Bilkent University

Department of Computer Engineering

CS319 Term Project

RISK

Analysis Report Iteration 2

Group 2I

Ahmet Kaan Uğuralp – 21803844

Selcen Kaya – 21801731

Alperen Can – 21601740

Şükrü Can Erçoban – 21601437

Selin Kırmacı – 21802177

Instructor: Eray Tüzün

Teaching Assistant(s): Barış Ardıç, Emre Sülün and Elgun Jabrayilzade

Table of Contents

1. Introduction

Many kings tried to conquer the world and be the overall ruler of the Earth for centuries. Some came close but none could actually achieve this goal. Risk is a board game in which the purpose is to achieve world domination and it allows people to achieve the goal in a fun and mostly friendly environment. Every player starts with the same conditions and equal amount of possessions but only the one with the best strategy can achieve the glorious end.

Our project is an interpretation of this board game in a digital environment as a computer game. The game is enhanced with colorful graphics, easy and fun user interfaces and exciting sound effects. It offers the players to have an amazing time with their “enemies” in a friendly competitive environment.

The goal is to implement this game in such a way that it will have an efficient backend as well as an interesting and user-friendly front end design. New and wonderful features will be added which will enhance the entertainment of the original game. We will use Java language to implement the game with object oriented design principles as all of the team members are comfortable with it because of previous courses and projects.

2. Overview

Risk is a multiplayer 2D game which can be played with 2 to 4 players. Although the original game allows more players, we set a player limit of maximum four in order to provide a faster flowing and a better interface for the game. After starting the executable file, the users will view the main menu where they have the options to start a new game, see how to play the game, view settings, quit or see the credits.

We improved the game by adding new features such as alliance, curse cards, retreat option for the defender and limited time for turns, which will be explained in detail in the following subsections.

2.1 Gameplay and Controls

At the beginning of each game, each player selects unique avatars or emperors, colors and nicknames to represent themselves. Subsequently, instead of manually choosing the territories, the system distributes territories and different troops equally to each player for a faster flowing.

Another feature for faster flowing is that players have a limited time during a turn. If time is up, other players will take turns by order. The players get an extra amount of troops at the start of each turn, depending on the number of regions under their control. Moreover, players are provided additional troops by taking the control of a whole continent, exchanging territory cards, or using the relevant curse card. During a turn a player can attack a territory adjacent to one of their own. The winner of the war is determined by dice roll between defending and attacking players as well as strength of troops on their territories. Alliance is an important feature at the attack phase of the game, which enables players to send troops to support the defense of a territory. After the war, fortify phase begins. Players can move troops from one of their territory to another, in order to enhance the strength of another, possibly more important territory. Afterwards, other players take turns in order. Ultimately, the game ends when a player wins by conquering all territories in the map, or by completing the secret mission depending on the game mode.

The game is controlled by mouse. Players will be able to click on a territory to attack that territory at the attack phase, or to strengthen territories at the fortify phase. Additionally, exchanging territory cards, using curse cards and accepting or refusing alliance offers are performed by clicking the relevant buttons presented on pop-up screens by the mouse.

2.2 Map

The game will be played on a map divided into continents which are then divided into territories. Players can attack enemy territories or move troops to their own territories, if there is a border between these regions. The map consists of six continents, formed by different numbers of territories. Taking control of an entire continent provides additional troops to player at the beginning of every turn. Number of the additional troops obtained varies for different continents depending on their size or their strategic importance.

2.3 Cards

The cards provide specific advantages to players. Cards are evenly distributed to players in the beginning of the game. They can also be obtained during the game with special conditions such as conquering a whole continent or capturing a territory.

2.3.1 Territory Cards

If a player successfully captures a territory, a territory card will be gained regardless of the number of the captured territories in a single turn. There are three different territory cards. A player should collect three proper territory cards to form a set, in order to exchange the set with additional troops at the beginning of the turn. The number of troops to be gained depends on the number of former card exchanges:

- 1st set exchange: provides 4 additional troops
- 2nd set exchange: provides 6 additional troops
- 3rd set exchange: provides 8 additional troops
- 4th set exchange: provides 10 additional troops
- 5th set exchange: provides 12 additional troops
- 6th set exchange: provides 15 additional troops

For more than six exchanges, the number of extra troops provided will be increased by five. For example, if total six set of exchanges have already done by players, the player exchanging the 7th set will receive 20 extra troops.

If a player has 6 or more proper territory cards forming a set, the cards must be exchanged at the start of the turn to lower that number to at most 4, as per the board game rules.

2.3.2 Curse Cards

Curse cards are special rare cards that can be used before the attack phase. It provides advantages such as skipping an opponent's turn or decreasing the number of the opponent's troops during a battle. It might be received by capturing a territory. However, there is a low possibility to obtain one. Four different curse cards existing in the game are explained below.

- **Epidemic**: Decreases opponent's troops during a battle.
- **Civil War**: Skips an opponent's turn.
- **Celebration**: Captures an amount of enemy troops to add them to your army.
- **Damage the Army**: Increases the possibility of taking damage.

2.3.3 Immunity Cards

Immunity cards are special rare cards that can be used before the attack phase. It provides immunity to a territory for a turn. It might be received by capturing a territory. However, there is a low possibility to obtain one.

2.4 Attack Phase

Players may declare a war to another player's territory through their own territories if these regions are sharing a land border or a sea border. The players can also choose to not attack and skip to the fortify phase.

2.4.1 Alliance

When a player declares war, defending player can offer to have an alliance to other players, not including the aggressor. If an alliance offer is accepted, the defending player can receive reinforcement from his ally's soldiers that have not been put into a territory yet, meaning soldiers are taken from other player's hands if there are any.

2.4.2 Capturing a Territory

The result of the war is determined by rolling dice. The aggressor can choose to roll one to three dice while attacking, where the defender can roll maximum two dice. The highest dice of players are compared. If the die of the aggressor is higher, the defender loses a unit strength. Similarly, if the die of the defender is higher, then the aggressor loses a unit strength. If more than two dice are rolled, second highest pair is compared as well as the first highest pair. Additionally, if the attacker rolls one or two more dice than the defender, the dice which cannot be compared are ignored.

The war lasts until there are no more troops to fight in that region or until the aggressor or defender decides to retreat from the war. Moreover, it should be noted that the attacking player can use as many units from its region. However, at least one unit has to stay in its territory, that is, the player cannot fully abandon its region. After a territory war ends, the player whose turn it is can declare war on another territory. Attack phase lasts until the player skips to fortify phase, loses its all units, or time is up.

2.4.3 Abandon a Territory

The defender can retreat if only he is controlling one of the nearby territories. If defender retreats from the war, his troops in the target territory will survive, which later allows him to fall them back to near territories.

2.5 Fortify Phase

Fortify phase begins after the attack phase ends. Player whose turn it is may send troops from one of their territories to another one in order to increase the strength of that region. Players can perform only one fortify move in a turn. When a player completes or skips the fortify phase, the player's turn ends.

2.6 Secret Mission Mode

As an alternative to classic game mode described above, secret mission mode can be played by clicking include secret missions checkbox at the start game menu. Every player will have a secret mission such as conquering a specific continent. Rather than conquering the whole map, completing the secret mission is enough to win the game in this mode.

3. Requirements

The functional and non-functional requirements of the project are described below in separate subheadings.

3.1 Functional Requirements

The functional requirements which are starting a new game, viewing how to play page, changing settings and seeing credits are explained below.

3.1.1 Start New Game

When the program is opened, the initial page displays the main menu. “Start New Game” option can be seen on the top of the menu and when the user clicks this button, the game starts after asking the initial conditions and statements. The user is directed to a page which asks the number of players as well as the names, colors and preferred avatars of the players.

3.1.2 How to Play

In the main menu and the pause menu during a game, there is a “How to Play” option which displays essential information about the game to the user containing rules of the game, controls and game explanations such as win conditions. This option helps first time players of the game as well as people who played other Risk game implementations by clarifying the differences.

3.1.3 Settings

The user can see the “Settings” option in the main menu and pause menu. This option allows the user to customize sound and music options which can either be adjusted or muted.

3.1.4 Credits

This option can be accessed from the main menu and when the user clicks this button, they see a screen displaying information, such as names, about the developers of the game.

3.2 Non-Functional Requirements

The non-functional requirements are explained in the following subheadings including interface, performance, maintenance and extendibility as well as usability.

