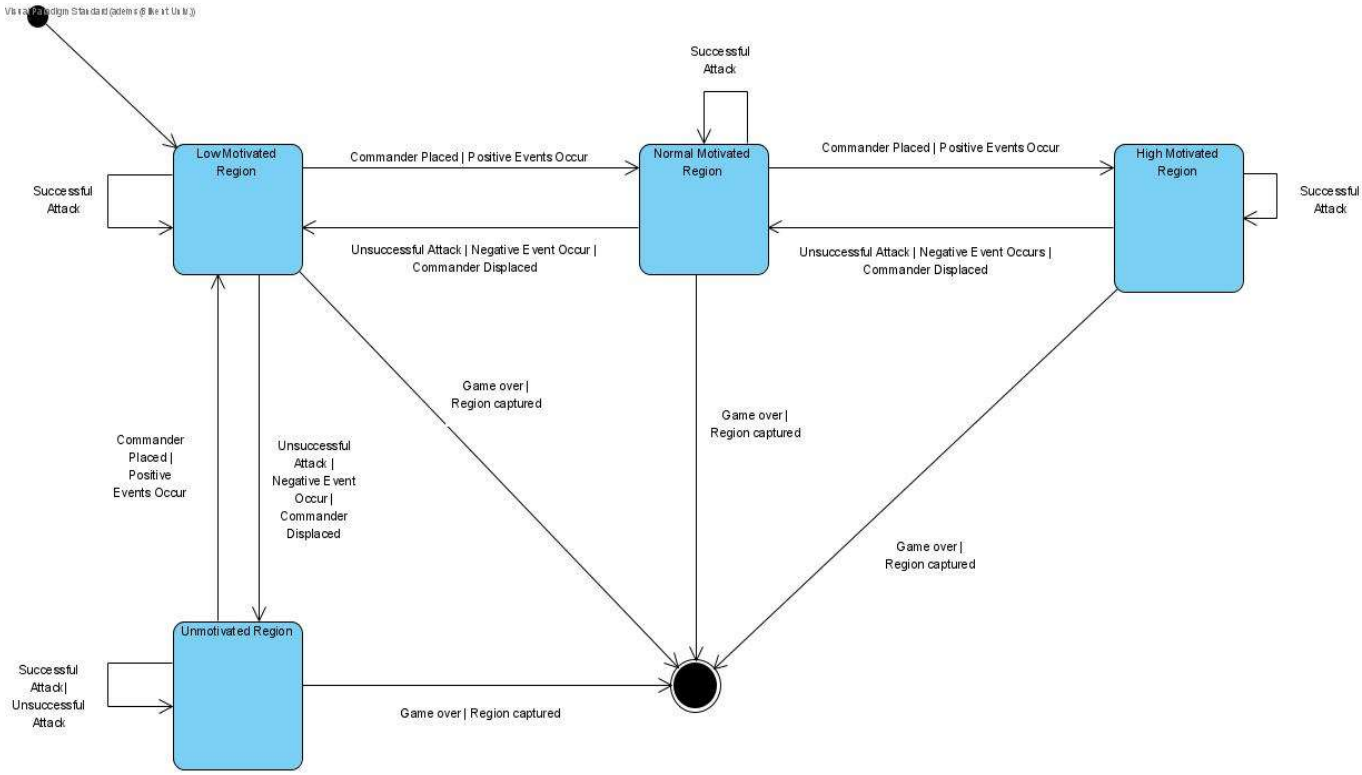

Figure 13: State diagram for motivation of a region.

The state diagram given above shows the motivation level of any region throughout the game. There are four possibilities for any region in terms of motivation. A region can be high motivated or normal motivated or low motivated or unmotivated. At the beginning of the game, each region is low motivated and when the owner of a region changes, this region becomes a low motivated region. The motivation level of a region might change according to the success of an attack. If the attacking region loses the war, motivation in this region decreases. Also, placing a commander increases the motivation of the region and displacing commanders decreases the motivation of the region. There are positive events like organizing festivals to increase motivation and this positive event increases motivation. Similarly, negative events like bad weather and plague decrease the motivation of the region. Lifetime of the region ends when the game is over or the region is captured by another player.

