CS 319 - Object-Oriented Software Engineering

Fall 2020

Project Analysis Report

Monopoly

Group 3D

Cansu Moran - 21803665

Elif Gamze Güliter - 21802870

Melisa Taşpınar - 21803668

Öykü Irmak Hatipoğlu - 21802791

Yiğit Gürses - 21702746

Table of Contents

Table of Contents 2

1 Introduction 5

2 Overview 5

2.1 Gameplay 5

2.2 Board 6

2.3 Properties 7

2.4 Chance and Community Chest Cards 7

2.5 Players 8

2.6 Editing feature 8

3 Functional Requirements 8

3.1 General Features 8

3.2 Playing the Game and Rules 9

3.2.1 Starting 9

3.2.2 Game actions 9

3.2.2.1 Property 10

3.2.2.2 Landing on Special Squares 11

3.2.3 End of the game 12

3.3 Editing the game features 13

3.3.1 Editing properties 13

3.3.2 Editing Joker squares 13

3.3.3 Editing pictures 14

3.3.4 Editing Miscellaneous Features 14

4 Non-Functional Requirements 14

4.1 Usability 15

4.2 User Friendliness 15

4.3 Reliability and Maintainability 16

5 System Models 16

5.1 Use Case Model 16

5.1.1 Use Case Name: Play a Game 17

5.1.2 Use Case Name: Edit a Board 17

5.1.3 Use Case Name: View Help 19

5.1.4 Use Case Name: Change Settings 19

5.1.5 Use Case Name: Display Credits 20

5.2 Dynamic Models 22

5.2.1 Sequence Diagrams 22

5.2.1.1 Editing the Game : Properties 22

5.2.1.2 Editing the Game : Joker Places 23

5.2.1.3 Landing on a Property and Buying it 24
5.2.1.4 Landing on a Joker Place and Paying Rent 25
5.2.1.5 Landing on Community Chest and Chance Squares 27
5.2.1.6 Changing Settings and Showing Credits Page 29
5.2.2 Activity Diagram 30
5.2.2.1 Play a game 31
5.2.2.2 Edit a board 32
5.2.3 State Diagram 33
5.2.3.1 Main state diagram 33
5.2.3.1.1 Game State 34
5.2.3.1.2 Editing State 35
5.2.3.1.3 Settings State 36
5.3 Object and Class Model 37
5.4 User Interface 41
5.4.1 Navigation Path 41
5.4.2 Screen Mockups 42
5.4.2.1 Main Menu 42
5.4.2.2 Player Selection Screen 43
5.4.2.3 Board Selection Menu 44
5.4.2.4 Settings Screen 45
5.4.2.5 Credits Screen 46
5.4.2.6 Help Screen 47
5.4.2.7 Edit Screen 48
5.4.2.8 Play Screen 51
5.4.2.9 In-game Menu 52
5.5 Priorities of Requirements 53
5.6 References 54

1 Introduction

Our group 3D, decided to implement the board game “Monopoly” for our CS 319 project. The game’s objective can be predicted from its new where players aim to become the one and only surviving player in the game by making other players bankrupt by buying all the properties, like the economic term monopoly, where there is a single presence dominating the market. Due to its numerous aspects such as properties, tokens (the pawns in Monopoly), chance and community chest cards, and money, it is a suitable game for implementing Object Oriented programming. Since it is a game we played during our childhoods and we know all the rules by heart, we chose to design Monopoly. However, we decided to add a function that will elevate the game in many ways such as creativity and flexibility.

What makes our Monopoly different from the original one is the fact that we are introducing an edit feature for Monopoly. There are various themes of Monopoly on the market such as Monopoly Turkey version, or TV characters version and if you want to play them, you should purchase all the themes you want separately. However, with our edit feature addition to Monopoly, the game can be edited in any way that the players wish. You can design your own version of Monopoly like Monopoly space or Monopoly Bilkent. You can even design the game in such a way that you create your own board game apart from Monopoly by making the board consist of Joker places. When you build your own version of Monopoly, you can save it and play it with your friends.

2 Overview

2.1 Gameplay

The players will start at the “beginning” square on the Monopoly board. In the beginning of the game all players will have the same amount of money that was predefined on the editing section or the amount of money selected by default. Each player will have their own tokens and move them by spinning the dice wheel. According to the number on the spinning wheel, players will move their tokens forward to the corresponding square. The square they landed on can be a property square or a joker square. If the square is a property square and it has not been purchased by another player, the player can buy this place if he/she has enough money.

However, if this property has been purchased by someone, the player who landed on that square should pay the corresponding money to the owner of that property. On the other hand, if the square is a joker square, players will perform the actions defined on that joker square like drawing a card or moving a few squares forward or backward. The contents of joker squares can be determined in the edit section. During their turns, players can purchase houses if they own all of the properties of a certain color. Players can even put hotels on those properties if there are already 4 houses on the property. By putting houses and hotels on the properties, the owners make the rent of these properties higher and higher with each house and hotel. The game goes on until everyone except one player gets bankrupted. The last one standing wins the game.

2.2 Board

The Monopoly board consists of 40 squares which can be property squares or special squares such as community chest or chance squares. With the help of the editing feature of our version, the user can design the board whichever way he/she likes. The pawns of the players will move across this board according to the result on the spinning wheel that corresponds to the dice on the original board game. The board has 2 card decks, chance card deck and community chest card deck from which players will draw from if they land on the related squares during the game. In the beginning of the game, pawns of all players will start at the same square and they will move clockwise unless a special case requires them to move backwards.
In the edit section, the board designed by the user can be saved to be used later or players can immediately play on the board that was just designed. Additionally, players can choose the board they want to play on from the boards that have been saved earlier or the default boards provided on the game. When editing the board, the user can be guided by the design of the default board.

2.3 Properties

Properties are the squares on the Monopoly board which can be buyed and therefore owned by players. When a player lands on a property owned by another player, he/she has to pay rent. Some properties have the same color groups. If a player manages to buy all properties on a certain color group, the player has the opportunity to put houses and hotels on those properties which increase the rent value of those properties. If the player has to find money, he/she may

choose to mortgage his/her properties after selling any houses or hotels on the particular property. By mortgaging the property, the owner gets a predetermined amount of money from the bank, still has the possession of the property, however, cannot get rent from that property. The player also has the opportunity to sell the property directly to the bank with a reduced amount of money compared to the buying price in return. Again, the player has to sell any hotels or houses before he/she decides to sell the property to the bank like mortgaging.
When editing the board, the user can also edit the properties. The user can change the names, color groups, and the locations of the properties. The user can also change the money rates of the properties, for example, the mortgage rate, which is for what percent of buying price should the property be mortgaged to, and the selling price, which is the price the bank pays the player when he/she sells his/her house. Other rates that can be changed are the hotel and house price and rent amount.

2.4 Chance and Community Chest Cards

There are two kinds of cards in the game, chance and community chest, which will be drawn when the player lands on certain squares. Chance cards often have movement actions such as “move 5 squares forward” or “move 8 squres backwards” which will take the players to another square where they can buy a property, pay rent to another player, or even draw another card. Chance cards can have custom actions such as go to jail or take a property from another player. On the other hand, community chest cards are often related to money. A community chest card can give the player some money, take taxes from the player, or even make the other players give money to the player.

2.5 Players

The game can be played by up to 4 players. The players can be selected and edited after the user presses the play button on the main screen. Each player chooses a color and a pawn to play with which will identify the player himself on the game by moving through the squares. All players start the game with a certain amount of money. When a player gets suspended due to drawing a card or landing on a square, the player cannot move for a specific number of turns, however he/she can still get rents from his/her properties.

2.6 Editing feature

The most distinguishing feature of our version of Monopoly is the editing feature which enables players to play the Monopoly theme they want to play by allowing them to design their games. The edited boards can be saved to be played with later. From the currency on the game to the picture on the chance card square our version offers many features to edit and customize the game. By pressing the edit button on the main menu screen, the user can reach the editing screen where he/she can change properties and joker squares. Users can add as many property or joker squares as possible which enables different approaches to Monopoly, one where there are few properties and a lot of joker squares contrary to the original Monopoly. The users can also change the appearance of the board, cards, and squares if they are planning to use a theme on the design such as computer Monopoly where, for example, chance cards have 0 and 1's and community chest is depicted as cloud services.

3 Functional Requirements

3.1 General Features

In our project, we will be implementing a computer version of the board game Monopoly. The game will not be played online, instead, it will be implemented as a one-computer game where players will play in turns. Different from the original Monopoly, our version can only be played by 2-4 players. We decided not to use the original limit, which is 2-8, because as the number of players increase, players will start to lose their attention since they will start to wait more for their turn to come. Therefore, we decided to limit the number of players to at most 4.

At the beginning of the game, a version of Monopoly that is similar to the board game will be offered to players. The players can either play this default version or they can go to the editor to create their own versions using the default one as a template. After saving their versions, users can still edit them and create new versions using their saved games as templates, or they might update their already existing versions. Users can play on any board that they have saved. The default version of the game has similar rules and squares to the original Monopoly. However, with the editor, the users can edit the neighborhoods and they can add different squares with properties that are not necessarily included in the original Monopoly.

3.2 Playing the Game and Rules

3.2.1 Starting

At the beginning of the game, the user is presented with Edit and Play options. Play option takes the user to a screen where they can select which board they would like to play on. After board selection, the number of players should be selected. Following that, players will enter their usernames in order. The first player who wrote her/his username will start the game and other player's turns will come in the same order with their names. Every user will be assigned a color of their choosing so that after they buy a property (see 3.2.2) the color of that property's square on the board will change to the player's color.
Every user will receive money at the beginning of the game i.e. 1500 $ which can be edited in the Editor if players ever want to start with different amounts of money (see section 3.3). All the players' tokens start from the Start square. The game begins when the first player rolls the dice, in other words, spins the wheel with numbers between 2 to 12. When a player's turn is over, they click the "Next Turn" button to pass the turn to the next player.

3.2.2 Game actions

The game will be auto controlled, which means that users will not have to move their tokens, they will only spin the wheel, which acts as a replacement to dice in the original game. According to the number that landed on the wheel, users' tokens will be moved to the designated square automatically.
The squares on the board can either be properties or special squares such as chance, community chest and Joker. According to the features of the square that the tokens arrive, players will be able to perform certain activities. After performing these actions, which will be explained below, players will declare that their turn is over by pressing the "Next Turn" button and the next user will spin the wheel for his/her turn.

3.2.2.1 Property

Buying a property : When a player arrives at a property, he/she can buy that neighbourhood if it is not already bought by other players. After buying that property, the color of the square will change into that player's color. In the board game version, players need to keep cards of the places that they own. However, in this version, they will not hold cards, instead, the places that they own will be indicated by their colors on the board.

Paying rent : If a player lands on a place owned by other players he/she must pay rent. The rent is taken from the user and is transacted to the owner automatically by the game. Rents increase with a predetermined percentage if the owner has houses and hotels built on that property. If the property is mortgaged, the owner can't collect any rent from other players who arrive at that property.

Selling a property: The properties can only be sold to the bank. To be able to sell a property, no buildings must be present on any of the properties of that color group. If there are any houses/hotels present on that color group, they must be sold first to be able to sell any property in that color group. In addition, that specific property should not be mortgaged. If all the conditions are met, the owner can sell their properties to the bank for a certain percentage of the initial paying price, which is 75% by default. This percentage can be changed in the editor.

Mortgage: Any property owned can be mortgaged. When a property is mortgaged, the owner receives half of the paying price of that property from the bank. The owner still holds the property, meaning other players can't buy a mortgaged property. However, the owner can't collect any rents from that property anymore. Rent can still be collected from properties of that same color group that are not mortgaged.

To be able to mortgage a property, there shouldn't be any houses or hotels at any of the properties of that color group. If there are houses and hotels in that color group, the owner must sell those houses/hotels to the bank first. Then the property can be mortgaged.

To lift a mortgage, the owner must pay the bank a certain amount of money. This money is calculated as the amount of mortgage money that the bank gives the owner (half of initial paying price) plus 10% of that money. To be able to build houses on any property of a certain color group, all the properties must be unmortgaged. Mortgaged property cannot be sold. The mortgage must be lifted first to be able to sell the property back to the bank.

Building houses and hotels: . To be able to build houses in any of the properties, the owner must own all properties of that color group. If they own all properties in the same color group, they can start building houses. The upper limit for houses is 4. The owner can't start building a second house on a property unless all of the properties in that color group already have a house on them. Similarly, they can't build a third house unless all

properties of that color group already have two houses and they can’t build a fourth house unless all properties of that color group already have three houses.
After building 4 houses on each property of that color group, the owner can start building hotels. A property can only have one hotel. When a hotel is built, it replaces all the houses that were built before and the owner starts getting hotel rent from the visitors.
Building houses and hotels is advantageous for players since the rent increases with a certain percentage when owners build houses and hotels. However, the cost of houses and hotels depend on the initial price of that property. In other words, in order to be able to build houses on more expensive properties, the owner must pay more.
Houses and hotels can be sold back to the Bank for half of the price paid for them. Similar to building houses, the selling of the buildings must be done in order. For example, the owner can’t sell two houses on one property. If they sell one house on that property, then the next house they sell should be on another property so that the number of houses decrease equally on properties of the same color. If there is a hotel on a property, it must be sold first, before any houses can be sold. When a hotel is sold, it will be replaced by 4 houses. These houses can be sold afterwards.

3.2.2.2 Landing on Special Squares

- **Joker Places:** There are certain squares in the original Monopoly that we decided not to include in our version. Instead, we decided to create a special Joker square that can be edited into a similar square to the ones included in original Monopoly or can be edited to be a completely different square.
  In traditional monopoly, there is “Jail” where players must go wait for a couple of turns if they land on the “Go to Jail” block. Furthermore there are two other special squares called Income Tax where players pay an amount of tax if they land on it and Free Parking where no action such as paying tax or paying rent is required. The Joker square in some sense is a combination of these squares in the original Monopoly. Instead of a classic jail, income tax or free parking places, there will be Joker squares that can have 3 different properties. These properties include winning/losing money, having to wait for a certain number of turns and moving a certain number of squares ahead. In the Editor, the user can create different Joker squares using a combination of these features.