3.2.1 Interface

The user interface is one of the most important features of a game since it can have critical effects on the users even with a first impression. It determines users’ involvement and can also improve continuity and cohesion of the game. Thus, to make a user-friendly interface, we will make all the buttons and options distinct and obvious, so accessibility and visibility will improve. In addition, we will design our graphics carefully in terms of intelligibility and tonality. These features make our game more attractive and captivating.

3.2.2 Performance

In the game, we will focus on response time and interface transition time since slow performance in terms of reaction time may push users to quit because of their exhaustion and annoyance. Moreover, to get more performance and for compatibility with every computer, the

system requirements will be at the lowest possible level. We also add some features which do not affect the performance such as sound and music, in order to get the highest performance along with the right ambiance and pleasantness.

3.2.3 Extendibility and Maintenance

The class models were designed to easily make adjustments and add new features for any improvement we might want to make in the future. For example, a feature we are planning to add involves certain advantages and disadvantages given to each player depending on their choices at the beginning of the game such as the capital and the emperor they chose. Also, object oriented programming concepts will be used in order to make extendibility convenient. For maintenance, we will pay attention to the implementation and design of our codes since it should be clearly understood by others. In addition, we will try to clear any ambiguous sections by writing coherent comments. Moreover, the game can be improved with artificial intelligence (AI) since with AI, a player can play the game against the computer.

3.2.4 Additional Requirement: Usability

Our purpose was to make the game uncomplicated, that is, effortless to understand the game rules and easy to play even for new starters. Therefore we have put a detailed How to Play section to main menu, in addition to user-friendly interfaces.

4. System Models

The use-case model and descriptions, object and class diagram, sequence diagrams, activity diagrams and user interface mock-ups are given in their respective subheading below.

4.1 Use Case Model

The use case diagram is given and the descriptions of five use-cases are given below it in their respective subheadings.

4.1.1 Use Case Diagram

The use case diagram is given in the Figure 1 below.

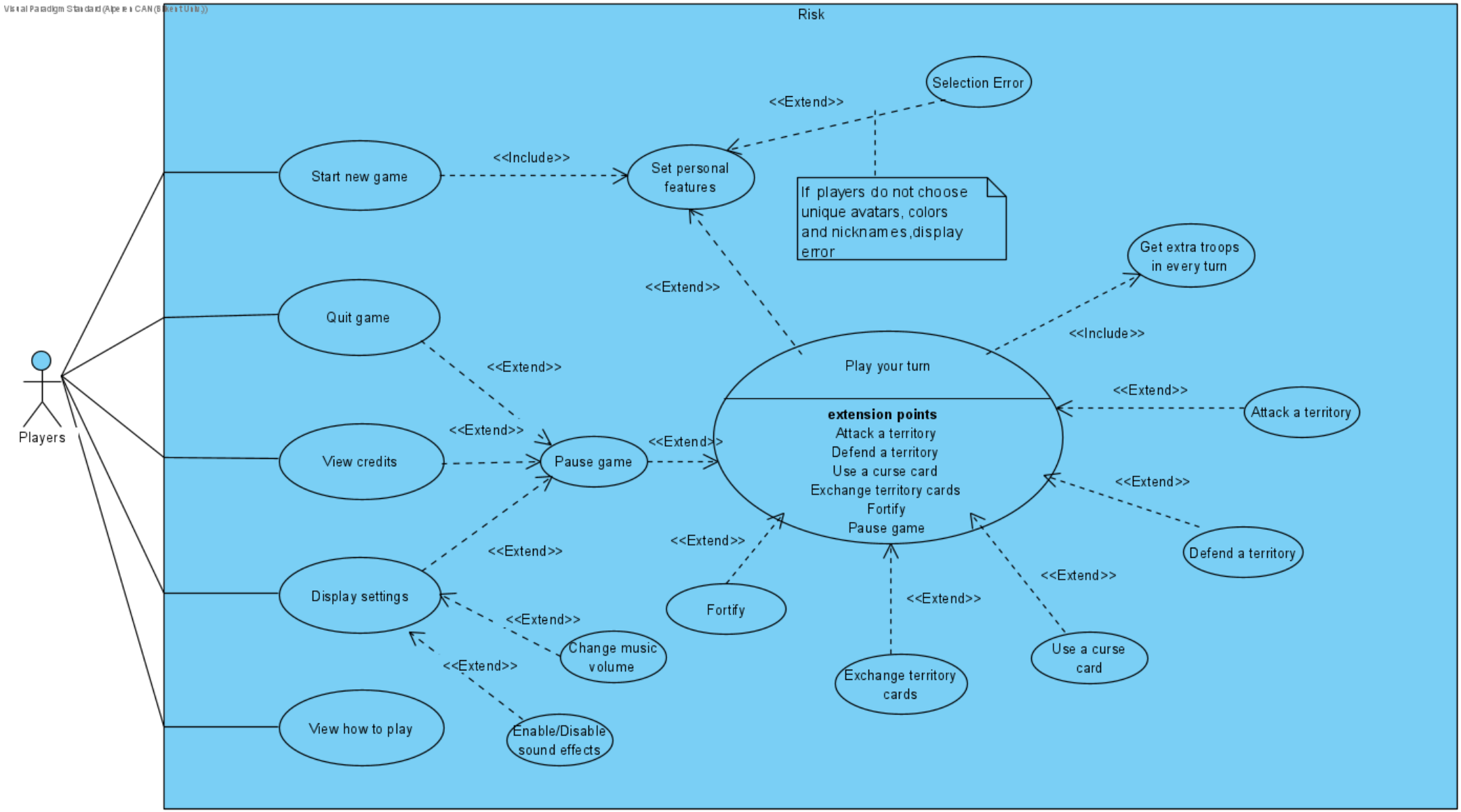


Figure 1: Use case diagram of the project.

4.1.2 Use Case Descriptions

The five use cases which are named start new game, how to play, settings, credits and pause are described below.

4.1.2.1 Use Case Name: Start New Game

Participating Actor: Players

Stakeholders and Interests: Players want to start the game.

Entry Conditions:

- Players clicks *Start New Game* button.
- Player selects the number of players.
- Players choose avatars, nicknames and colors to represent them.
- Players clicks *Continue* button.

Exit Conditions:

- Player clicks *Quit* button from pause menu, OR
- Player conquers all territories in the map to win the game.

Main Flow of Events:

1. Player select the number of players.
2. Players enter nicknames and choose their avatars and colors.
3. Player starts game.
4. System sets the distribution of the troops and territories for each player.
5. Player whose turn it is adds new troops to their territory(s) at the beginning of the turn.
6. Player may attack other players' territories.
7. Player may move troops to their other territories.
8. Other players take turns.
9. A Player wins the game by conquering all territories or by completing the secret mission, depending on the game mode.
10. The winner is demonstrated on the screen at the end of the game.

Alternative Flows of Events:

2.1 Players choose unique avatars, nicknames and colors.

a. Player starts game.

2.2 Players do not choose unique avatars, nicknames and colors.

a. Warning message is displayed in order to urge players to choose unique avatars, nicknames and colors.

6.1 Player attacks to other players' territories.

6.1.2 Attacker and defender clash by throwing dice.

a. Aggressor conquers the territory.

b. Aggressor fails to conquer the territory.

6.2 Player does not attack other players' territories.

a. Player skips the attack phase to begin fortify phase.

7.1 Player moves troops to their other territories.

a. Player increases the strength of their other territories.

7.2 Player does not move troops to their other territories.

a. Player skips the fortify phase to let other players take turn.

4.1.2.2 Use Case Name: How to Play

Participating actor: Player

Stakeholders and Interests: Player wants to be informed about the game rules.

Entry Conditions:

- Player clicks *How to Play* button from main menu, OR

- Player clicks *How to Play* button from pause menu.

Exit Condition:

- Player clicks *Back* button.

Main Flow of Events:

1. Player clicks *How to Play* button.
2. System displays how to play screen.

4.1.2.3 Use Case Name: Settings

Participating actor: Player

Stakeholders and Interests: Player wants to change the music volume and enable/disable sound effects.

Entry Conditions:

- Player clicks *Settings* button from pause menu.

Exit Conditions:

- Player clicks *Back* button.

Main Flow of Events:

1. Player clicks *Settings* button.
2. Player adjusts the volume, alters the status of sound effect checkbox.
3. Player clicks *Save* button.
4. Music and sound properties will be changed.

Alternative Flow of Events:

2.1 Player does not change the music and sound options.

a. Music and sound properties remain same.

3.1 Player returns menu without clicking *Save* button.

a. Music and sound properties remain same.