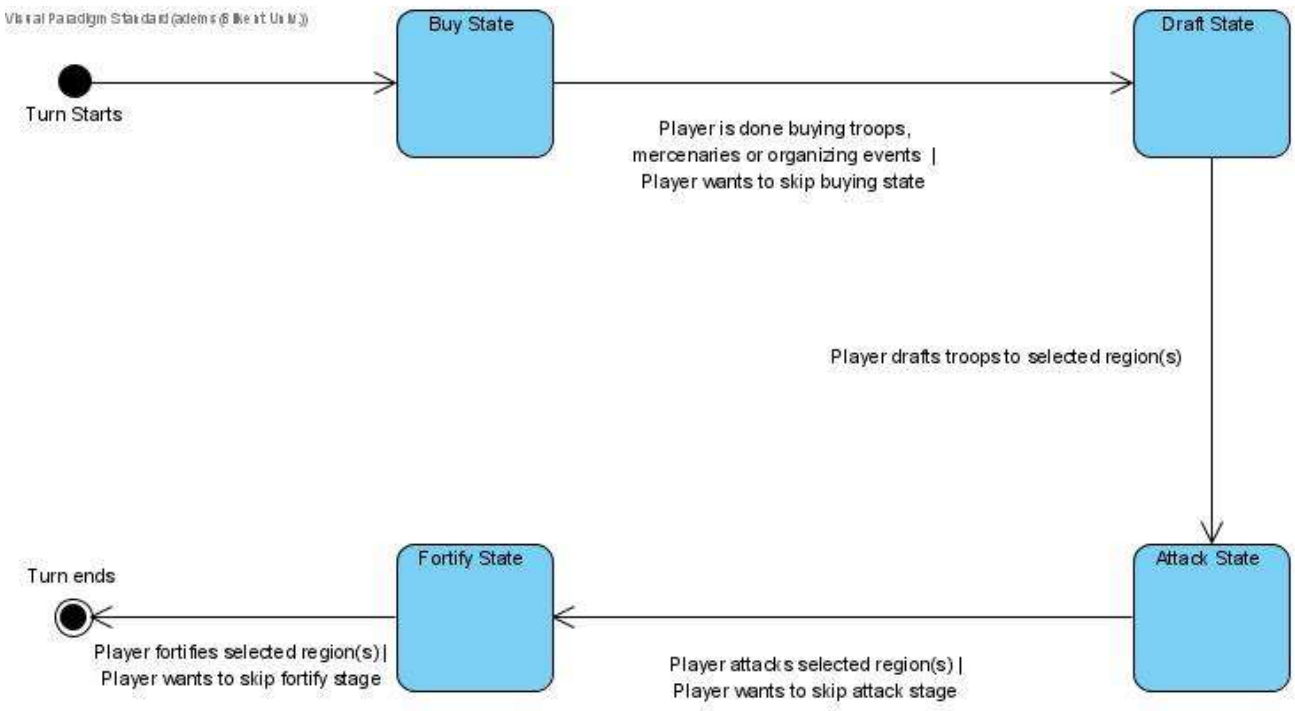

Figure 14: State diagram for turn of a player

The state diagram given above shows how the turn of one player processes. When a player gets the turn, the player enters the buy stage. In this stage, the player can buy troops, mercenaries or organize events. Also, the player may want to skip this stage without doing anything. Then the player comes to the draft stage. In this stage, the player places the troops given to him/her by the game at the beginning of each turn to the regions that she/he wants. The player cannot skip this stage as she/he has to draft the soldiers given to regions. After the draft stage is completed, the player comes to the attack stage. In this stage, the player can choose the area he/she wants to attack and the number of soldiers he/she wants to attack. As long as the player is eligible to attack, the player can attack. There is no constraint on how many times attacking. It only depends on whether the player can attack or not. Also, the player may want to skip this stage without doing anything. After the attack stage is completed, the player comes to the fortify stage. In this stage, the player can fortify soldier(s) from one region to another region. Also, the player may want to skip this stage without doing anything. After this stage, the turn of the player ends.