These Joker places can be made into a Jail, if the user adds a certain number of turns to be waited for players who land on this square. They can also be made into an Income Tax square by adding money actions to that square. They can also be made into a Free Parking square by not adding any money or movement actions. The features of these Joker squares will be up to players, therefore, if they prefer, they can have no jail in the game or they can have multiple jails or even have a jail where if you go to jail you earn/lose money. Unlike the neighborhoods, Joker places cannot be bought.

- **Community Chest and Chance :** These places are another type of square where players can land on. When players land on either one of these, they will draw a card from the corresponding deck of cards. The task on the card will be done automatically by the game. Some of the cards will have randomly generated numbers (i.e. the amount of money that will be earned, number of squares that the player will move forward/backward).

3.2.3 End of the game

A player loses the game if he/she gets bankrupted. When a player starts the turn with a negative balance of money, the player should make the balance positive by selling hotels or houses, mortgaging or selling the properties, or if the player is lucky, getting money from the chance or community chest cards. If the player is not able to make the balance positive or at least 0, he/she cannot move to the next round and the player has to declare bankruptcy. The game ends when all the players except one goes bankrupt. The player who doesn’t go bankrupt wins the game.

3.3 Editing the game features

If the user chooses the Edit option, they can edit any one of the saved boards, including the default board. On the board, all the squares except the default Start square can be edited. There is no limit to how many squares of one type the user can have. In other words, if the user wants, they can make all the squares Properties or they can make all of the Chance squares. It all depends on their preferences. Each square must be one of 4 types of squares: Property, Chance, Community Chest and Joker. The features of Property and Joker squares can be edited. Chance and Community Chest squares, on the other hand, can only be added to the board. No further editing can be done on these two types of squares, since the actions on these squares are determined by card decks that cannot be edited, only the currency is replaced(see 3.3.4).

3.3.1 Editing properties

One of the square types that the user can add is Properties, which are explained on 3.2.2.1. The users can add as many properties as they want and they can edit their features. Each property should be given a name, color group and initial buying price. All of these features can be edited on the editor. The user can either add the property to an already existing color group or they can create a new color group. If they choose to add a new color group, they must select a color from the color chart and determine a name for that specific color group.

3.3.2 Editing Joker squares

As mentioned in the section 3.2.2.2, Joker places can be edited according to players' preferences. Unlike the board game version, players might prefer both having rewards and punishment in their game or they might prefer only one of them. For example, they can prefer all Joker places to have punishments like Jail in board game versions.

There are three main features that Joker places offer to players : waiting for a few turns, being moved to few squares forward or backward and get/lose money. Users can't make a Joker square have the properties of both waiting for a few turns and moving forward/backward. Either waiting or moving can be chosen for a square. The third feature, getting/losing money can be used in combination with the other two features or on its own. For example, they can choose the Joker place to move the player's token to 2 squares ahead and give the player $50 or take $50 from the player. The user can also create a Joker square with no purpose, similar to "Free Parking" in the original Monopoly. Extensibility of editing Joker places will be limited by the game. For example, if players create a Joker square to make the player move ahead, there will be a limit to the number of squares they can move such as being able to move at most 5 squares.

3.3.3 Editing pictures

On certain squares of the board, pictures can be uploaded. These squares include Joker squares and Chance and Community Chest squares. Pictures can also be attached to the tokens of the players to create a more custom game. Pictures can be selected from the computer and uploaded to the game for each one of the specified squares or tokens.

3.3.4 Editing Miscellaneous Features

The user can make further customizations on the game like changing the currency or the percentages of house costs or selling rates. For example, if the user is planning to make a space-themed Monopoly, the currency of the game can be changed to meteors. When the player is buying a property, he/she will pay 100 meteors instead of 100 Monopoly Dollars. Consequently all the properties, chance and community chest cards, etc. will be updated regarding the new currency. Another feature that can be edited is the percentages on the game. For instance, the user can decide on the rate between the purchase price and the rent of the same property or the mortgage rate.

4 Non-Functional Requirements

The original limit given to the number of players that can play a single game is 2-8. However, we made the decision to change this limit into 2-4 on our game. Our reasoning is that the UI can get cluttered and confusing with 8 different players on the screen. Also, it can get boring to wait for 7 people to go through their turns especially when playing the game on one computer.

4.1 Usability

- Anyone that knows how to play regular monopoly and has basic computer skills should be able to play the default game mode. Getting used to the GUI should not take more than ten minutes. To achieve this, we will try to make the UI resemble the real-life version of the game whenever we can. In cases we cannot, such as the buttons to choose actions, will be self explanatory with the texts and or icons on them.

4.2 User Friendliness

- It is possible to create many kinds of boards in the Board Maker section. However, these boards might not always be enjoyable or even playable. We will limit the editability of the board to some extent in order to make it harder for the users to accidentally create unplayable boards. For instance, we created different categories of squares which have different editable properties. This will make sure the user cannot select conflicting properties for one square.
- Moreover, we have limited the number of squares a player can be made to move with a Joker square. That is, a Joker square with the moving property (making a player that lands on it moves a certain number of squares) can move that player at most the length of the board, in order to make the game more playable and to avoid loops.
- We will ask for confirmation when the user tries to exit Board Maker when there are unsaved changes. Also, we will ask for confirmation if the user creates a new board and tries to save it on an existing board. As a final measure, the recently deleted boards will be kept in a separate folder so they can be recovered if needed.
- We added titles on each screen to tell the user where they are. This way they will not get lost between the menus.

4.3 Reliability and Maintainability

- The game should work on all up-to-date Windows computers with JVM installed.
- The game should not crash.

5 System Models

5.1 Use Case Model

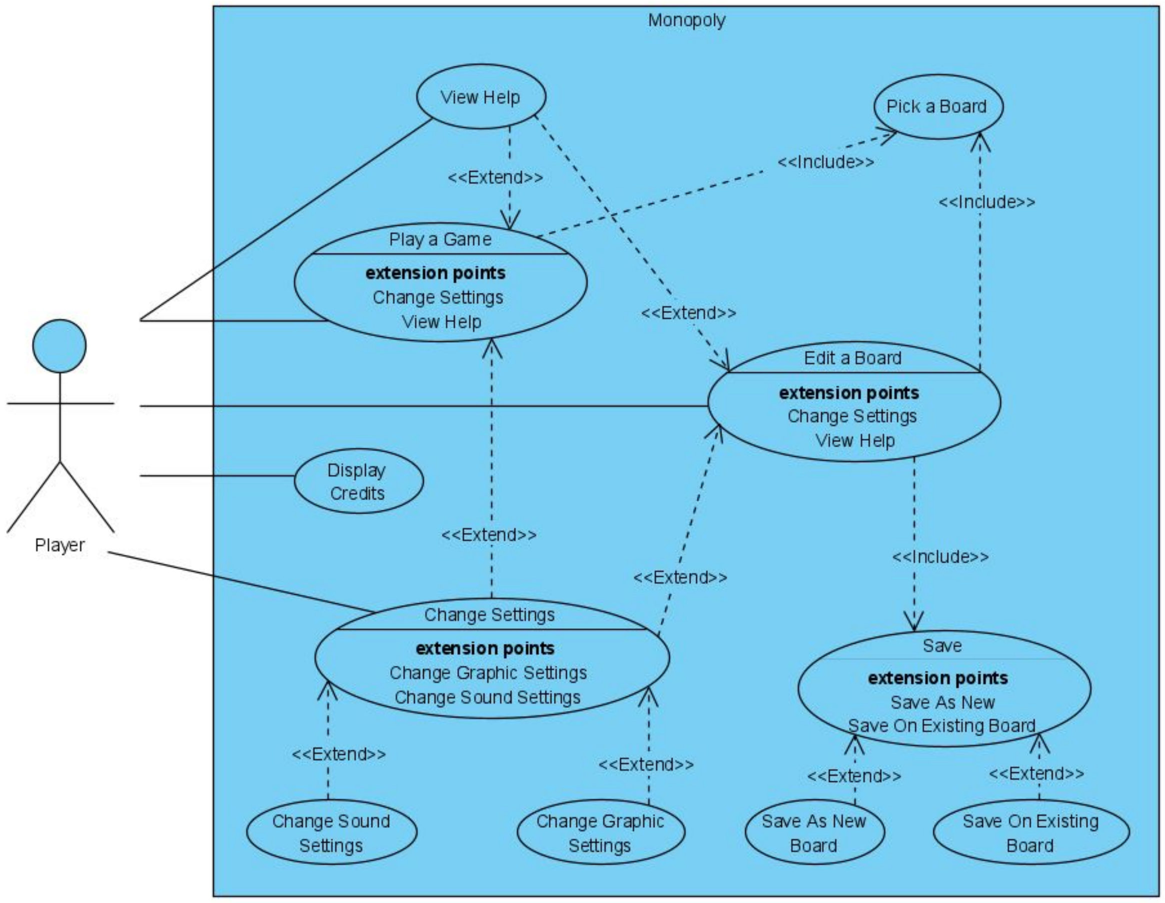


Figure 5.1 Use Case Diagram

5.1.1 Use Case Name: Play a Game

Participating Actor: Player/Players

Stakeholders and Interests: Player wants to start a new game session.

Pre-Conditions: Player is in the main menu.

Post-Conditions: Player is in the main menu.

Entry-Condition: Player clicks the “Play” button on the main menu. User selects a board from the list to play on.

Exit-Condition: Player clicks the “Exit” button. This can happen during the game session or when the game session ends.

Basic Flow:

1. Player clicks the “play” button on the main menu
2. System displays the available boards to play on as a list
3. Player selects a board from the list
4. System initializes a new game session with the settings of selected board
5. Player spins the spinning wheel, takes actions and ends their turn
6. System processes the spinning wheel and actions, gives the control to next player in line
7. Players keep taking turns until they go bankrupt
8. Only one player is left who is not bankrupt
9. The game session ends, winner is displayed on the screen
10. Player clicks the “Exit” button and returns to main menu

Alternative Flows:

1. Players want to return to main menu before the game session ends
    1.1. Player opens in game menu by pressing “Esc”
    1.2. Player clicks the “Exit” button
    1.3. System asks for confirmation
    1.4. Player clicks “Yes”
    1.5. System exits to main menu

5.1.2 Use Case Name: Edit a Board

Participating Actor: Player

Stakeholders and Interests: The player wants to edit a board or create a new one.

Pre-Conditions: The player is in the main menu.

Post-Conditions: The player is in the main menu.

Entry-Condition: Player clicks the “Board Maker” button on the main menu. User selects a board from the list to edit or clicks the “Create New Board” button.

Exit-Condition: Player clicks the “Exit” button.

Basic Flow:

1. Player clicks the “Board Maker” button on the main menu
2. System displays the available boards to edit as a list
3. Player selects a board from the list to edit
4. System loads and displays the board editor screen with the settings from selected board
5. Player selects a square on the board
6. System displays options to change settings for that square
7. Player makes changes on the square
8. Player is done with the changes and clicks the “Save” button
9. System asks for confirmation
10. Playes clicks “Yes”
11. System saves the changes
12. Player clicks the “Exit” button and returns to the main menu

Alternative Flows:

1. Player wants to create new board instead of editing an existing one
   - 1.1. Player clicks the “Create New Board” button instead of selecting a board from the list
2. Player wants to save edited the board as a new board
   - 2.1. Player clicks the “Save As New” button instead of the “Save” button
   - 2.2. System asks for a name
   - 2.3. Player enters name
   - 2.4. System saves the current board settings as a new board with the given name
3. Player wants to exit without saving
   - 3.1. Player has unsaved changes
   - 3.2. Player clicks the “Exit” button
   - 3.3. System asks for confirmation with the warning message “You have unsaved changes. Do you wish to exit without saving?”
   - 3.4. Player clicks “Yes”
   - 3.5. System exits to main menu

5.1.3 Use Case Name: View Help

Participating Actor: Player

Stakeholders and Interests: The player wants to learn how to play or how to edit boards.

Pre-Conditions: The player is in the main menu or the in-game menu.

Post-Conditions: The player is back at the screen before they view help.

Entry-Condition: Player clicks the "Help" button on the main menu or the in-game menu.

Exit-Condition: Player clicks the "Back" button.

Basic Flow:

1. Player clicks the "View Help" button on the main menu
2. System loads the "Help" screen where information about the game is displayed in text
3. Player reads the text and acquires required information
4. Player clicks the "Back" button
5. System exits to main menu

Alternative Flows:

1. Player clicks the "View Help" button in the in-game menu instead of the main menu
    - 1.1. When Player clicks the "Back" button, system returns to the screen the "View Help" button was clicked instead of going directly back to main menu

5.1.4 Use Case Name: Change Settings

Participating Actor: Player

Stakeholders and Interests: The player wants to learn how to play or how to edit boards.

Pre-Conditions: The player is in the main menu or the in-game menu.

Post-Conditions: The player is back at the screen before they change settings.

Entry-Condition: Player clicks the “Settings” button on the main menu or the in-game menu.

Exit-Condition: Player clicks the “Back” button.

Basic Flow:

1. Player clicks the “Settings” button on the main menu
2. System loads the settings menu
3. Player clicks “Fullscreen Mode" button to enter fullscreen when the game is not or “Exit Fullscreen Mode” button to exit fullscreen when the game is on fullscreen mode.
4. Player clicks the “Back” button
5. System asks for confirmation if the player wants to keep the new settings
6. Player clicks “Yes”
7. System saves the changed settings permanently
8. System exits to main menu

Alternative Flows:

1. Player clicks “Change Settings” in the in-game menu instead of the main menu
    - 1.1. When Player clicks the “Back” button, system returns to the screen the “Change Settings” button was clicked instead of going directly back to main menu
2. Player wants to return to old settings instead of saving the new ones
    - 2.1. When the system asks for confirmation, player clicks “No” instead of “Yes”
    - 2.2. System returns to the previous settings that existed before the “Change Settings” button was clicked

5.1.5 **Use Case Name:** Display Credits

Participating Actor: Player

Stakeholders and Interests: The player wants to learn about the developers.

Pre-Conditions: The player is in the main menu.

Post-Conditions: The player is in the main menu.

Entry-Condition: Player clicks the “Credits” button on the main menu.

Exit-Condition: Player clicks the “Back” button.