4.1.2.4 Use Case Name: Credits

Participating actor: Player

Stakeholders and Interests: Player wants to know the developers of the game.

Entry Condition:

- Player clicks *Credits* button from main menu.

Exit Condition:

- Player clicks *Back* button.

Main Flow of Events:

1. Player clicks *Credits* button.
2. The contributors of the game are shown on the screen.

4.1.2.5 Use Case Name: Pause

Participating actor: Player

Stakeholders and Interests: Player wants to pause the game.

Pre-condition:

- The game must have started.

Entry Condition:

- Player clicks *Pause* button.

Exit Conditions:

- Player clicks *Continue* button, OR
- Player clicks *Quit* button.

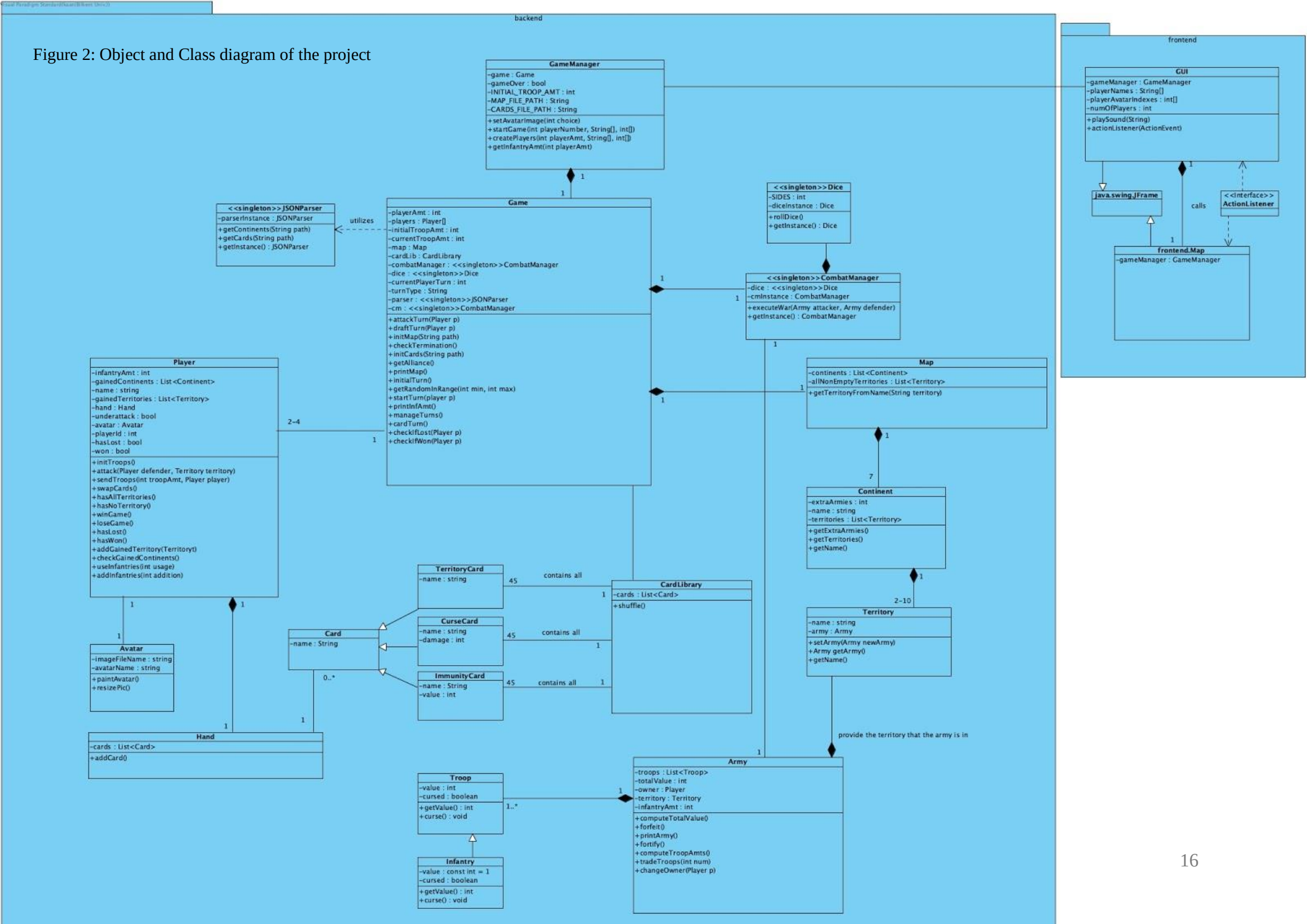
Main Flow of Events:

1. Player clicks Pause button.
2. The game is paused.

4.2 Object and Class Diagram

The object and class diagram is given in the figure (Figure 2) and descriptions of each class is located under it.

Figure 2: Object and Class diagram of the project



The descriptions and the contents of the classes given in the diagram above (Figure 2) are as follows:

4.2.1 Game class

Attributes:

- private final int playerAmt: The amount of players
- private Player[] players: Contains player of the game
- private final Dice dice: the rolled dice
- private int currentTroopAmt: The current amount of troops of a player
- private CardLibrary cardLib: The collection of the cards
- private Map map: the map object of the game
- private final JSONParser parser:
- private final CombatManager cm: The object of CombatManager class to determine the winning side of a war
- private int initialTroopAmt: The initial number of troops of a player
- private final int CONTINENT_AMT: The number of continents

Methods:

- private void initMap(String path): Create map by parsing the data from a JSON file
- public void printMap(): prints the game map
- private void initCards(String path): Create cards by parsing the data from a JSON file
- public void initialTurn(): When the game starts, each player gets their continents and army
- private static int getRandomNumberInRange(int min, int max): return random number between min-max both inclusive
- private void startTurn(Player p): At the beginning of each turn, the player gets their infantries
- private void printInfAmt(): It prints number of infantries of a player
- private void manageTurns(): Manages the each turn to continuity with each features
- private void draftTurn(Player p): Replacing the player' troops inside his territories
- private void attackTurn(Player p): Determines which will win the declared war.
- private void cardTurn(): Checks the conditions where a player receives a card and the player determines whether he will use card or not.

- private void checkIfLosed(Player p): Checks the losing condition for a player.
- private void checkIfWon(Player p) : Checks the winning condition for a player.
- private void checkIfWon(Player p) : Checks the ending condition of the game.

4.2.2 Player class

Attributes:

- private int infantryAmt: The number of infantry for a player
- private final String name: The name of the player
- private final Avatar avatar: The avatar object which is chosen by player
- private Hand hand: Contains the cards of the player
- private boolean underAttack: boolean type to check whether the player is under attack or not
- private ArrayList<Continent> gainedContinents: The list of player's gained continents
- private ArrayList<Territory> gainedTerritories: The list of player's gained territories
- private final int playerId: The Id of the player
- private boolean hasLost: Checks whether the player is lost or not
- private boolean won: Checks whether the player has won or not

Methods:

- public void winGame() : Controls whether the player win the game or not
- public void loseGame() : Controls whether the player lose the game or not
- public ArrayList<Continent> getGainedContinents() : Return the gained continents list
- public int getInfantryAmt() : returns the number of infantries of a player
- public ArrayList<Territory> getGainedTerritories(): Return the gained continents list
- public void addGainedTerritory(Territory t): Adds the gained territorry from the war to the gained territory list
- public void checkGainedContinents() : Checks the continent whether it is gained earlier or not
- public boolean hasNoTerritory(): Sets the number of territories to zero for a player
- public boolean hasAllTerritories(): Sets the number of territories to six which is the maximum for a player

- public void useInfantries(int usage): Decrements the number of infantries by the usage number
- public void addInfantries(int addition) : Increments the number of infantries by the addition number
- void sendTroops(int troopAmt, Player player): Sends the number of troops to opponent player for a war
- void attack(Player defender, Territory territory) : To declare war to another player for specific territory

4.2.3 JSONParser class

Methods:

- public ArrayList<Continent> getContinents(String path): Takes the JSON file's path as the parameter. Creates and returns a continents list by parsing the data from a JSON file.
- public ArrayList<Card> getCards(String path) : Takes the JSON file's path as the parameter. Creates and returns a cards list by parsing the data from a JSON file named path

4.2.4 Map class

Attributes:

- private final ArrayList<Continent> continents: The list of the continents in the map

Methods:

- public ArrayList<Territory> getAllNonEmptyTerritories() : Get all territories which has an army in it
- public Territory getTerritoryFromName(String territory) : Returns the territory with the corresponding name, returns null if not found

4.2.5 CombatManager class

Attributes:

- private final Dice dice: The dice object to be rolled

Methods:

- public void executeWar(Army attacker, Army defender): Execute war between two armies and update the armies and territory owner according to winner

4.2.6 Army class

Attributes:

- private ArrayList<Troop> troops: The list of troops which the army has
- private int totalValue: Total value of all of the troops in this army
- private Player owner: Owner of the army
- private Territory territory: Territory where the army is in
- private int infantryAmt: The number of infantries the army has
- private int calvaryAmt: The number of calvary the army has
- private int artilleryAmt: The number of artillery the army has

Methods:

- private void computeTotalValue(ArrayList<Troop> troops): Return total value of the army in terms of infantries
- public ArrayList<Troop> getTroops(): Return the troops from the list
- public Player getOwner(): Gets the owner of the army
- public int getTotalValue(): Gets the total troops in the army
- public void forfeit(): Reduces one infantry from the army
- public void fortify(ArrayList<Troop> newTroops): Increase the amount of soldiers
- private void computeTroopAmts(ArrayList<Troop> troops): Computes the total number of troops in an army
- public int getInfantryAmt(): Gets the number of infantries in a army
- public int getArtilleryAmt(): Gets the number of artillery in a army
- public int getCalvaryAmt(): Gets the number of calvary in a army

- public void tradeTroops(int num): Player can trade 5 infantries for one calvary and 10 infantries for one artillery

4.2.7 Dice class

Attributes:

- private static final int SIDES: It represents the number sides of the dice

Methods:

- public int rollDice(): Rolls a dice randomly

4.2.8 Troop class

Attributes:

- protected int value: The value of the troop

Methods:

- public int getValue(): Gets the value of the troop

4.2.9 Infantry class

Attributes:

- constant int value = 1: The value of Infantry

4.2.10 Avatar class

Attributes:

- private final String imageFileName: Contains the avatar image path.

Methods:

- public void paintAvatar(): Prints and draws the avatar
- public void resizePic(): Resize the avatar image

4.2.11 Continent class

Attributes:

- private final int extraArmies: The armies are provided by continent for each turn
- private final String name: The continent name
- private final ArrayList<Territory> territories: The list of territories which are belongs to the continent

Methods:

- public int getExtraArmies(): Gets the extra army number
- public ArrayList<Territory> getTerritories(): Return the list of territories which continent has
- public String getName(): Return the name of the continent

4.2.12 Territory class

Attributes:

- private final String name: The name of the territory
- private Army army: The collection of armies which are placed in the territory

Methods:

- public void setArmy(Army newArmy): Sets the army with the new after the war to update the condition
- public Army getArmy(): Gets the army in the territory
- public String getName(): Gets the name of the territory

4.2.13 Hand class

Attributes:

- private ArrayList<Card> cards: The list of cards which a player has

Methods:

- public void addCard(Card card) : Add card to the list which a player has

4.2.14 Card class

Attributes:

- private final String name: The card name
- private final Troop troopType: To determine which type of troops can use the card

4.2.15 CurseCard class

Attributes:

- private final int damage: To use in wars, determine the amount of damage to opponent

4.2.16 TerritoryCard class

Attributes:

- private final String territory: To understand the card which territory it belongs

4.2.17 ImmunityCard class

Attributes:

- private final int value: Contains the value of the card (for how many turns the player will be immune).

4.2.18 CardLibrary class

Attributes:

- private ArrayList<Card> cards: The list of card in the library

Methods:

- private void shuffle(): At the beginning of the game, mix the deck

4.2.19 GameManager class

Attributes:

- private Game game: Creating a game object

- private boolean gameOver: boolean type to check the game is finished
- private final int INITIAL_TROOP_AMT: At the beginning, destribute each player equal number of troops
- private final String MAP_FILE_PATH: To get map from the file
- private final String CARDS_FILE_PATH: To get cards from the file

Methods:

- public void startGame(int playerNumber,String[] playernames, int[]playerAvatars): Starts the game creating each player and give the initial statements
- private Player[] createPlayers(int playerAmt,String[] playernames, int[]playerAvatars) : Get the names and avatar pics of the users and create the players accordingly
- private int getInfantryAmt(int playerAmt) :return the number of infantries for each player based on the number of players
- private String setAvatarImage(int choice): returns the path of the avatar image with the given choice

4.2.20 GUI class

This class contains the main menu and game start panel which contains the fields for player names, colors and avatars.

Attributes:

- private GameManager gameManager: Creates a gameManager instance to set up the Game by taking in the player properties.
- private String[] playerNames
- private int[] playerAvatarIndexes
- int numOfPlayers

4.2.21 Map class

This class contains the UI elements for game map and interactions between players and the map. Map class gets created from the GUI class after the players fill their required fields and click start game button.

Attribute:

- private GameManager gameManager: GUI class passes this attribute to the Map class.

4.3 Sequence Diagrams

The sequence diagrams that refine the use case descriptions are given in subheading below. Some sequence diagrams and extensive method parameters are left out for readability and simplicity.

4.3.1 Start New Game

The diagram for this scenario is given in the figure below (Figure 3). The GUI and some method parameters class are left out for ease of read, as with those the diagram would be too complicated. The user is in the main menu and clicks the “Start New Game” button which starts the game after asking for number of players, avatar and more. The player can choose to enter these values or use the preset values to start the game, or click on the “Back” button to go back to the main menu. After the user continues, the GameManager class calls the startGame() function and sets the gameOver Boolean to false. This class also initializes each player with a player array and sends it to the Game class. The Game class initializes cards and the map via initMap() and initCards() functions. After all initializations are over, the initialTurn() method loads all the armies, territories and cards for each player and randomly determines the order of players. Then, players start to play their turns. After each turn, the Game class checks whether the game is over by the Boolean gameOver returned from playing a turn diagram. If the game is finished, the GameManager class ends the game, otherwise the nextTurn() function is called by the Game class which goes on to playing another turn.