5.3. Object and Class Model

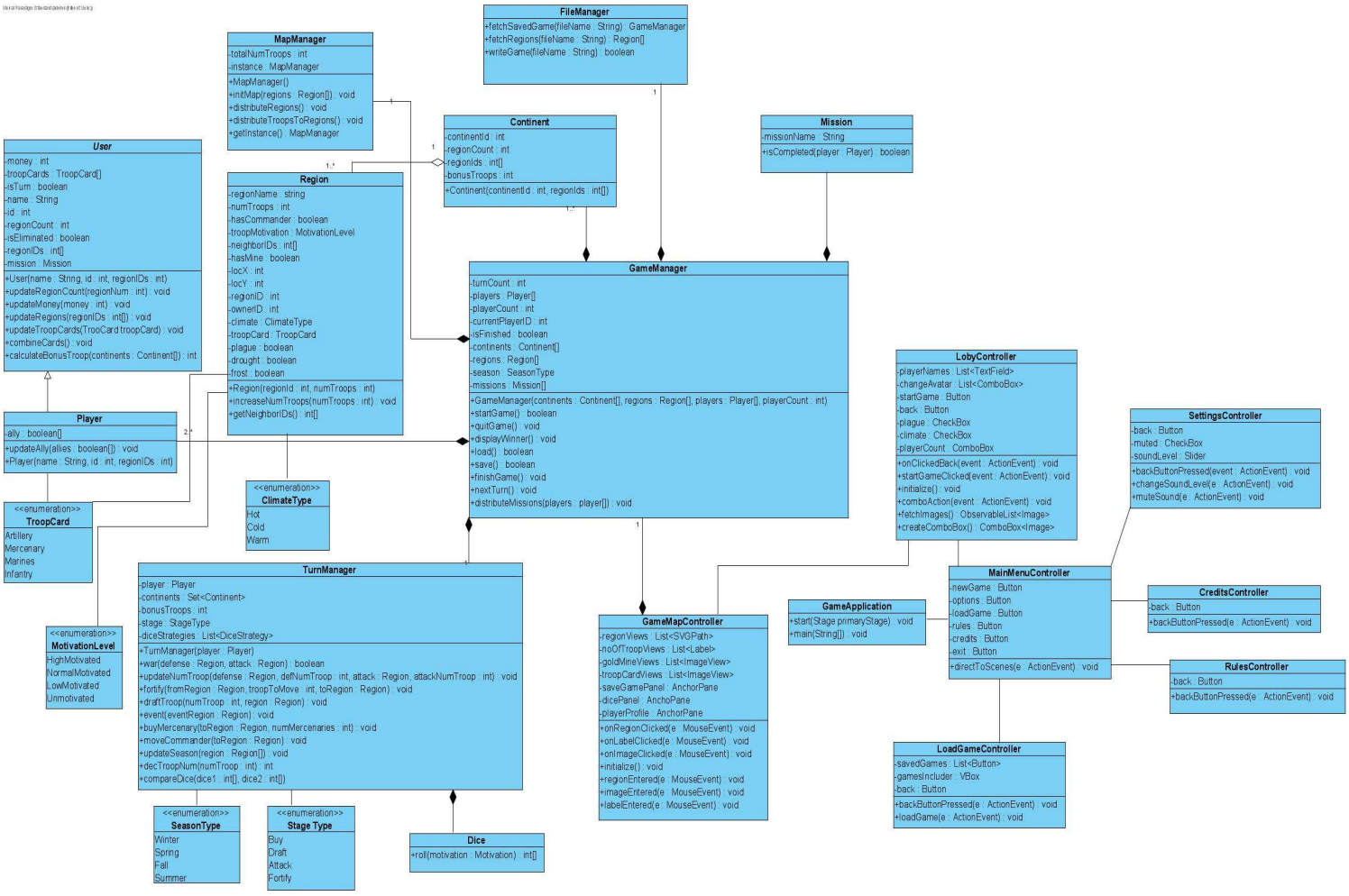

Figure 15: Object and Class Model

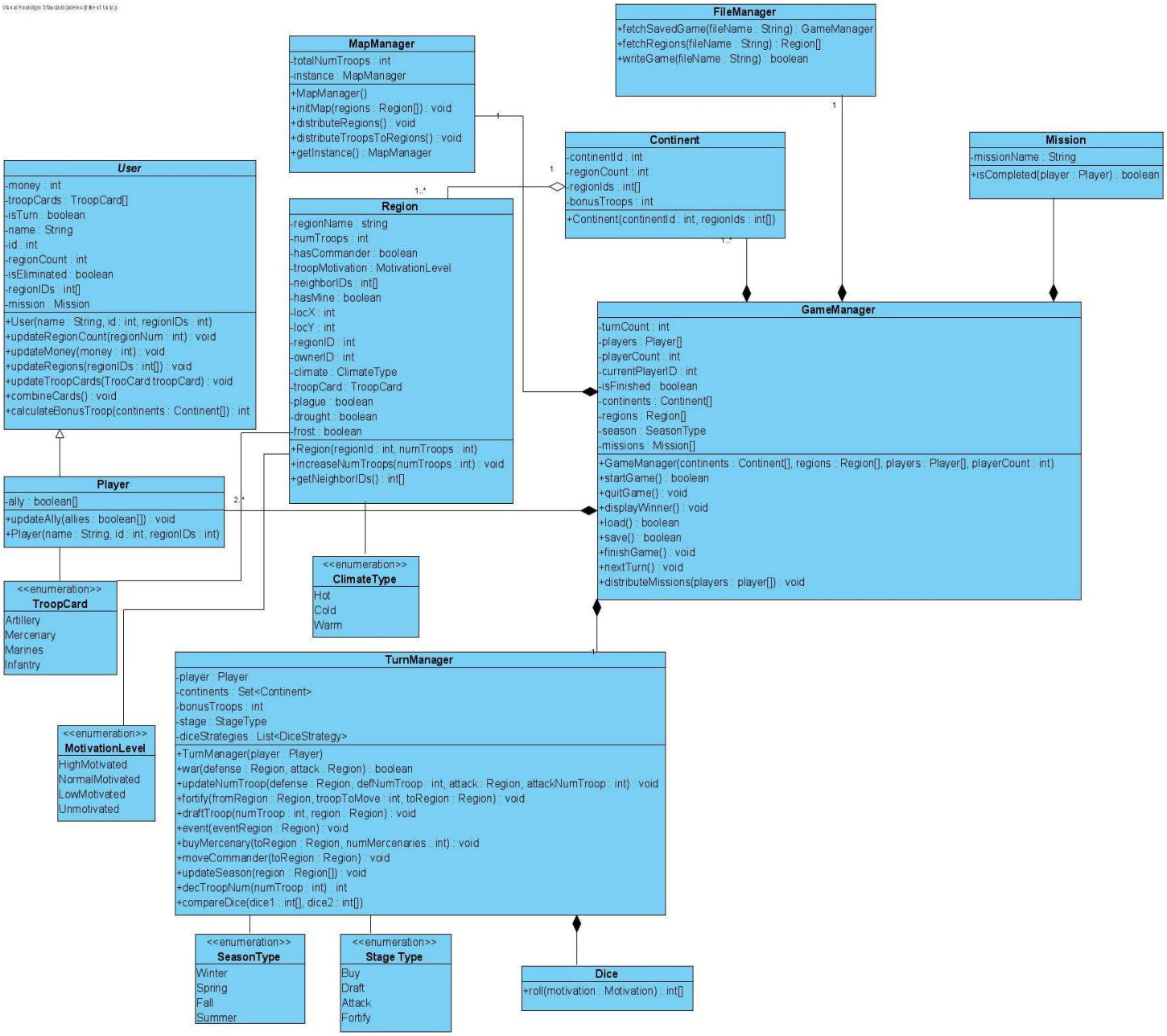

Figure 16: First Part of the Object and Class Model

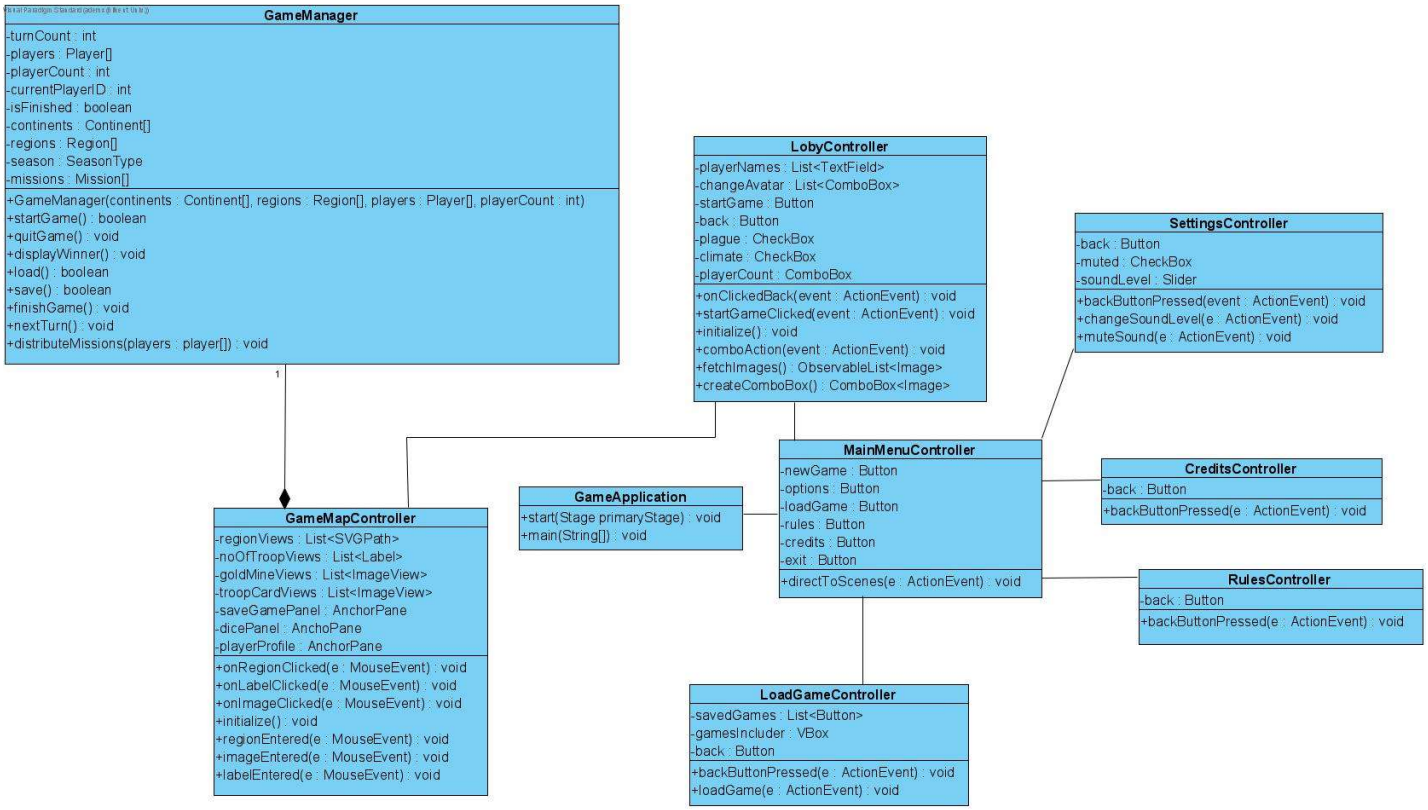

Figure 17: The Second Part of the Object and Class Model

❖ **Controller and Boundary Classes**

LobbyController: This class will contain GUI objects and manage the changes coming from the lobby view and then initializes the game.