Basic Flow:

1. The player clicks the “Credits” button on the main menu
2. System displays the information about developers on the screen
3. The player clicks the “Back” button
4. System exits to main menu

Alternative Flows:

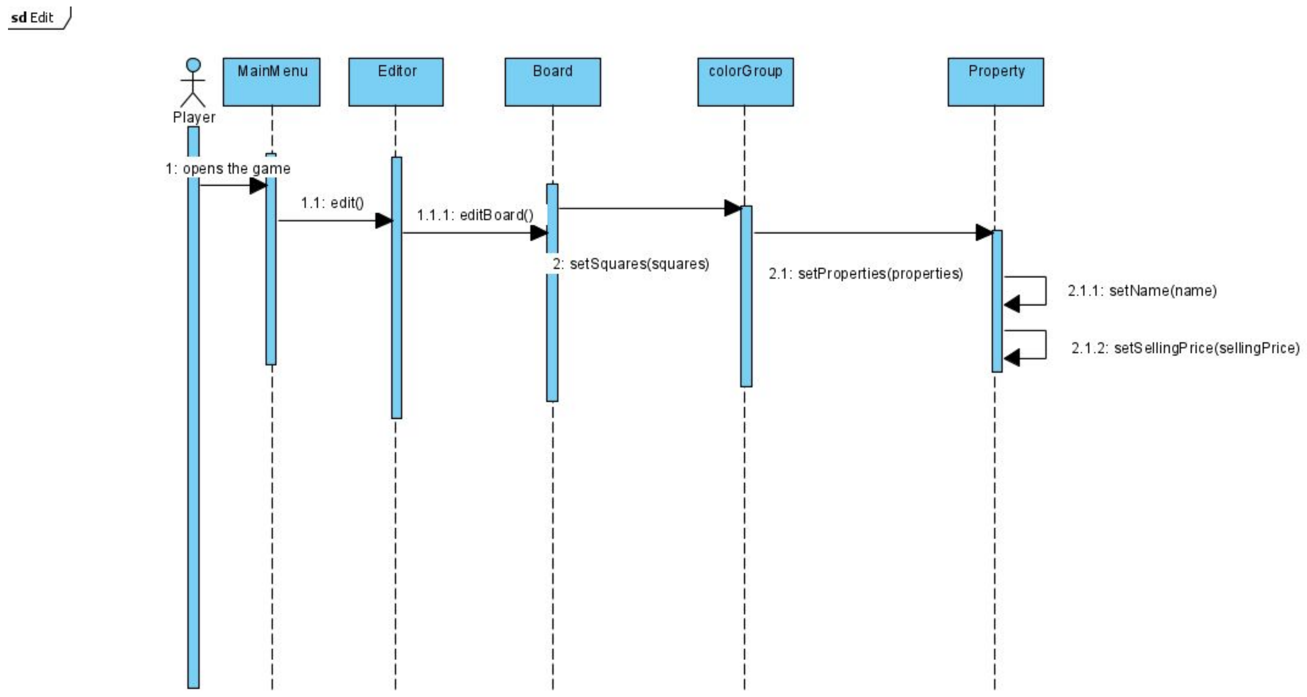
None.

5.2 Dynamic Models

5.2.1 Sequence Diagrams

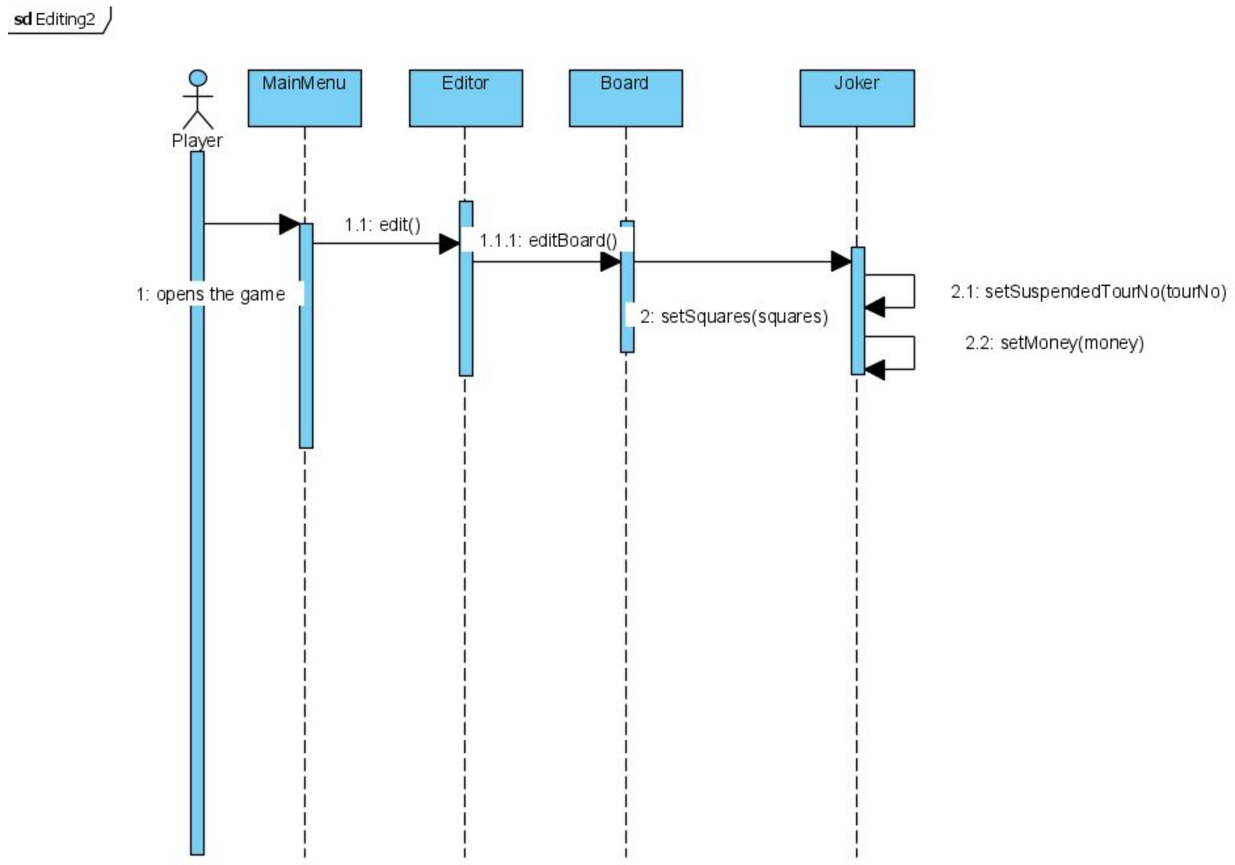
5.2.1.1 Editing the Game : Properties

Scenario : In this scenario, the player prefers to edit the board given in the game. In order to obtain this, the player presses the edit button in the main menu and a message is sent to Editor class. Following that, the editor class sends a message to Board class and Board class sends a message to Property in order to enable editing. In the diagram given below, the player prefers to edit the selling price and a name of a certain property in a ColorGroup. However, the player can also edit other attributes of a property such as its rental price.



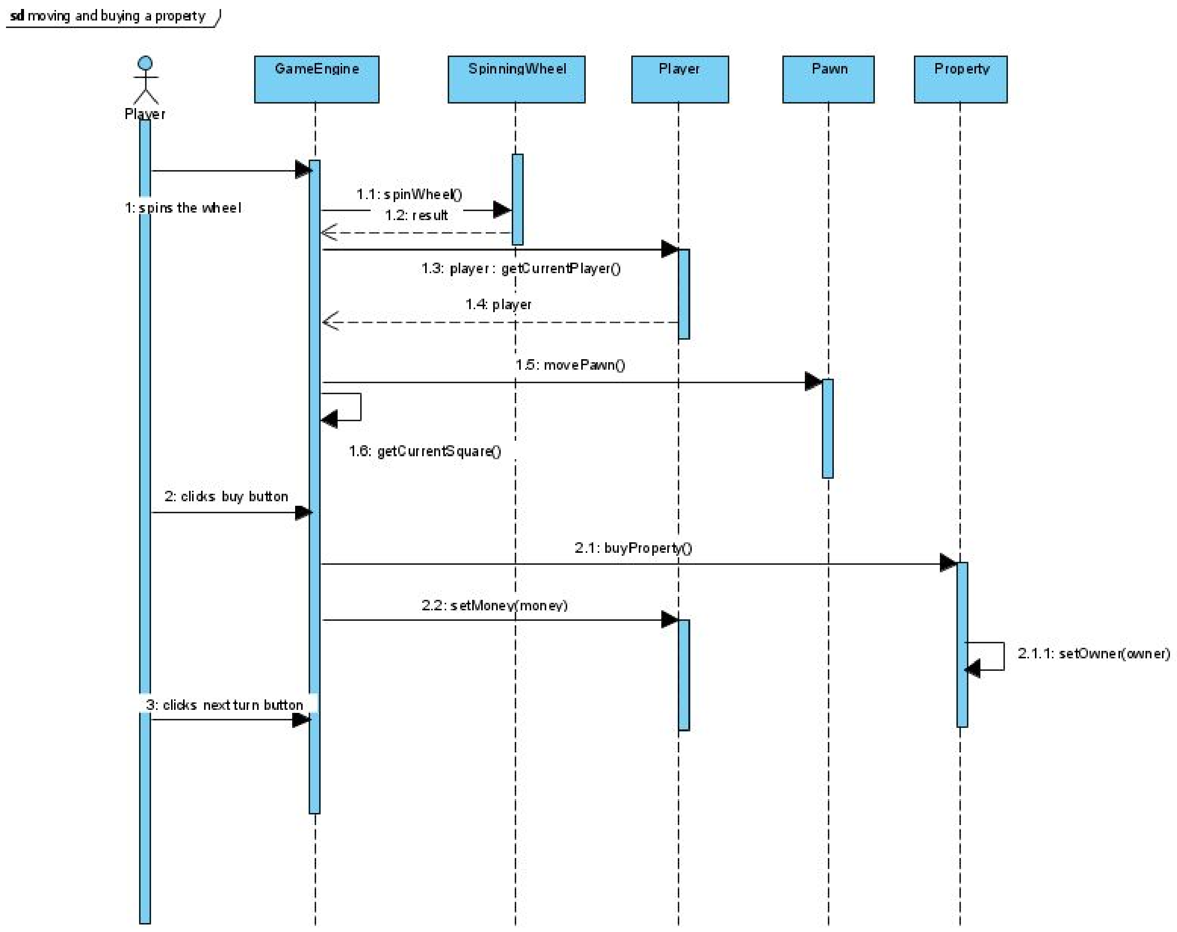
5.2.1.2 Editing the Game : Joker Places

Scenario : In this scenario, the player changes some of the actions required in a Joker place. First, the player presses the edit button in the main menu and sends a message to the Editor class. Following that, the Editor class provides a required function to send a message to the game board and Board class provides functions to enable editing its squares. In the diagram given below, the player prefers to edit the Joker place as a place for suspension like “Jail” as well as a place for money punishment. Which means, during the game, if a player lands on this Joker place, he/she has to pay an amount of money and wait for a few turns as a punishment.



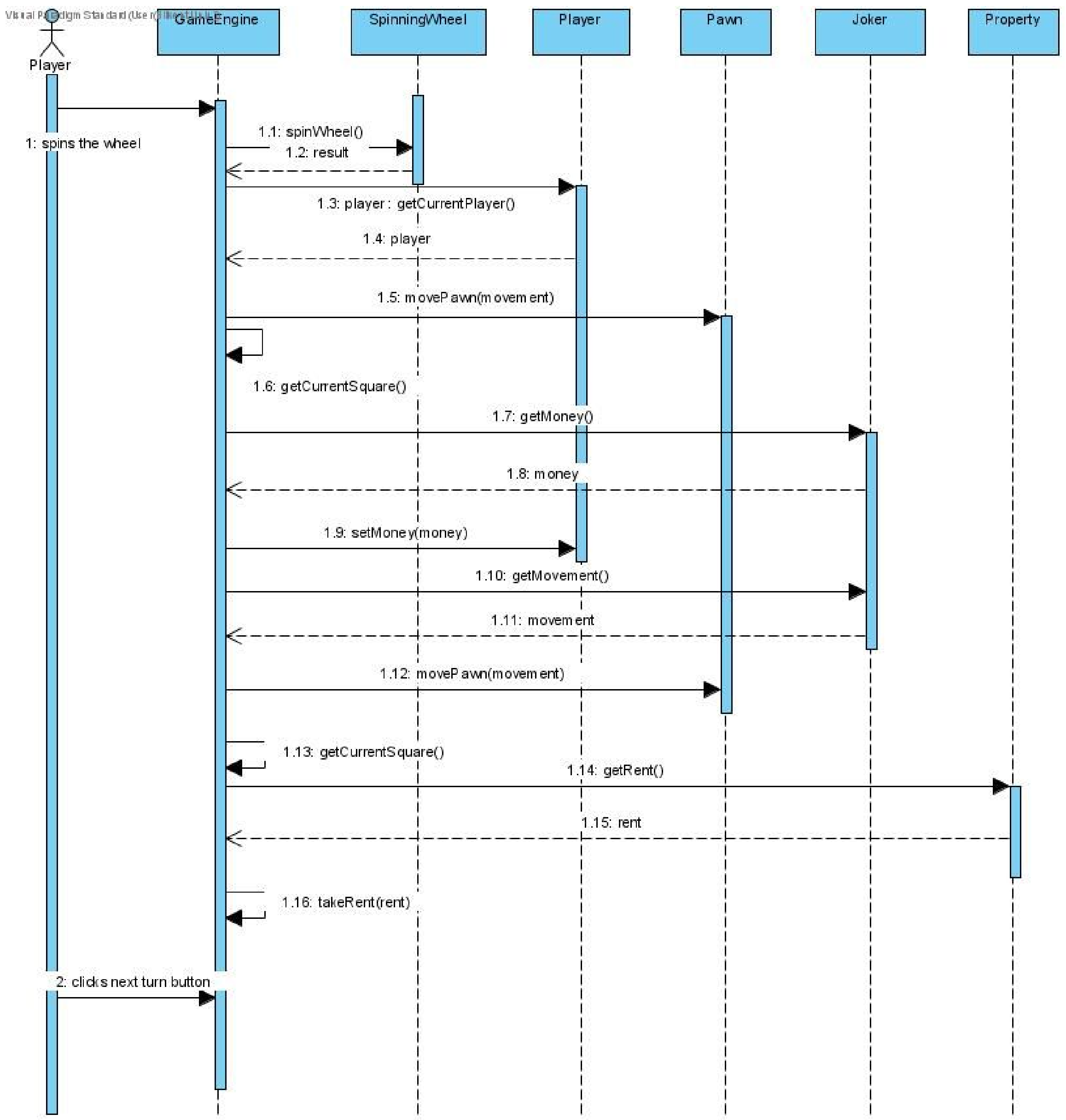
5.2.1.3 Landing on a Property and Buying it

Scenario : During the game, the player spins the spinning wheel when it's his/her turn to play. SpinningWheel class returns the result which is randomly generated between 2-12 to GameEngine. Based on the result, the player's pawn will be moved to the square which is the “result” ahead of the player's current square. After the pawn is moved, assuming the square is a property square which hasn’t been bought yet, the player will be able to buy it. Following that, the property's current owner will be updated as the player by Property class and player’s money will be updated by the message that Game Engine sends to the Player lifeline. At last, the player will press the next turn button which will send a message to GameEngine in order to continue the game.