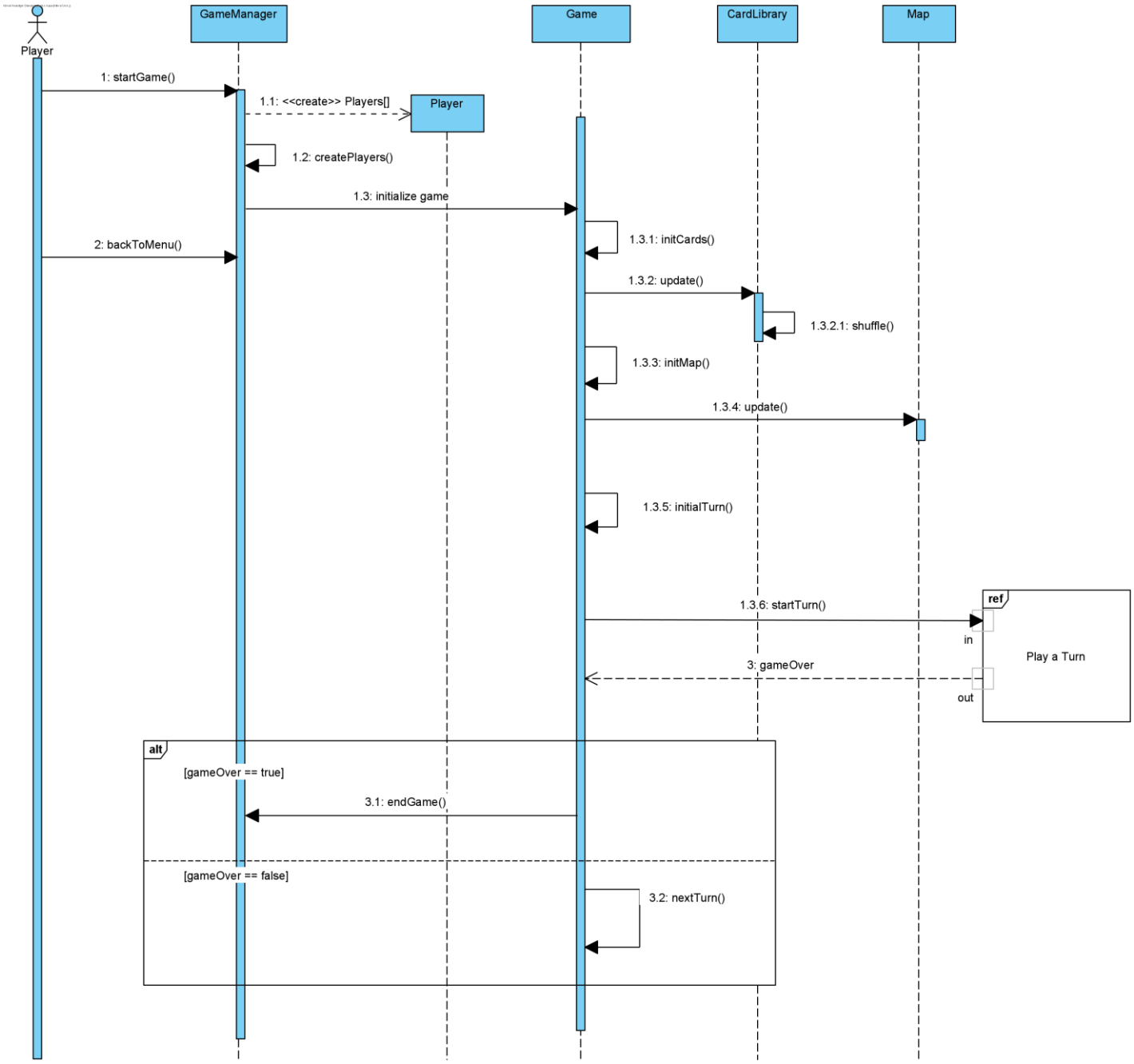


Figure 3: Sequence diagram for Start New Game scenario.

4.3.2 Attacking during a Turn

There are many options a player can choose during their turn such as drafting, exchanging cards, attacking and more. The attacking option is given in a sequence diagram in Figure 4. The player clicks the “Attack” button on their screen during their turn. The attack(territory) function is called with the territory the attacker chose as the target. The CombatManager class calls the executeWar(attacker, defender) function which handles all battles. Both the attacker and the defender, who is the owner of the target territory, roll dice and

the winner of the dice roll is chosen by the CombatManager class. this class updates the armies and the target territory depending on the outcome of the war.

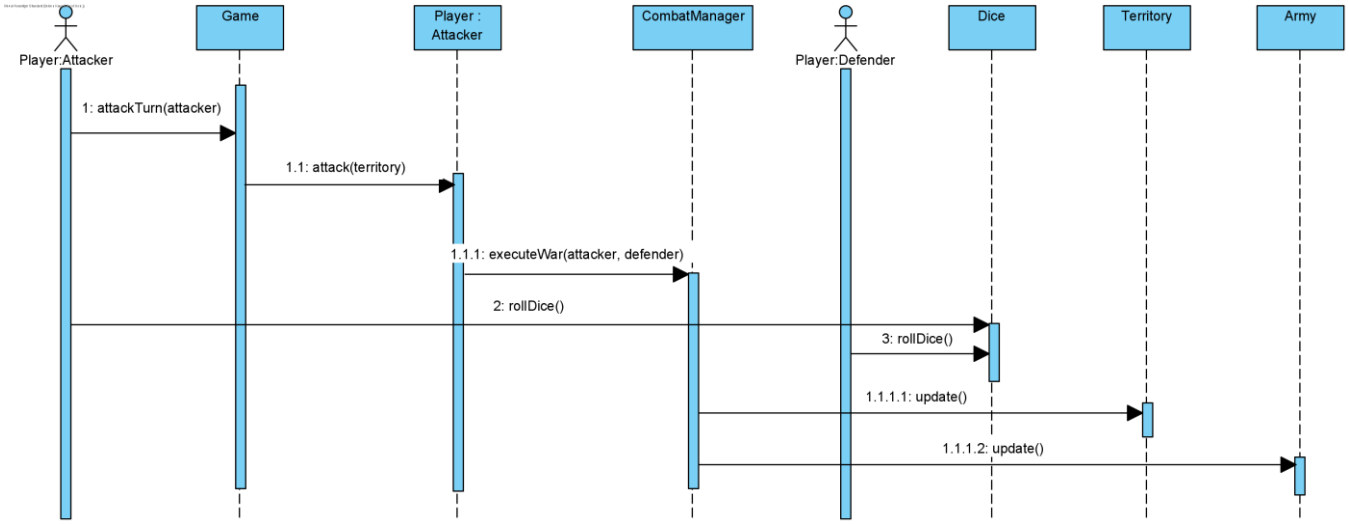


Figure 4: Sequence diagram for Attacking during a Turn scenario.

4.3.3 How to Play

The diagram of this scenario is given in the figure below (Figure 5). The user is in the main menu or in the pause menu during a game and clicks the how to play button. A page showing the description and the mechanics of the game is shown when the “How to Play” button is clicked through the GUI class. The user can come back to the menu by pressing the “Back” button. An almost-identical diagram for Credits scenario is left out for simplicity.

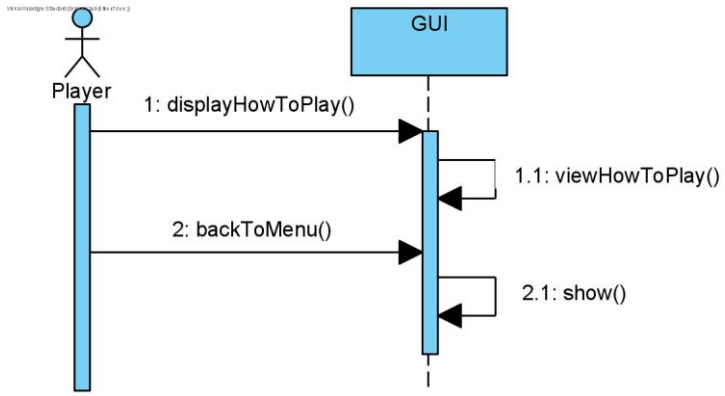


Figure 5: Sequence diagram for How to Play scenario.

4.3.4 Settings

The diagram for this scenario is given in Figure 6 below. The user is in the main menu or the pause menu during a game and clicks on the “Settings” button. The settings for the game which include sound and music are displayed by the viewSettings() function in the GUI class. The user can choose to adjust the volume of sound and music or mute them and click the “Save” button to save their settings, which are updated if saved. The user can come back to the menu by pressing the “Back” button. A similar diagram for Pause scenario is left out for simplicity.

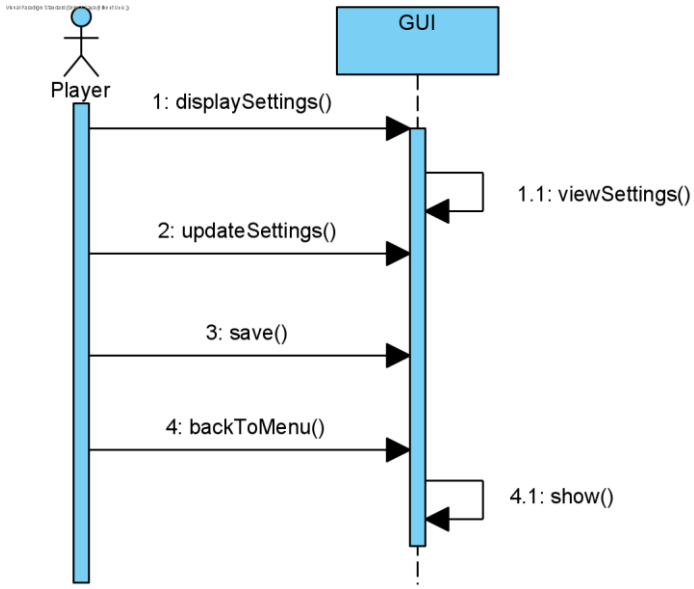


Figure 6: Sequence diagram for Settings scenario.

4.4 Activity Diagram

The activity diagram representing the actions of a player in a turn is given in the figure below (Figure 7).