GameMapController: This class will contain GUI objects and manage the changes coming from the game map view and then update the game map view.

SettingsController: This class will contain GUI objects and manage the changes coming from the settings view and change the sound settings of the game.

MainMenuController: This class will direct users to other scenes in the game such as load game, rules, options, lobby and credits.

LoadGameController: This class will contain GUI objects for loading a game and enable users to load saved games.

RulesController: 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.

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

❖ **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 a new game is started.

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.

Dice: This class will be responsible for the abstraction of dice rolled while wars take place in the game.

Mission: This class will be responsible for the abstraction of missions assigned to players when the game started.

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

5.4.1. Navigational Paths

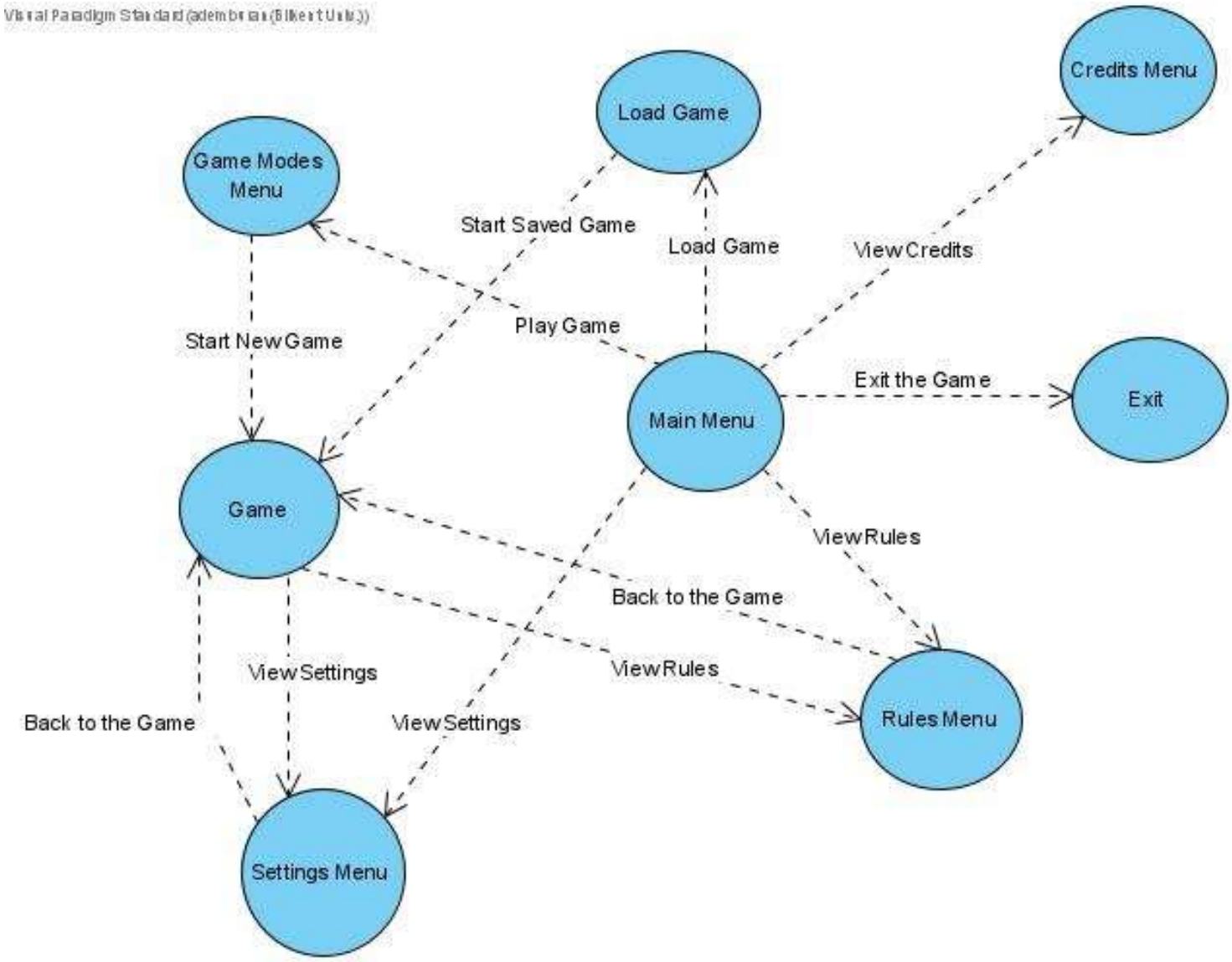

Figure 18: Navigational path of the game.