5.2.1.4 Landing on a Joker Place and Paying Rent

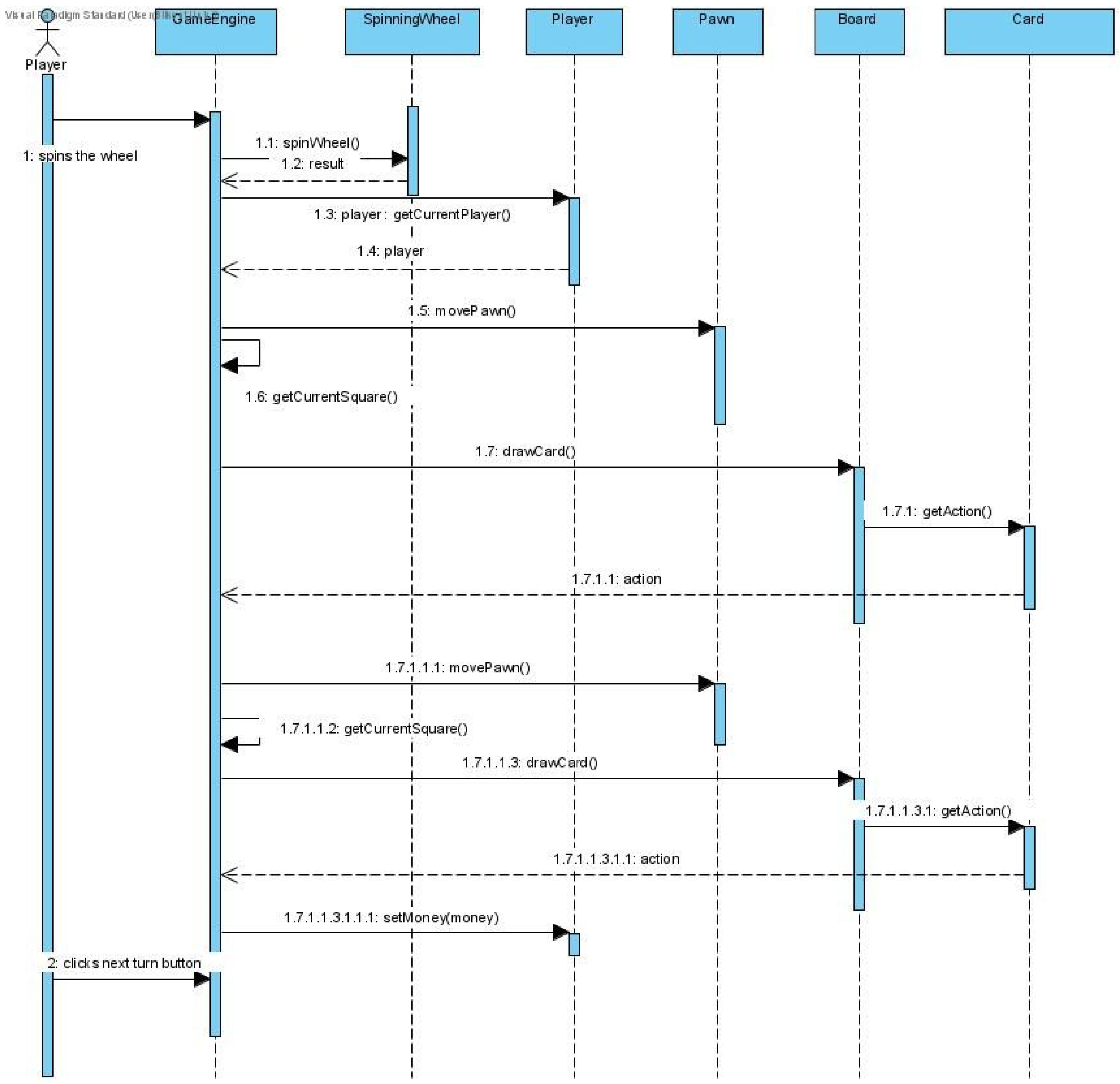
Scenario : In this scenario, the player spins the spinning wheel and based on the result that the spinning wheel generated, the player's pawn is moved to a Joker place by GameEngine. The Joker place that the player landed on requires moving a few squares ahead and earning money as prize. Joker class returns the money prize to GameEngine and Game Engine sends a message to Player class in order to update player’s money. Following that, Joker place returns the amount of squares that the player’s pawn will be moved ahead to the GameEngine. The pawn is moved to a new square by the message that GameEngine sends to the Pawn class. The final square that the player's pawn lands on is a property which is owned by other players. Therefore, the player has to pay rent which is done by the message that GameEngine sends to the Player class. At last, the player presses the next turn button in order to continue the game.


Visual Paradigm Standard (User(Bilkent Univ.))
Player
GameEngine
SpinningWheel
Player
Pawn
Joker
Property
1: spins the wheel
1.1: spinWheel()
1.2: result
1.3: player : getCurrentPlayer()
1.4: player
1.5: movePawn(movement)
1.6: getCurrentSquare()
1.7: getMoney()
1.8: money
1.9: setMoney(money)
1.10: getMovement()
1.11: movement
1.12: movePawn(movement)
1.13: getCurrentSquare()
1.14: getRent()
1.15: rent
1.16: takeRent(rent)
2: clicks next turn button

5.2.1.5 Landing on Community Chest and Chance Squares

Scenario : In this scenario, first the player lands on Community Chest square then, lands on Chance square.

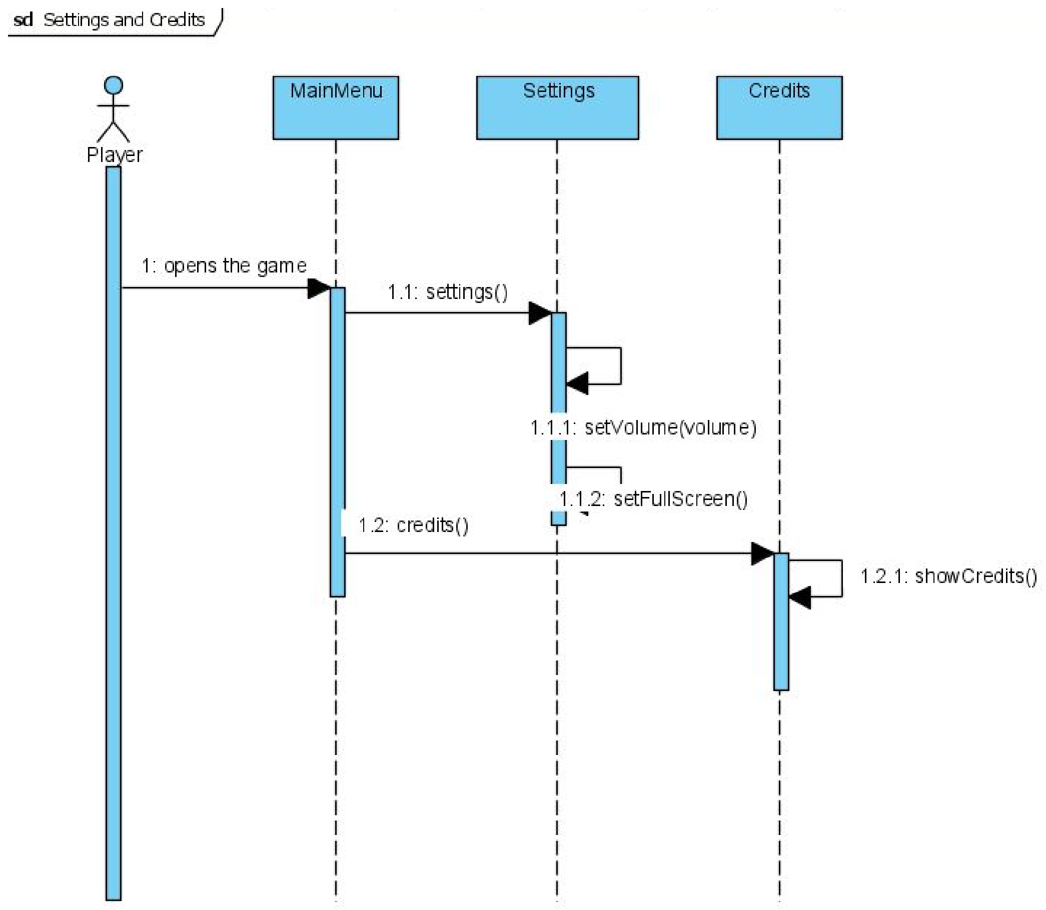
First, the player spins the wheel and according to the result that the spinning wheel generated, the player's pawn is moved to a Community Chest square by the Game Engine. Then the Community Chest card is drawn and Card lifeline receives a message to return the action required on the drawn card to Game Engine. According to the drawn card, the Game Engine needs to move the player's pawn a few squares back, however, after the pawn is moved back, it lands on a Chance square. Chance card is drawn again by the Game Engine and it gives money to the player as a prize. At last, player’s money is updated according to the prize by the message that Game Engine sends to the Player class. Then, the player clicks the next turn button in order to continue the game.


Visual Paradigm Standard (User(Billy(Hill)))
Player
GameEngine
SpinningWheel
Player
Pawn
Board
Card
1: spins the wheel
1.1: spinWheel()
1.2: result
1.3: player : getCurrentPlayer()
1.4: player
1.5: movePawn()
1.6: getCurrentSquare()
1.7: drawCard()
1.7.1: getAction()
1.7.1.1: action
1.7.1.1.1: movePawn()
1.7.1.1.2: getCurrentSquare()
1.7.1.1.3: drawCard()
1.7.1.1.3.1: getAction()
1.7.1.1.3.1.1: action
1.7.1.1.3.1.1.1: setMoney(money)
2: clicks next turn button

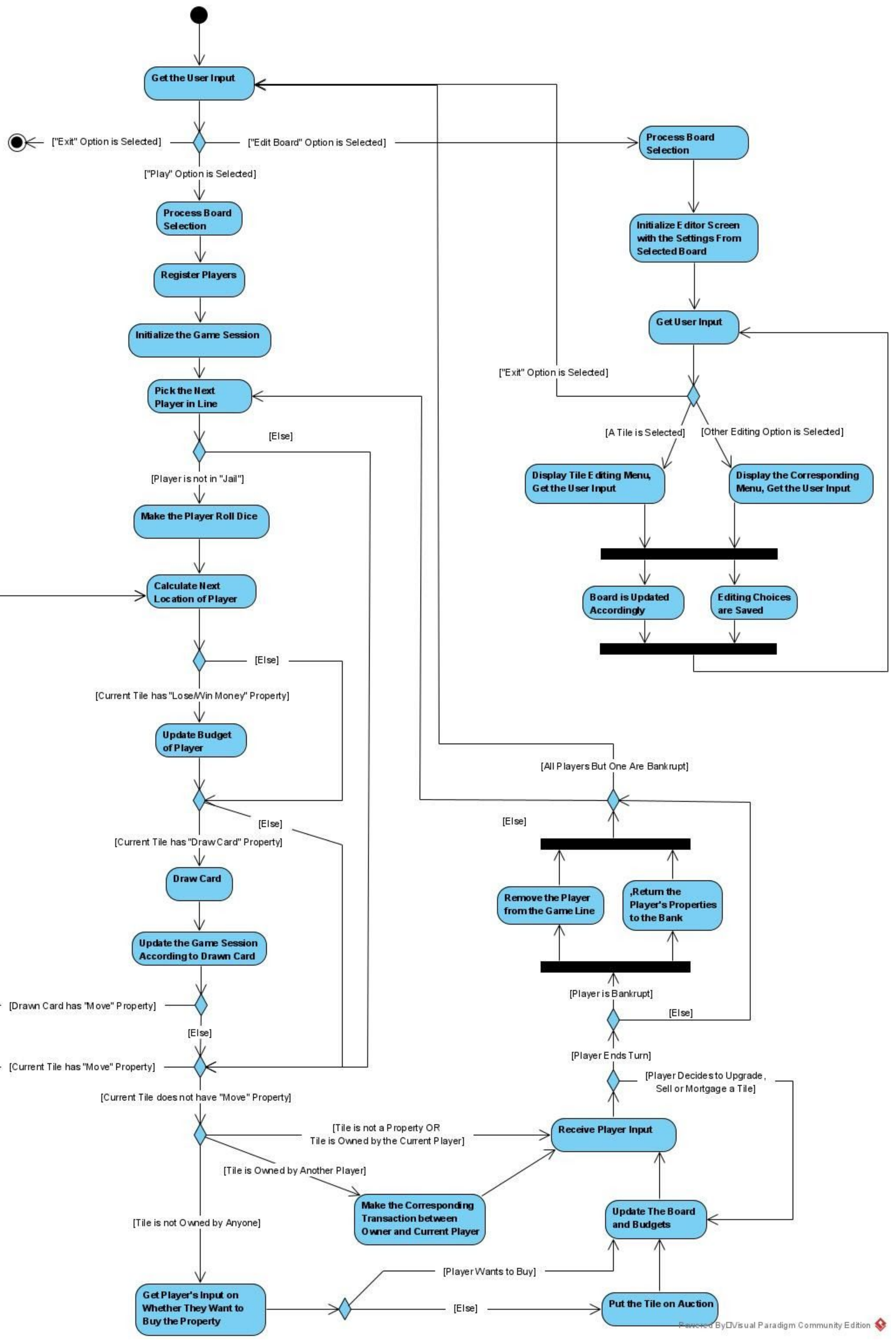
5.2.1.6 Changing Settings and Showing Credits Page

Scenario : In this scenario, the player changes the screen size and volume of the game, then opens the Credits page in order to see credits. First, the player opens the game and the main menu appears. Following that, the player presses the settings button and sends the Settings class a message to show settings. In the Settings page, the player changes volume level and screen size to full screen. At last, the player comes back to the main menu and presses the credits button in order to see developer credits. After the button is pressed, the main menu sends a message to Credits class to display its content.