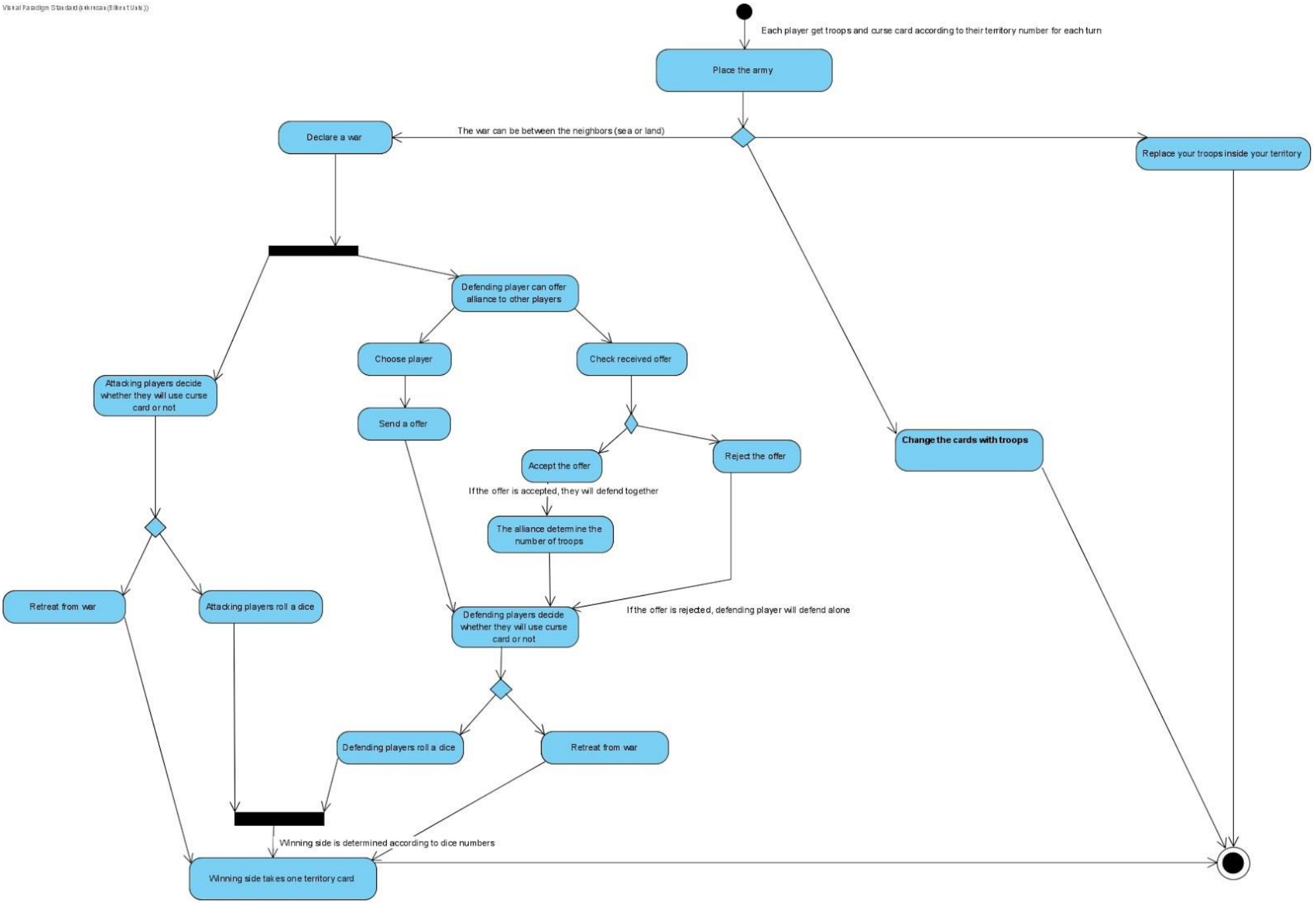


Figure 7: Activity diagram of a turn.

The turn starts with placing the new army which is given to the player by the system according to the number of territories. The player can put their troops in any territory which they own. Then, the player has three alternatives to continue their turn.

The first action is that a player can declare a war on a territory owned by another player, but to declare a war, the opponent's territory must have a border to the player's territory, and the border can be either land or sea. In a turn, a player can declare only one war. After declaring a war, the defending player can request help from other players, apart from the attacker, by asking for an alliance. The defender can send the request to other players by simply choosing a player to offer alliance to. After sending the offer the receiving player can either accept or reject the request. For war preparation, the attacker will decide the number of troops which are sent to war, the number of troops can be at most 3. This operation is also done by the defending side, but they can assign maximum two troops per defender. Then, each side decides whether they will use a curse card or not. The curse cards affect progress of the war since these cards provide advantages to war, but these cards are limited for every player. When every factor is released, attacking, and defending sides have a chance to retreat. The preparations have been made for the war and to determine which side will win, each war participant rolls dice, and the number of dice is related to the number of troops. After this operation, the dice results are compared, and the winning side is determined. Then, the winning side gains one territory card. After these actions, the turn is finished for this player and the next player continues the game.

The second action is replacing your troops inside your territories. This action is important for developing strategies. This operation starts with choosing a territory which troops will be taken from. Then, the player should decide the number of troops to be sent. However, the player cannot leave any territory which they own empty, so at least one troop has to stay. After determining the number of troops, the destination territory should be chosen. After these actions, the turn is finished for this player and the next player continues the game.

The third action is changing the cards in the hand with troops. The number of troops the player can get after trading depends on how many times they have traded in the past and these specific amount of troops are explained in the report above. After this, the turn is finished and the next player continues the game.

4.5 State Diagram

The state diagram of a turn is given in Figure 8 below.

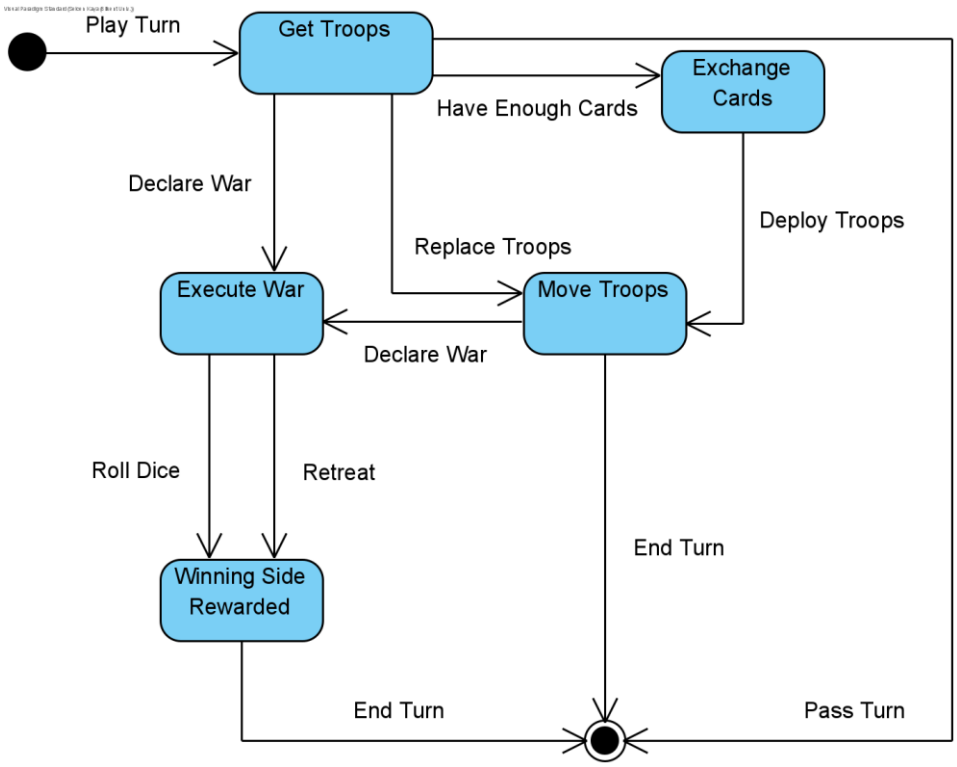


Figure 8: State diagram of a turn.

The turn of a player starts by getting more troops from the game, the number of which is 3. After this, there are 3 paths a player can follow. The player then can choose to move their troops on the map by replacing them. The player can exchange the cards they have for a certain number of troops and deploy them, which also enables the player to move the troops in their possession. Another path allows the player to completely skip their turn, and end it. The last option is to execute a war which can be done after getting troops or moving troops. There is two ways a war can end, which is either by a dice roll or when one side retreats. After the war, the turn is over.

4.6 User Interface Screen Mock-ups

The mock-ups of the user interface screens are shown below. These mock-ups demonstrate the general look of the game, which will most probably change as the game is developed

however the basics will be the same. The components used in the making of these mock-ups were found on the internet which include the map [1], the background of the menus [2], the avatar icons [3] [4], the pop-up menus and card icons [5].

4.6.1 Main Menu

Main menu, presented in Figure 9, gives the player the options to start a new game, learn how to player the game, quit the game, go to settings of the game and learn the developers of the game.