5.4.2. Screen Mock-Ups

5.4.2.1. Main Menu

Figure 19: 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

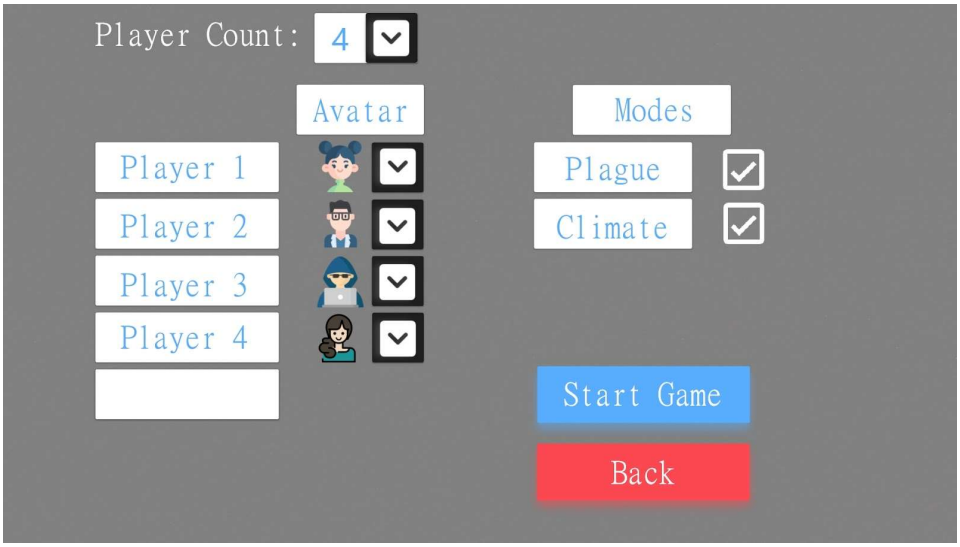

Figure 20: Game Settings is the screen in which players choose avatars and adjust the game modes