There are other buttons in the main menu such as the help button which explains game play and rules. That scenario is not given in this report, however, it has the same logic as showing the credits page. The only difference is in 1.2, the message is called help() and it is sent to Help lifeline. Then, the Help class returns a self message in order to show its content like in message 1.2.1 in the diagram below.



5.2.2 Activity Diagram


Get the User Input
["Exit" Option is Selected]
["Edit Board" Option is Selected]
["Play" Option is Selected]
Process Board Selection
Register Players
Initialize the Game Session
Pick the Next Player in Line
[Else]
[Player is not in "Jail"]
Make the Player Roll Dice
Calculate Next Location of Player
[Else]
[Current Tile has "Lose/Win Money" Property]
Update Budget of Player
[Else]
[Current Tile has "Draw Card" Property]
Draw Card
Update the Game Session According to Drawn Card
[Drawn Card has "Move" Property]
[Else]
[Current Tile has "Move" Property]
[Current Tile does not have "Move" Property]
[Tile is not a Property OR Tile is Owned by the Current Player]
[Tile is Owned by Another Player]
[Tile is not Owned by Anyone]
Make the Corresponding Transaction between Owner and Current Player
Get Player's Input on Whether They Want to Buy the Property
[Player Wants to Buy]
[Else]
Put the Tile on Auction
Update The Board and Budgets
Receive Player Input
[Player Decides to Upgrade, Sell or Mortgage a Tile]
[Player Ends Turn]
[Else]
[Player is Bankrupt]
Remove the Player from the Game Line
,Return the Player's Properties to the Bank
[Else]
[All Players But One Are Bankrupt]
Process Board Selection
Initialize Editor Screen with the Settings From Selected Board
Get User Input
["Exit" Option is Selected]
[A Tile is Selected]
[Other Editing Option is Selected]
Display Tile Editing Menu, Get the User Input
Display the Corresponding Menu, Get the User Input
Board is Updated Accordingly
Editing Choices are Saved
Powered By Visual Paradigm Community Edition

At the start, the system is at the main menu waiting for user input. In the main menu, the user can choose to exit, play a game or edit a board. If the user selects exit, the program will terminate. Other than exit there are the options to “play a game” and “edit a board”.

5.2.2.1 Play a game

If the user selected “play a game” a board selection menu will appear displaying available boards. Then the user can select a board to play on and enter the players’ names. Then the system initializes a game session with the given player names and selected board settings. After that the following game loop keeps happening until there is only one player that is not bankrupt:

System selects the next player in line. If the player is not in jail, they spin the spinning wheel. System calculates the player's position after spinning the wheel and processes the requirements of the newly entered square. This possibly includes changing the player’s funds and drawing a chance card. If the new square or the drawn chance card requires movement, a new location is calculated and this process repeats. If there is no requirement to move again we continue. At this point, the player will be standing on a square and there are four cases:

Case 1: Square is not a property. Nothing happens.
Case 2: Square is a property and is owned by the player. Nothing happens.
Case 3: Square is a property and is owned by another player. The player’s funds are changed accordingly.
Case 4: square is a property and is not owned by anyone. The player is asked if they want to buy the square. If the answer is yes the square is now owned by the player and their funds are adjusted accordingly. The board gets updated based on the new owner.

Exactly one of these cases is processed. Then the system waits for player input (this is also where a player that was in “jail” would start instead of spinning the wheel). If the input is upgrading or downgrading a square the board is updated accordingly and the system waits for player input. Else the input is “end turn”. If the player has negative funds when ending a turn, the player is considered bankrupt and removed from the line. The board is updated accordingly. Then there are two cases:

Case 1: There are more than one non-bankrupt players. Then we pick the next player in line and go to the beginning of the game loop.
Case 2: There is only one player left who is non-bankrupt. Then the game ends and the system goes back to the main menu after showing the game results.

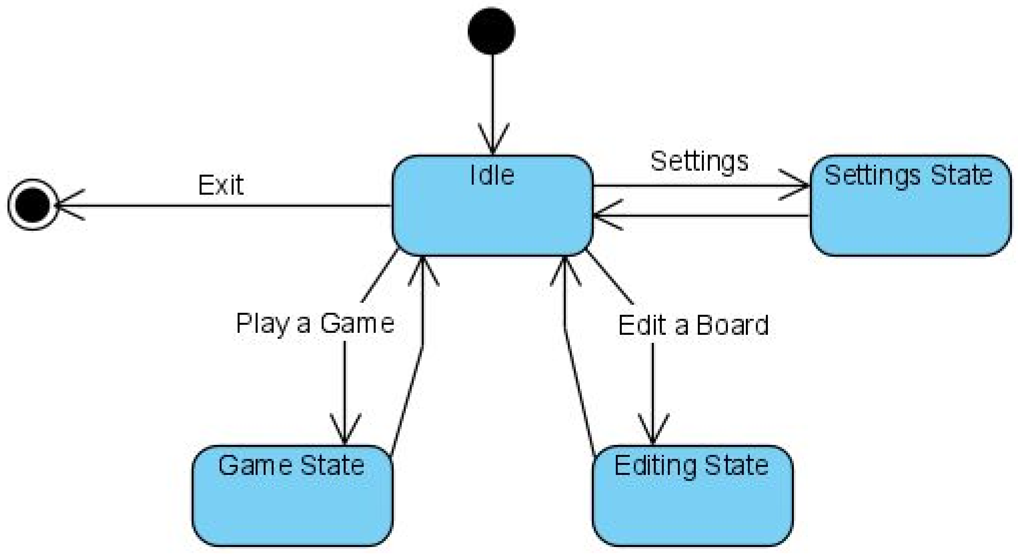
5.2.2.2 Edit a board

If the user selected “edit a board” a board selection menu will appear displaying available boards. Then the user can select a board to edit or “create a new board”. Creating a new board will create a copy of the default board for the player to edit. The system initializes the board editing menu based on the settings of the selected board. Then the player can keep making changes to the board until the following loop ends:

System gets user input. There are three cases:
Case 1: The player selected a square. The menu to edit that square’s settings appears. The user edits some of those settings. The changes are applied and the board is updated accordingly. System goes back to waiting for input.
Case 2: The player selected some other setting (like the name of the currency or chance deck). The menu to edit that specific setting appears if it exists. No menu might be needed for simple edits like changing the name of the currency. The user edits the desired setting. The changes are applied and the board is updated accordingly. System goes back to waiting for input.
Case 3: The input is to exit. Then the loop ends and the system opens the main menu.

5.2.3 State Diagram

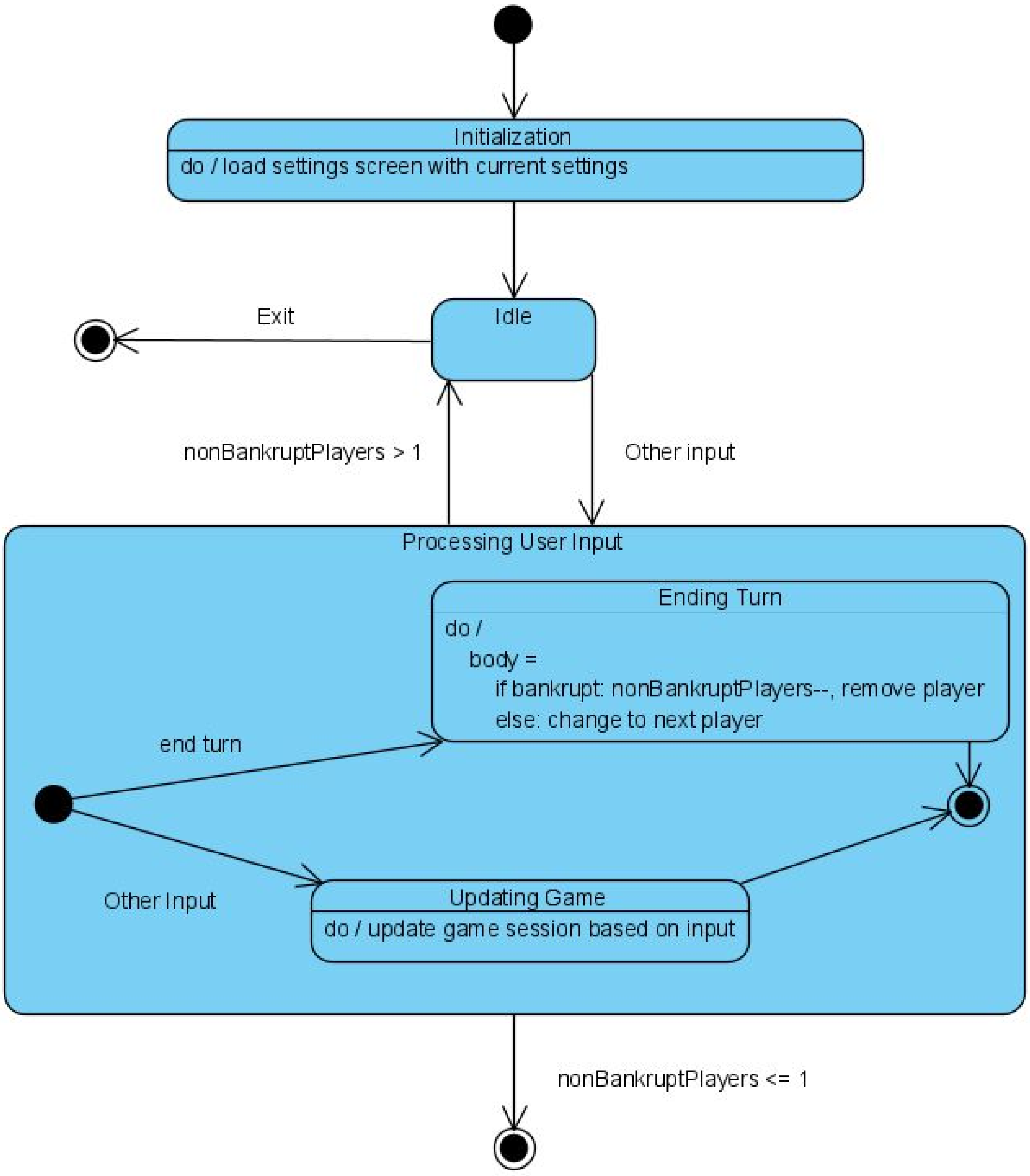
5.2.3.1 Main state diagram



The main state diagram is quite simple since we made state diagrams that show in detail what its components do. From the initial state, the main menu is loaded and the system waits for input in the idle state. When the system gets an input, it changes to a new state as shown in the figure. If the input is exit, then the next state is the final state. Otherwise there are three possible inputs leading to three different states as listed below.

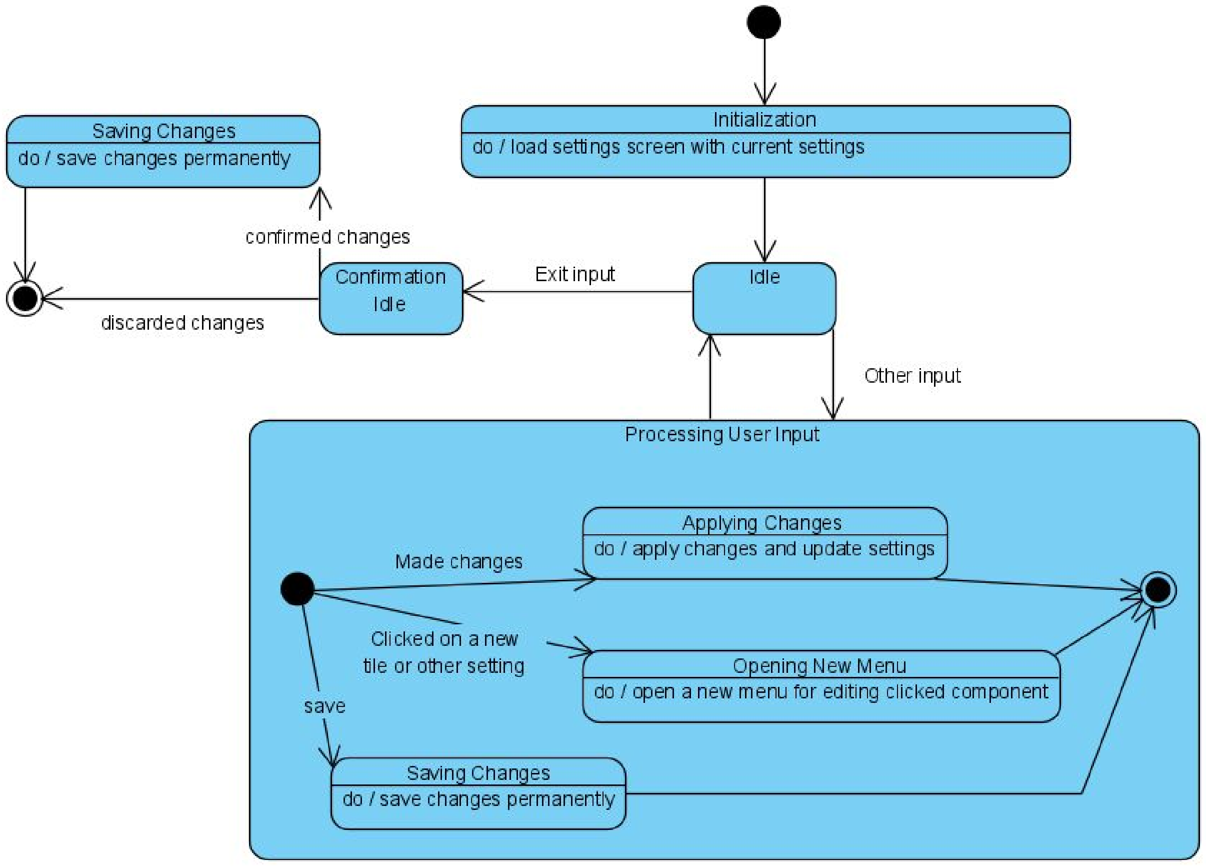
5.2.3.1.1 Game State

In this state, a new game session is initialized with the given settings (selected board and player names). Then the system is idle waiting for player input. If the input is "exit", the system moves to the final state (meaning we return to the idle state in the main state diagram). If the input is something other than exit, the system moves to "processing user input" state. Here if the input is end turn the "Ending Turn" state processes the end of turn as shown in the image. If the input is something other than "end turn" the "Updating Game" state processes the input and updates the game session accordingly. Then the "Processing user input" state ends and the system goes back to "idle" state waiting for user input.