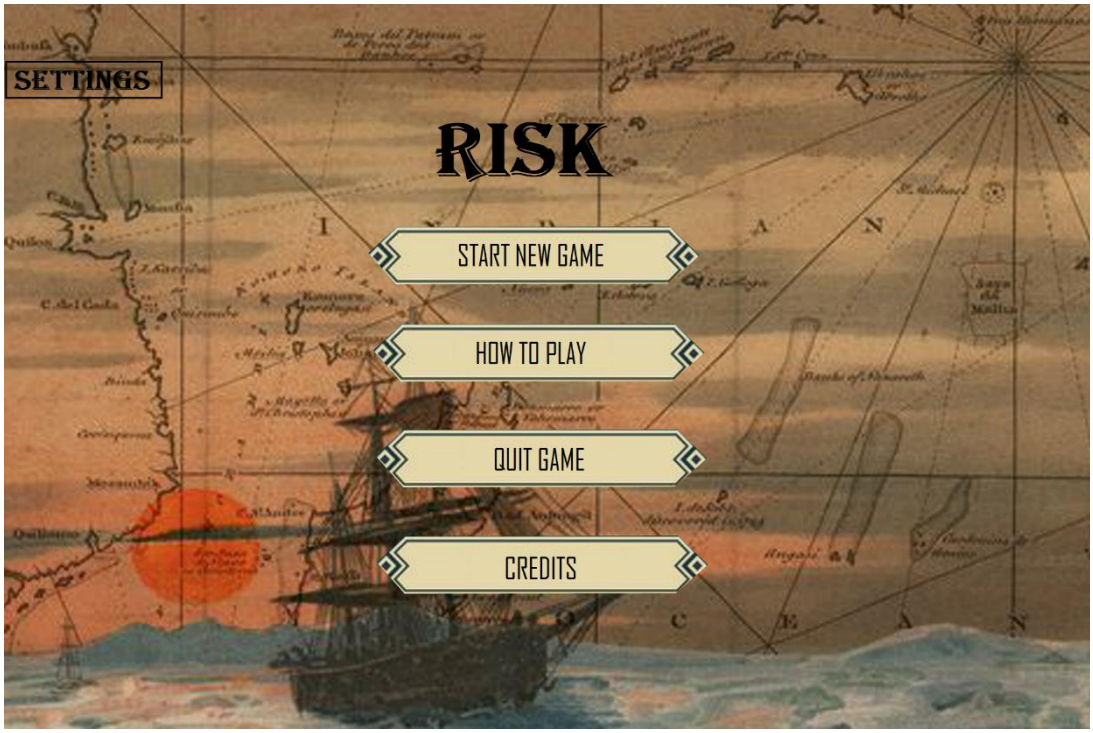


Figure 9: Main menu interface.

4.6.2 How to Play

How to play page, presented in Figure 10, gives the player information about the rules of the game. Player can go back to the main menu or the pause menu with the “BACK” button.

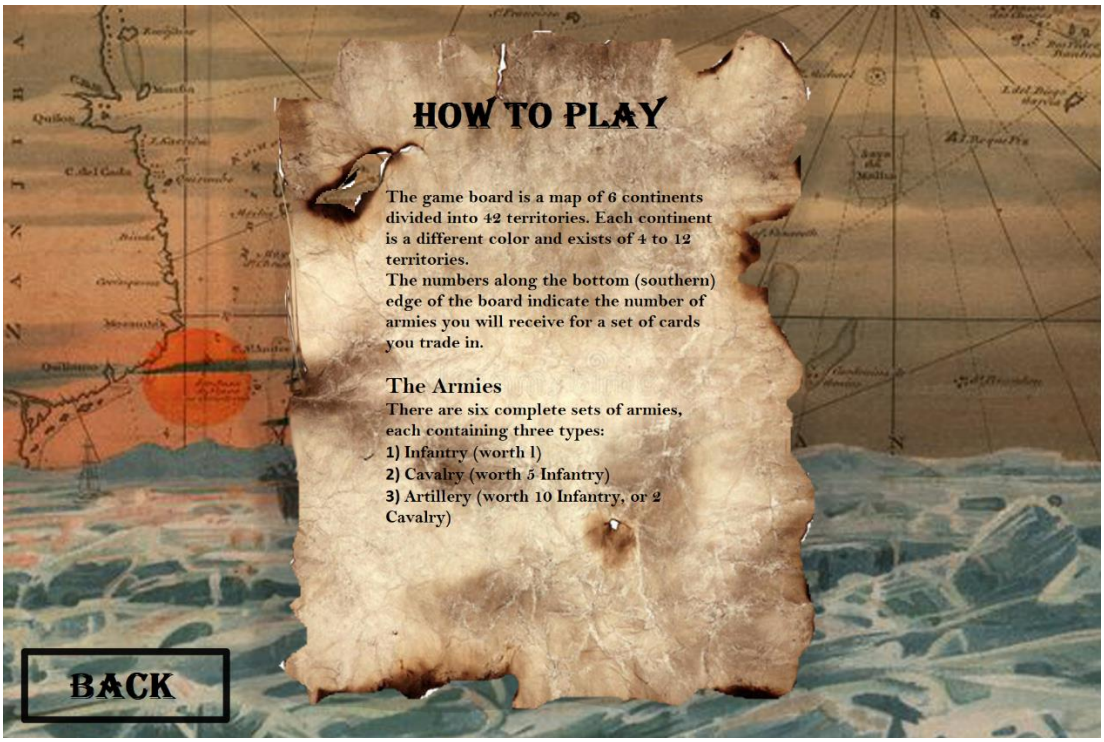


Figure 10: How to Play interface.

4.6.3 Settings

In settings page given in the figure below (Figure 11) the user will be able to change the volume or completely turn of the sounds and music of the game. Sound are the sound effects for the gameplay while music is the general background music that plays throughout the game. Player can save the changes with “SAVE” button or go back to main menu or pause menu with “BACK” button without saving the changes.

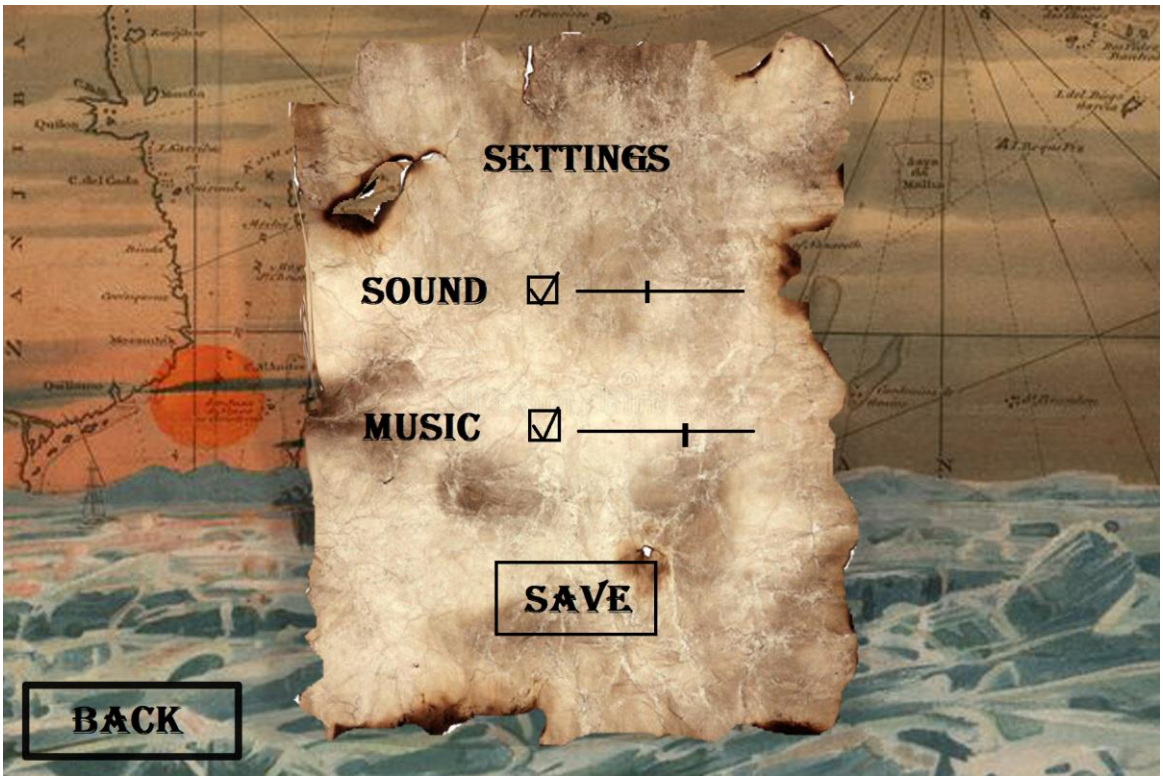


Figure 11: Settings interface.