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

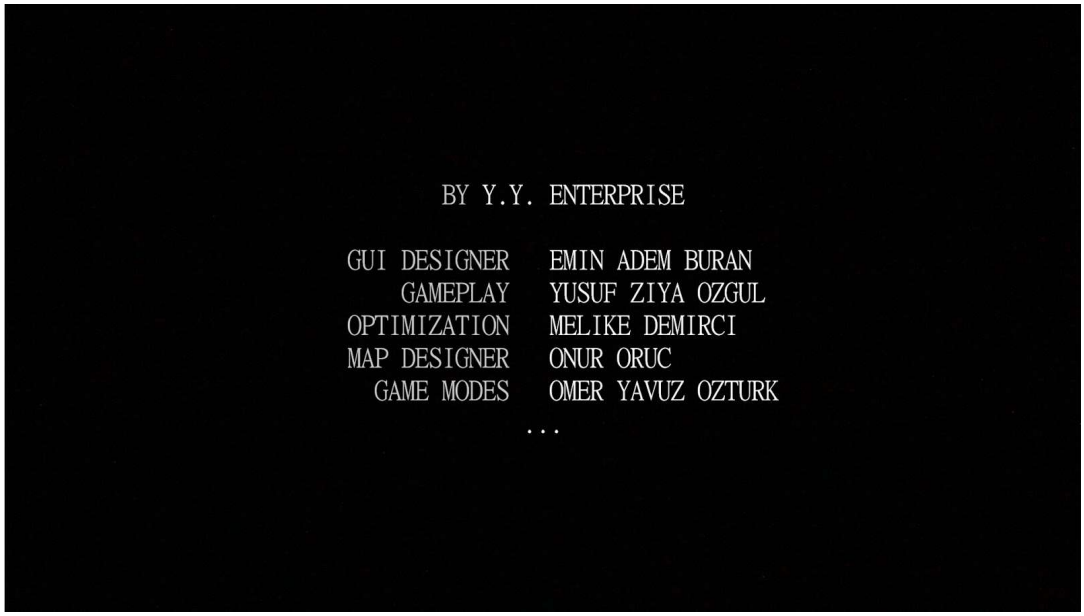

Figure 21: Credits page displaying the game contributors

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

5.4.2.4. Settings

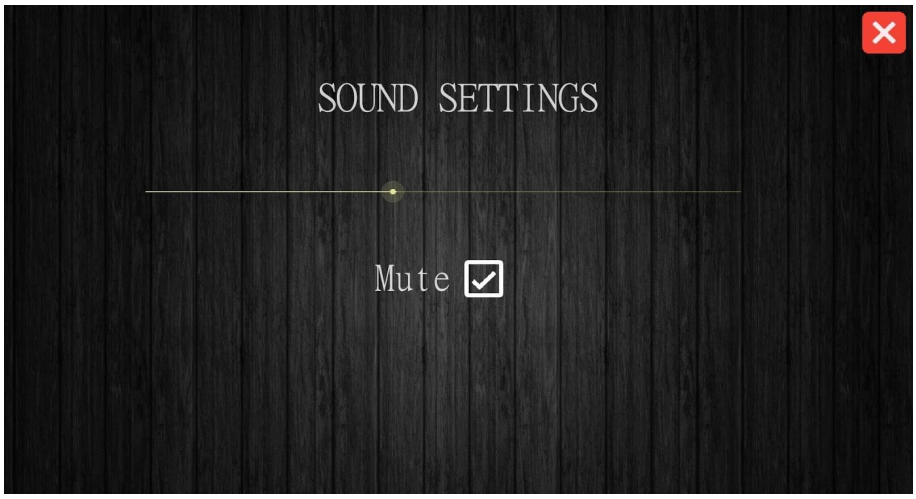

Figure 22: Sound Setting screen navigating the music volume in the game

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

5.4.2.5. Rules

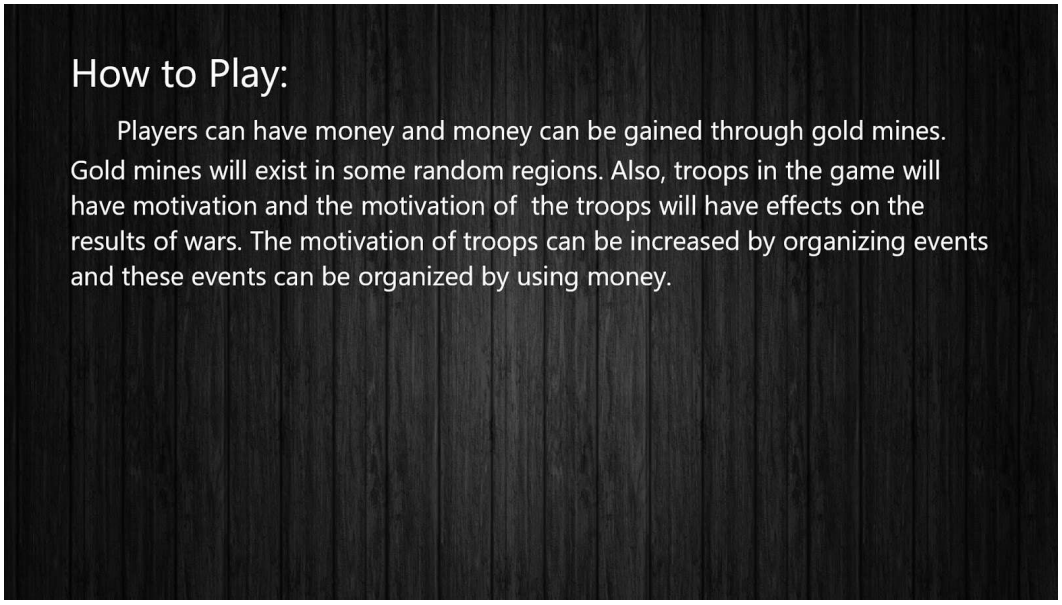

Figure 23: Rules screen displaying the game rules

In the Rules, players can read the game rules.