Initialization
do / load settings screen with current settings
Idle
Exit
nonBankruptPlayers > 1
Other input
Processing User Input
Ending Turn
do /
body =
if bankrupt: nonBankruptPlayers--, remove player
else: change to next player
end turn
Other Input
Updating Game
do / update game session based on input
nonBankruptPlayers <= 1

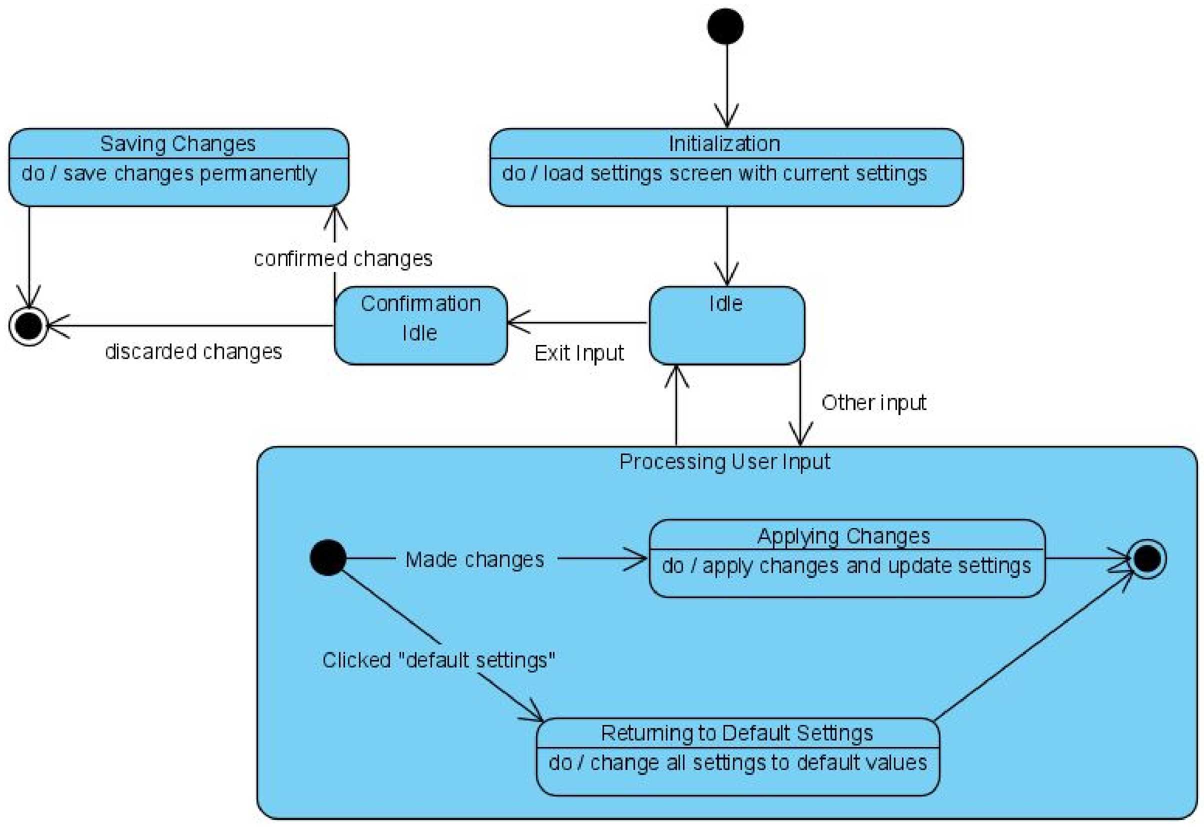
5.2.3.1.2 Editing State

This state is quite similar to the game state. The differences are in the “processing user input” state. Here there are three different possible inputs leading to three different possible sub-states as shown in the figure. Then the “processing user input” state finalizes and the system goes back to “idle” state waiting for user input. The other difference from game state is the confirmation sequence after the “exit” input. Instead of directly finalizing, this time system waits for another input to confirm if the user wants to permanently save the changes.

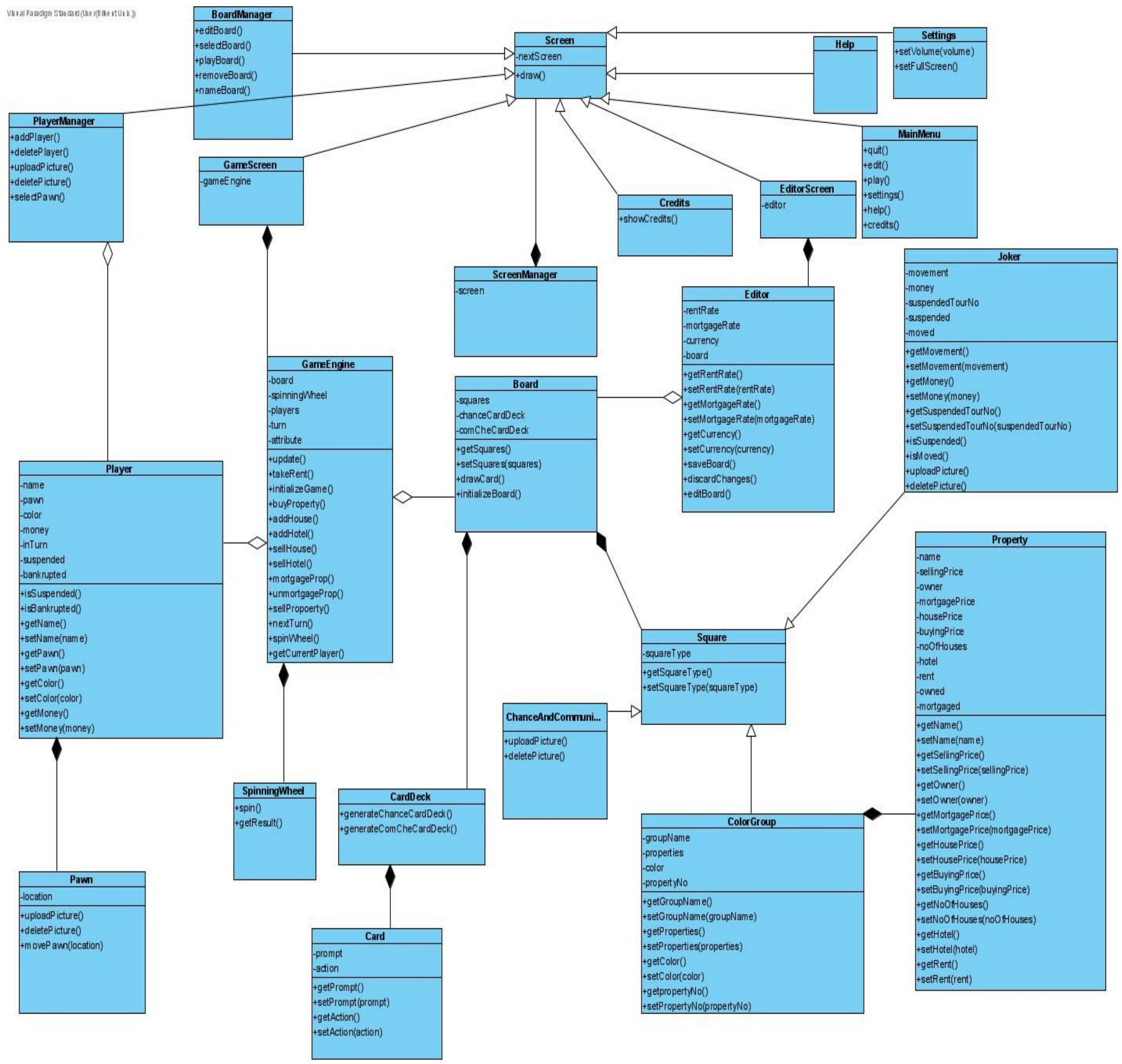

Initialization
do / load settings screen with current settings
Saving Changes
do / save changes permanently
confirmed changes
Confirmation Idle
Exit input
Idle
discarded changes
Other input
Processing User Input
Applying Changes
do / apply changes and update settings
Made changes
Clicked on a new tile or other setting
Opening New Menu
do / open a new menu for editing clicked component
save
Saving Changes
do / save changes permanently

5.2.3.1.3 Settings State

This diagram is very similar to the editing diagram state with the only difference being in the “processing user input” state. This time we have the same “made changes” option and the “applying changes” state from the previous section but the other two options are missing. There is a new option for handling “default settings” input. This input leads to the “Returning to default settings” setting where all settings are reverted to their default values. When this is done, the “processing user input” state ends.


Saving Changes
do / save changes permanently
Initialization
do / load settings screen with current settings
confirmed changes
Confirmation Idle
Idle
discarded changes
Exit Input
Other input
Processing User Input
Made changes
Applying Changes
do / apply changes and update settings
Clicked "default settings"
Returning to Default Settings
do / change all settings to default values

5.3 Object and Class Model



MainMenu: MainMenu class is defined for the main menu in the game which will be automatically opened when players open the game. It will provide functions for playing the game, editing the game, settings, quitting the game, showing the help page and showing credits page.

Settings: Settings class is defined for the system settings such as volume or entering/exiting fullscreen size in the game.

Help: Help class is defined for the help page where game rules and game play is described.

Credits: Credit class is defined for the credits page in the game. It only displays developer credits.

Editor: Editor class is defined for the editing features in the game. It has aggregation relationships with EditorScreen and Board classes and it provides functions for editing the game.

Board: Board class is defined for the game board in the Monopoly game. It interacts with many classes such as Editor, CardDeck, Square and GameEngine. It contains squares such as properties and chance and community chest cards. It also has chance and community chest card decks.

BoardManager: BoardManager class is defined for managing Monopoly boards in the game. In the game, players will be able to edit the given game template. Furthermore, players will be able to save their edited templates on a page. This class will provide functions for naming these edited templates as well as selecting them from the page to play.

PlayerManager: PlayerManager class is defined for a “manager” between Screen and Player classes. It will provide functions for adding a player to the game, deleting a player, updating player photos and selecting the token of a player.

Player: Player class is defined for the players in the game. It will provide functions for setting a pawn to the player. Furthermore, it will provide information about the player. For example, it will keep the name, the amount of money that player has and associated color of a player. It will also check whether it's that player’s turn in the game or whether the player is bankrupt or not.

GameEngine: GamEngine class is defined for the game engine of the game. It will be the general controller of the game by handling most of the features in game play and controlling interactions between many classes. Furthermore, it will provide functions for buying/selling houses and properties, updating game screens as well as controlling player turns.

GameScreen: GameScreen class is defined in order to create a gate between GameEngine and Screen classes.

EditorScreen: EditorScreen class is defined in order to create a gate between Editor and Screen classes.

SpinningWheel: SpinningWheel class is defined for the spinning wheel in the game which is designed for the “rolling dice” feature in Monopoly. Instead of rolling dice, players will spin a spinning wheel, and based on the results, their pawns will be moved automatically by the game. SpinningWheel class has a strong aggregation relationship with GameEngine class and it will provide functions for spinning the spinning wheel and returning its results to GameEngine.

Pawn: Pawn class is defined for tokens or in other words, pawns in the Monopoly game. It has a strong aggregation relationship with the Player class. The most important function that this class will provide is a function which moves the pawns of the players on the game board.

Square: Square class is defined for the squares in the Monopoly board. In our Monopoly, there are three three types of squares such as ColorGroup, Joker and ChangeandCommunity which are each defined as subclasses of Square class. It will also provide functions for editing the square types in the game.

ColorGroup: ColorGroup class is a subclass of Square superclass which is defined for the “color-grouped properties” in Monopoly. There will be 8 default color groups for properties in the game like in the Monopoly board game. However, players will be able to change the number of color groups, the number of properties inside color groups or the color of these groups and ColorGroup class will provide functions for these changes.

Property: Property class is defined for the properties in the Monopoly game. It has an aggregation relationship with ColorGroup class. It also handles editing Property squares. For example, if players prefer to change the features of a property such as its name, price, rent price, house price and hotel price, Property class will handle required functions.

Joker: Joker class is another subclass of Square superclass and it is defined for the Joker places in the game. If players prefer to edit the game, they can change the actions in Joker places. For example, if a Joker Place requires waiting for 3 turns, players can change that action into moving 3 places ahead and Joker class will provide required functions for editing Joker places.