4.6.4 Credits

Credits page, as shown in Figure 12, contains the names of the developers of the game. Player can go back to main menu with “BACK” button.

Figure 12: Credits interface.

4.6.5 Setting the Game

The player will be directed to this page given in the figure below (Figure 13) when they start a new game. They will be able to choose the number of players. Every player will be able to write their name, choose their territory color and their avatar as well. With “CONTINUE” button game will start with the chosen options. With “BACK” button changes will be lost and player will go back to the main menu.

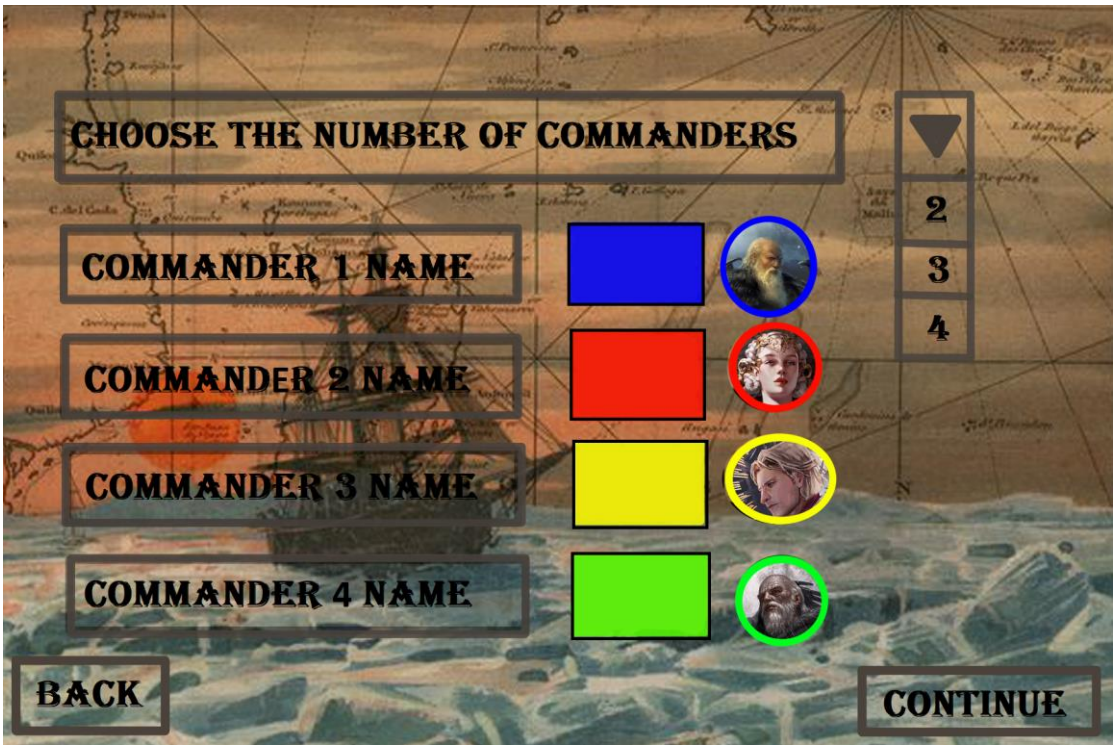


Figure 13: Setting the Game interface.

4.6.6 Gameplay

This page shown in Figure 14 is the main game view. Player's avatars will be displayed in the corners and when its player's turn, their avatar will be lit up. In the bottom, the cards of the player will be displayed and will be used when it's their turn if they want. Player will be able to attack the territories of their enemies if it is in the ruleset. Player will also be able to retreat from their territory if they find it necessary. Player is also able to pause the game with the button in the top left corner.

Figure 14: Gameplay interface.

4.6.7 Attack Pop-up

When a player decides to attack this page, given in Figure 15, will pop up. Players can roll the dice with “ROLL DICE” button and decrease the number of dice that they would like to attack with, with down arrow button. Player under attack can also call for help from other players with “ALLIANCE” button. With “BACK” button attack will be declined.

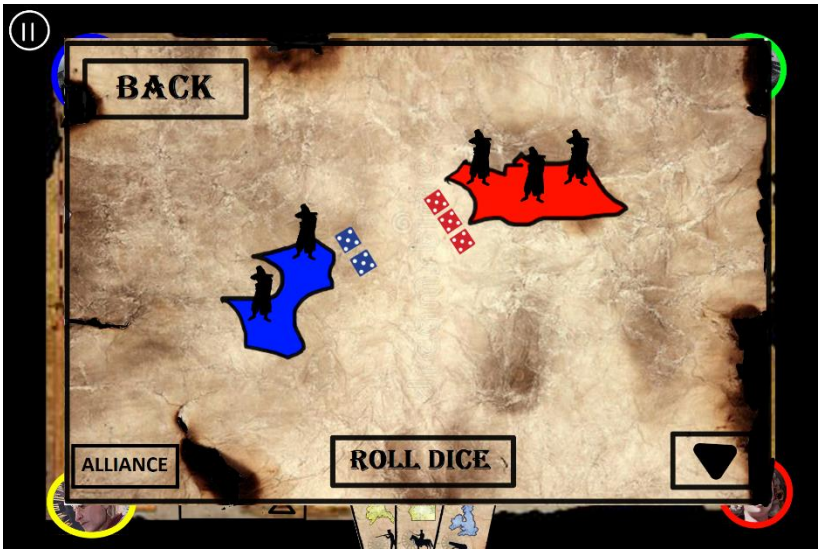


Figure 15: Attack pop-up interface.

4.6.8 Pause Game

In this menu given below (Figure 16), player is able to remind themselves of the rules of the game with “HOW TO PLAY” button that will take the player to how to play page or change the settings of the game with “SETTING” button that will take the player to the settings page. Player will be able to continue the game with “CONTINUE” button and quit the game with “QUIT” button and go back to the main menu.

Figure 16: Pause Menu interface.

4.6.9 Information Pop-up

This pop-up, shown with Figure 17, will be shown to the player when it’s their turn when new soldiers are added to their army or when they exchange their cards with soldiers. The player will be able to close this pop-up with the “X” symbol located at the top right of the pop-up.

Figure 17: Information Pop-up interface.

5. Improvement Summary

In the second iteration, we mainly made changes to the system models based on both feedback and peer reviews. We also updated the general game ruled and descriptions based on the changes in rules for our implementation of Risk. The changes are explained in the list below.

- The structure of the report was improved to allow for higher readability of the diagrams.
- The requirements section was improved in terms of descriptions and a new non-functional requirement, usability, was added.
- The use case model of the project was updated and redone to include the gameplay related cases.
- The class diagram was updated as well as the class descriptions according to the design goals and rules. Unnecessary classes were removed and more important ones added.
- The sequence diagrams were updated to convey better scenarios and also for integrity with the class diagram. Each description was rewritten accordingly. The number of sequence diagrams was also lowered based on feedback

- The activity diagram was updated based on the game rules and new paths added to our implementation. The necessary changes were done to the description of the diagram.
- A state diagram showing playing a turn was added which did not exist in the first iteration. A description for the chart was also added.

6. Conclusion

The analysis of our implementation of the classic board game Risk has been described with the content above. The overview of the implementation clarified the components, mechanics and the rules of the game as well as new features added by us. The functional requirements explained all the actions that the users could perform and the non-functional requirements presented the functionalities that needed to be considered in order to reach a wide and happy user base. The system models were planed out and carefully produced by all of us, with giving as much attention to detail as possible. The diagrams and descriptions will help us in the implementation of this project.

7. References and Glossary

[1] “Risk: An Unexpected Journey”. Zorcon’s World. http://zorconsword.blogspot.com/2012/12/risk-unexpected-journey-part-4.html (accessed October 24, 2020).

[2] Design You Trust. https://designyoutrust.com/ (accessed October 24, 2020).

[3] Reddit. https://www.reddit.com/r/ImaginaryWesteros/comments/elgb0m/lord_commander_mormont_by_ryan_valle/ (accessed October 24, 2020).

[4] Reddit. https://www.reddit.com/r/gameofthrones/comments/bj2tey/no_spoilers_jaime_lannister_painting_by_jewell/?utm_source=ifttt (accessed October 24, 2020).

[5] “Risk Oyun Kılavuzu”. Hasbro. https://www.hasbro.com/common/documents/dad2886d1c4311ddbd0b0800200c9a66/D3A98A4650569047F5C718754E1236CF.pdf (accessed October 24, 2020).