5.4.2.6. Game Screen

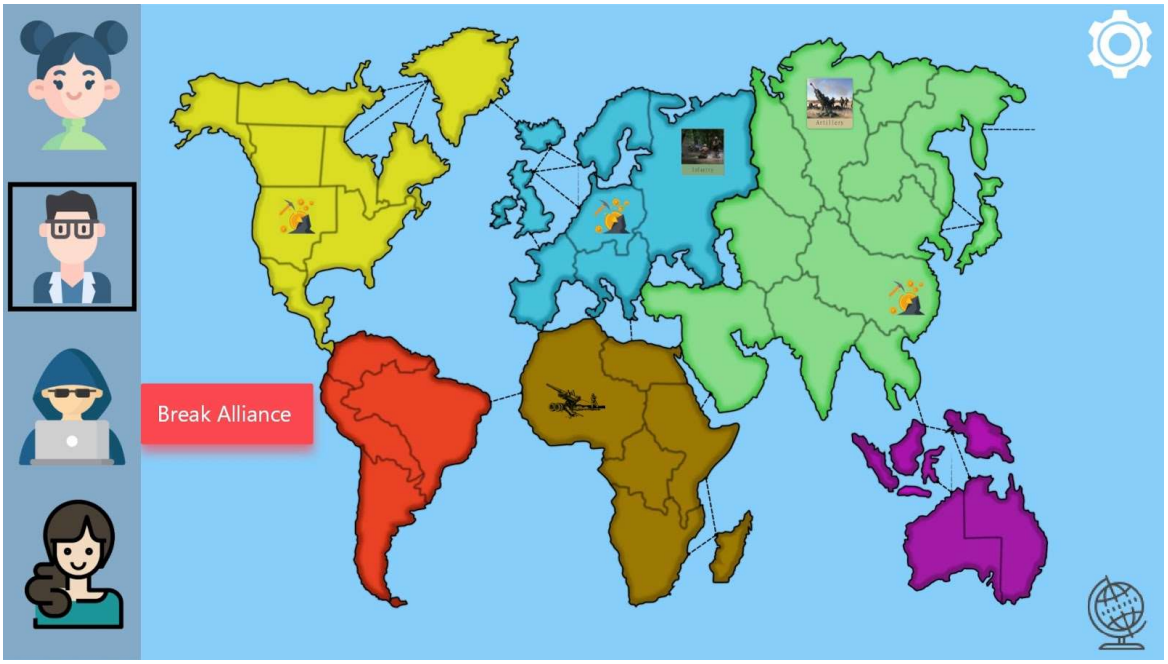

Figure 24: Game Screen displayed when the game has started

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

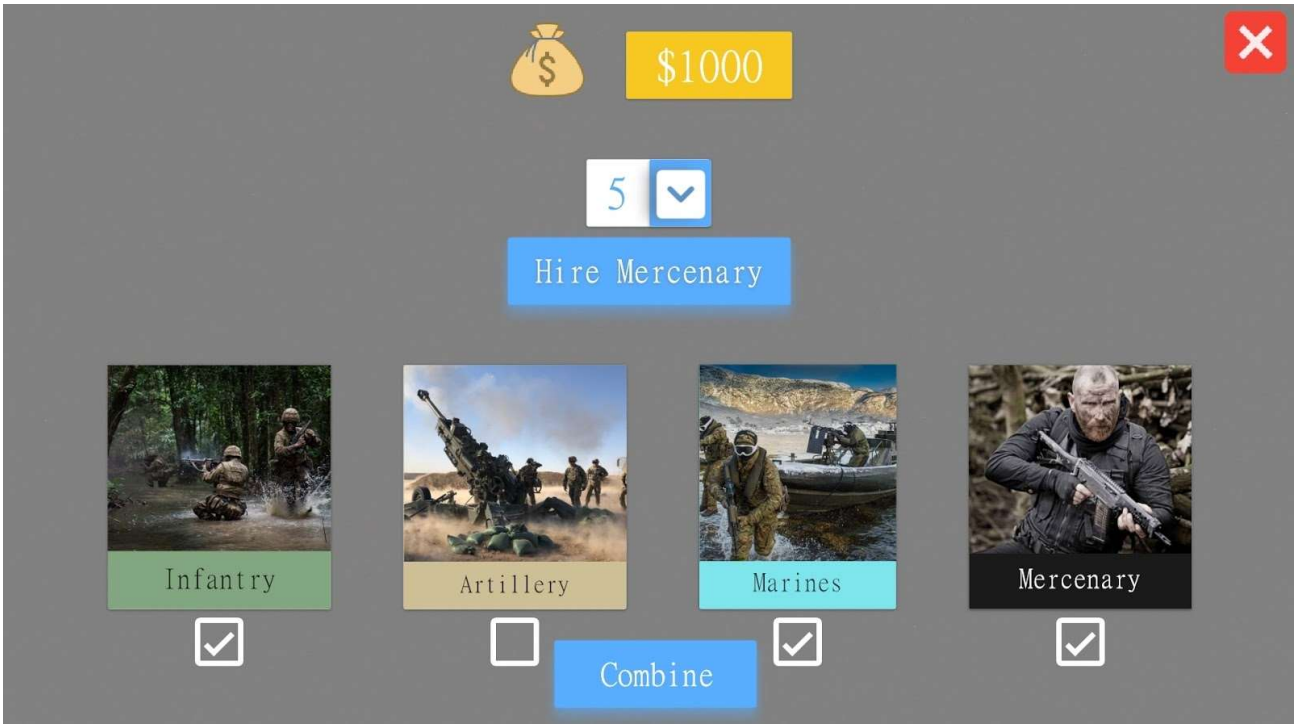

Figure 25: Player profile screen [1], [2], [3], [4]

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. Improvement Summary

When we were writing this report we made some corrections based on the feedback for Analysis Report Iteration 1. While making corrections, we have added new things to this report. First of all, we have revised our object class model and we modified some classes in this model and added some new classes. We have renamed the MapManagement, GameManagement, TurnManagement classes and new names are MapManager, GameManager and TurnManager. We have modified some classes in the Analysis Report Iteration 1 to new controller classes: GameMapController, LobyController, SettingsController, SettingsController, MainMenuController, CreditsController and RuleController. Also, we created a GameApplication class. Another thing we did is that we have revised and renamed the attributes and methods that exist already. We have revised our state diagram for the

turn of the players and we have rebuilt it based on given feedback. In addition to it, we have designed two state diagrams for the regions in terms of attacking ability and motivation level. First one shows how attacking ability changes for any region throughout the game and second one shows how motivation level for any region changes throughout the game. Another improvement is that we have broadened the use case diagram. We have added new use cases for gameplay on the use case diagram in Analysis Report 1 and we have written explanations for them. BreakAlliance, MakeAlliance, FortifyRegion, BuyMercenaryorEvent, DraftTroops, AttackEnemies, TryToCompleteMission, SaveGame, MoveCommander and CreateNewGame are use cases we added. We have updated the extension points of PlayGame and PauseGame use cases. Lastly, we have added missions to our game's functionality. In order to add missions to game, we have added a new class called "Mission" to our class diagram. This class will be responsible for the abstraction of missions assigned to players when the game started.

7. 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.