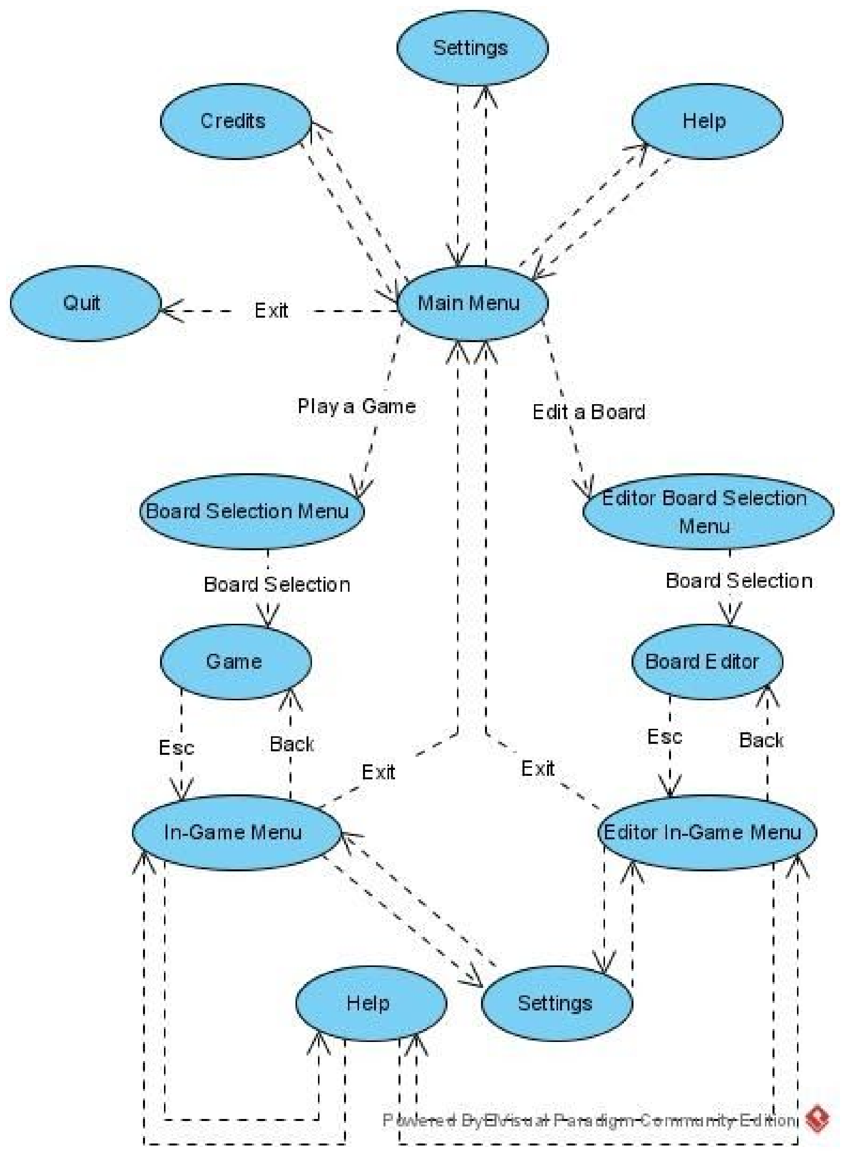
ChanceandCommunity: ChanceandCommunity class is the last subclass of Square superclass. This class is defined for the chance and community chest squares in the game. The only editable feature of these squares is the background picture of them. Therefore, this class will only provide functions for editing pictures for editing purposes.

CardDeck: CardDeck class is defined for the Chance and Community Chest card decks in the Monopoly game and it will provide functions for generating these decks.

Card: Card class is defined for Chance and Community Chest cards in the Monopoly game. It has a strong aggregation relationship with CardDeck class and it will keep the prompt and required action for a card.

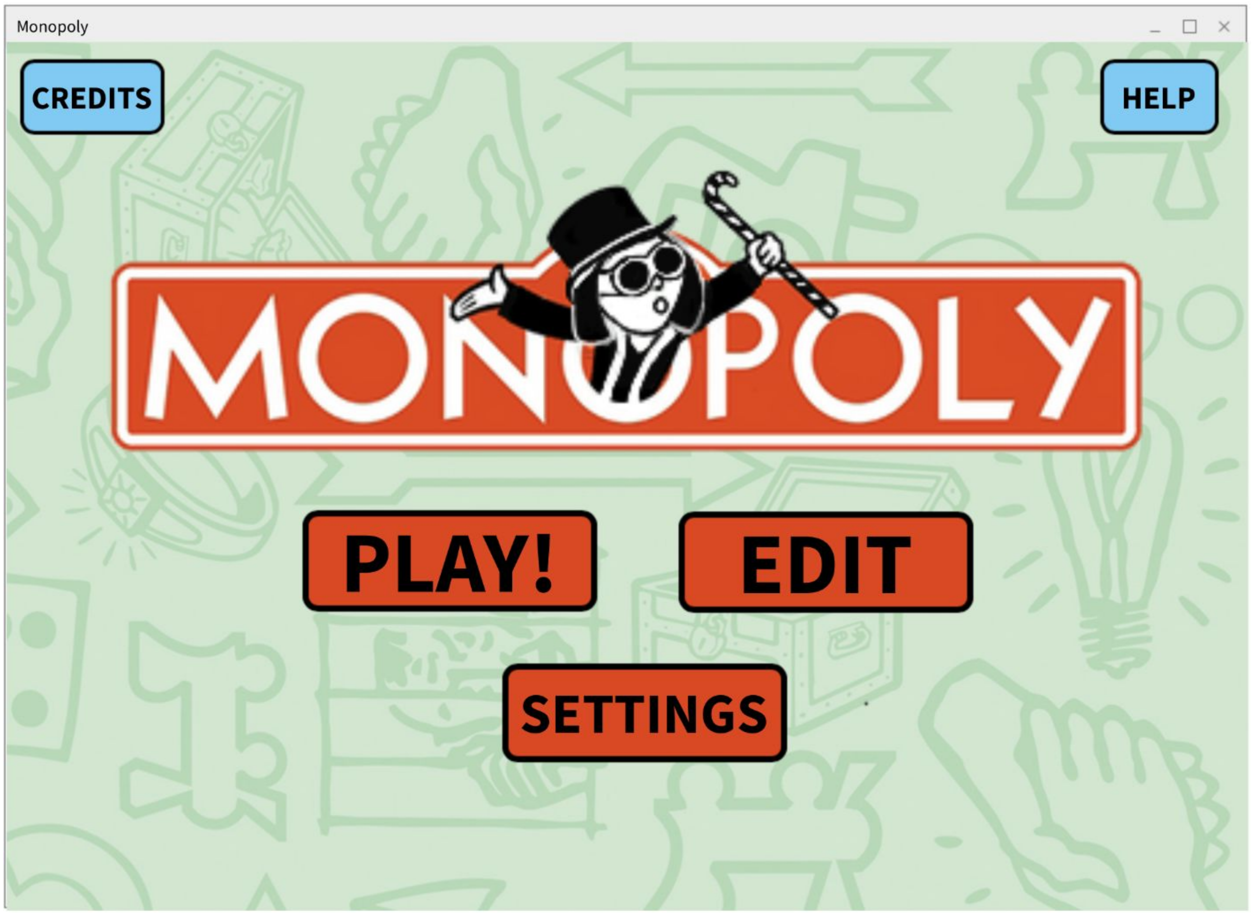
5.4 User Interface

5.4.1 Navigation Path



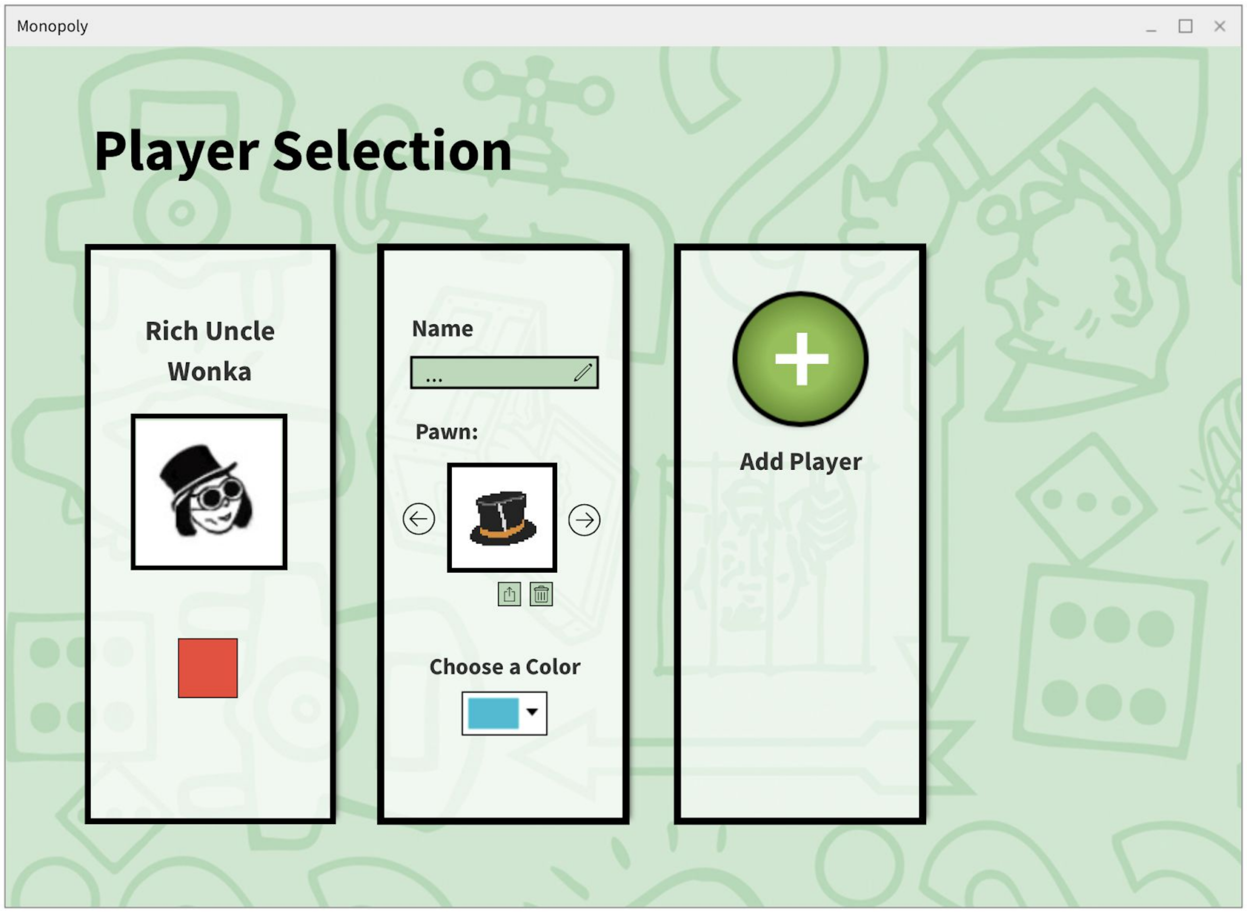
5.4.2 Screen Mockups

5.4.2.1 Main Menu



This is the main menu screen of our Monopoly. It is also the first screen the players will come across with when they open the game. From this window, players can either play the game directly by pressing the “Play!” button, or they can move on to the edit screen by pressing the “Edit” button. Apart from these, there are also 3 smaller buttons named “Settings”, “Credits”, and “Help”. By pressing the settings button, players will reach the setting screen where they can adjust game volume or enter or exit fullscreen mode. The credits button will open the credits screen where users can find information about the game and its developers, us. Finally, the help button will take the users to a manual text about how to play this Monopoly game.

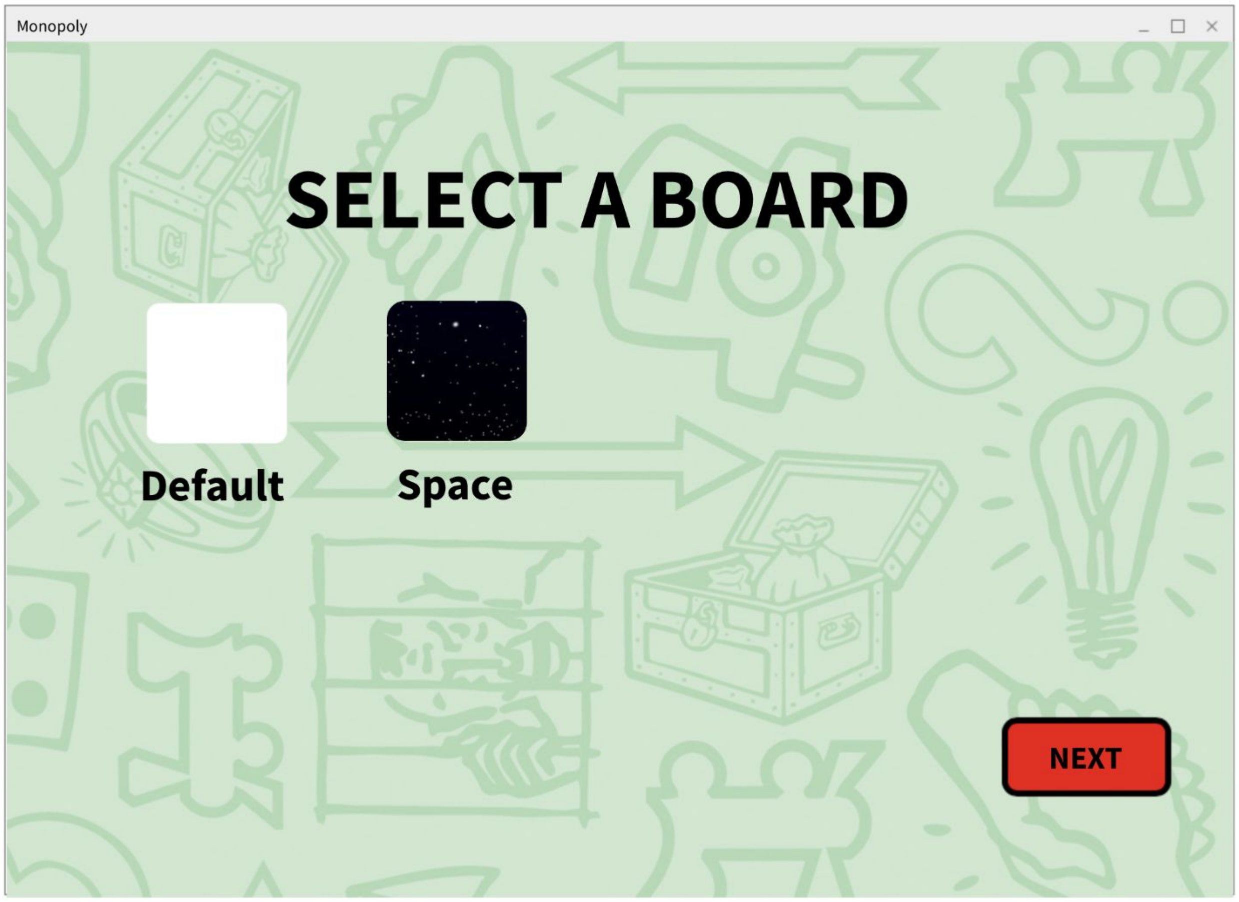
5.4.2.2 Player Selection Screen



In this screen, the players of the game will be added. By pressing the green add player button users will be able to add up to 4 players. The leftmost subscreen shows the profile of a player named “Rich Uncle Wonka”. The red rectangle represents the color the player chooses which will indicate his/her properties on the game board.

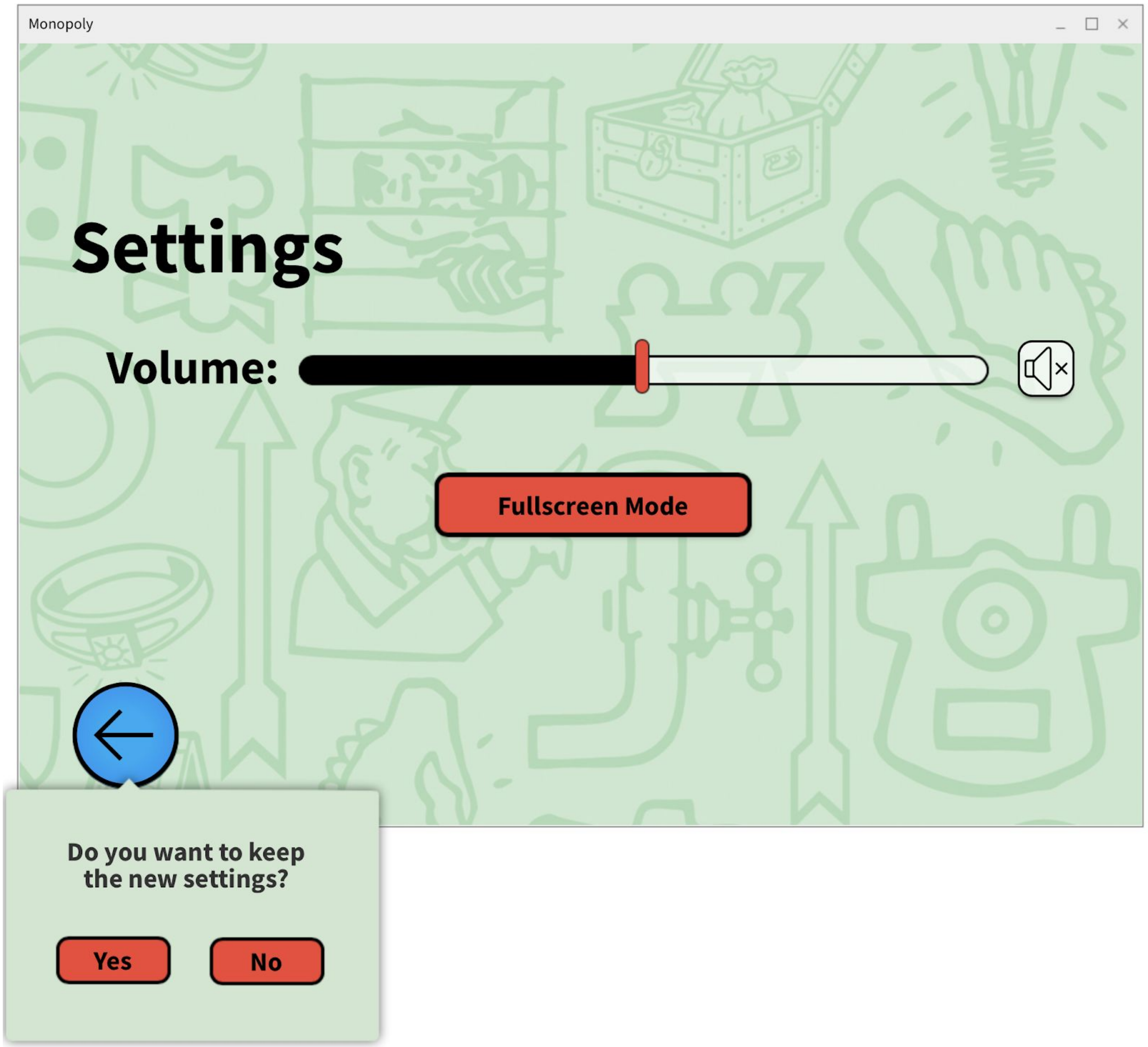
The subscreen in the middle shows an unfinished profile. The player will enter his/her name on the text box under the “Name” title. Then, players can upload or delete their profile pictures using the buttons on the bottom of the big picture frame. After that, users can set their pawns by choosing preset pawns by browsing them using the arrows located on both sides of the pawn picture or like the profile picture, users can add their own pawns to the game.

5.4.2.3 Board Selection Menu



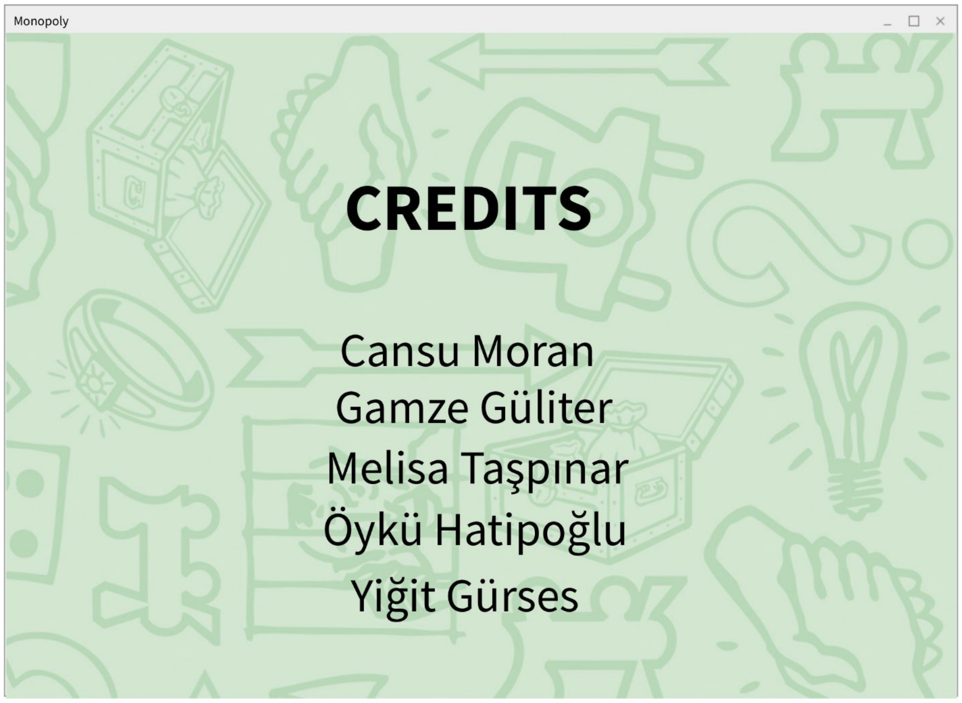
Board Selection Menu appears right after the user chooses whether they would like to play the game or edit a board. In both scenarios, Board Selection Menu comes and displays the saved boards. From this menu, the user can choose which board they would like to edit or play. During the first run, the Board Selection menu only has the default board. The users can edit the default board according to their preferences and save their changes as a new board or they can save it on top of the default board. In this mock-up screen, there are two boards, Default board and Space board. In this scenario, the user had already edited the default board and saved the changes as a new board, called “Space”.

5.4.2.4 Settings Screen



This screen shows the settings of the game. Players can adjust the volume using the slider or they can enter the fullscreen mode using the "Fullscreen Mode" button. When pressed, this button turns into a "Exit Fullscreen Mode" button. Additionally, when the player presses the arrow at the bottom left of the screen, the player can exit the settings screen after he/she answers the confirmation window that pops up.

5.4.2.5 Credits Screen



CREDITS

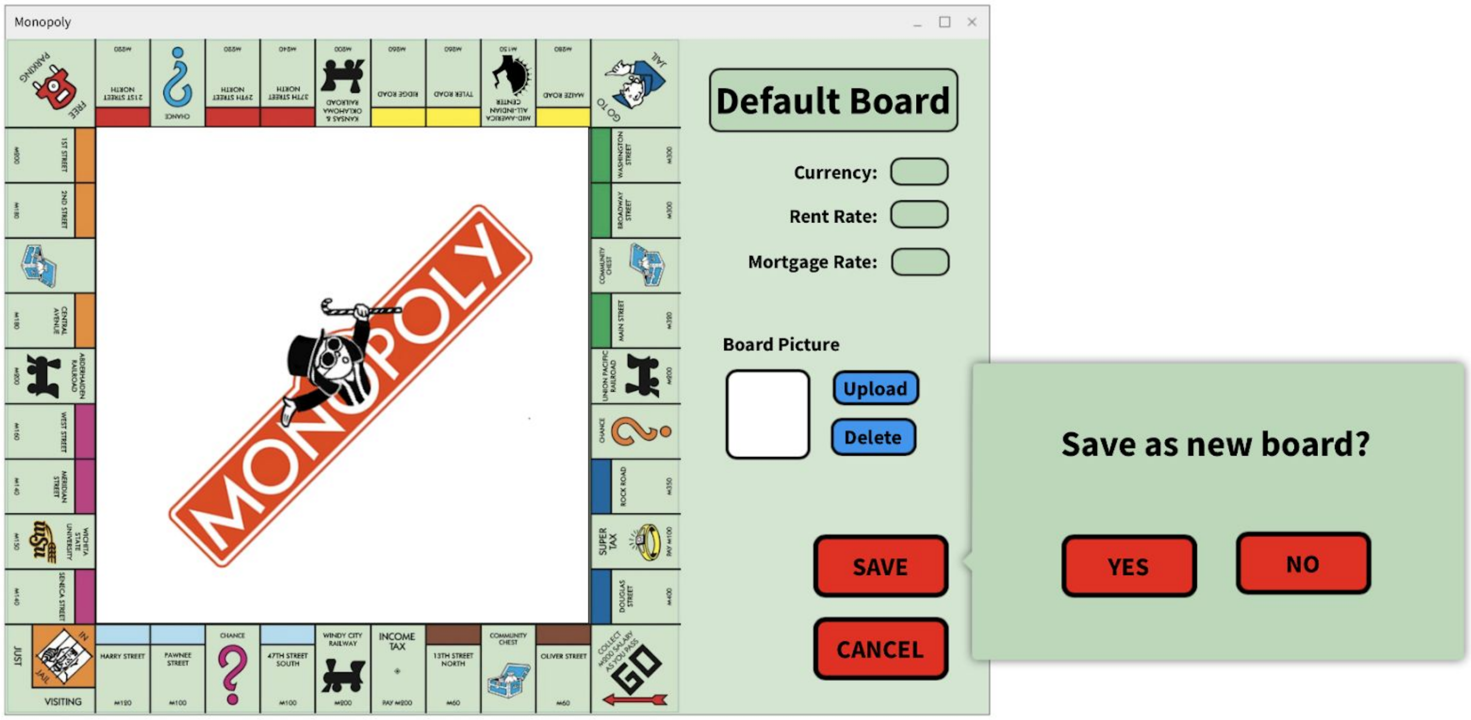
Cansu Moran
Gamze Güliter
Melisa Taşpınar
Öykü Hatipoğlu
Yiğit Gürses

Credits screen displays the names of the creators of the game.

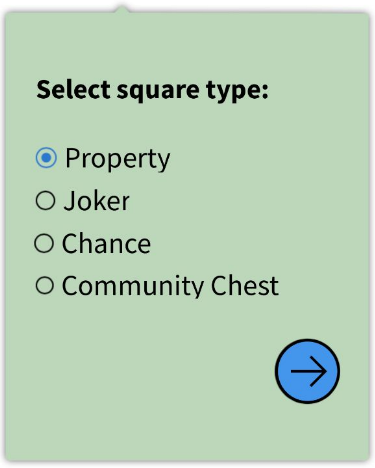
5.4.2.6 Help Screen

Help screen will contain an informative text about how to play the game and how to edit a board.

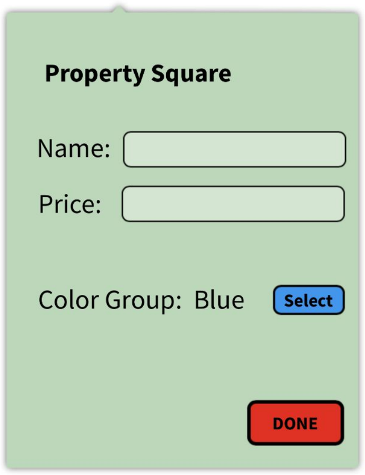
5.4.2.7 Edit Screen

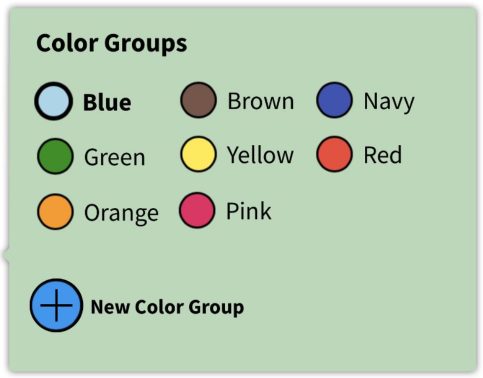


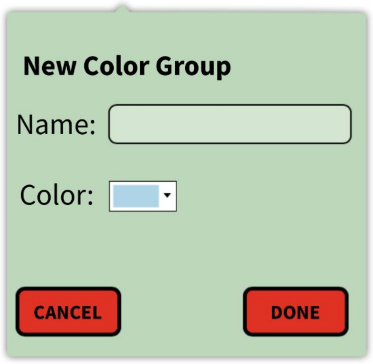
Edit screen is where the user can edit the board they have selected from the Board Selection Menu. Information such as the name of the board, the currency name and rent and mortgage rate on the game can be edited directly from the Edit screen, as displayed above. The user can also add a board picture, which will be placed in the middle of the board behind the Monopoly logo, and will be used as the icon of the board on the Board Selection Menu. To edit the squares, the user should click on the square that they would like to edit and a pop-up screen will appear. After completing all the changes, the user can save the board as a new board, or they can save it on top of the board that they have selected.



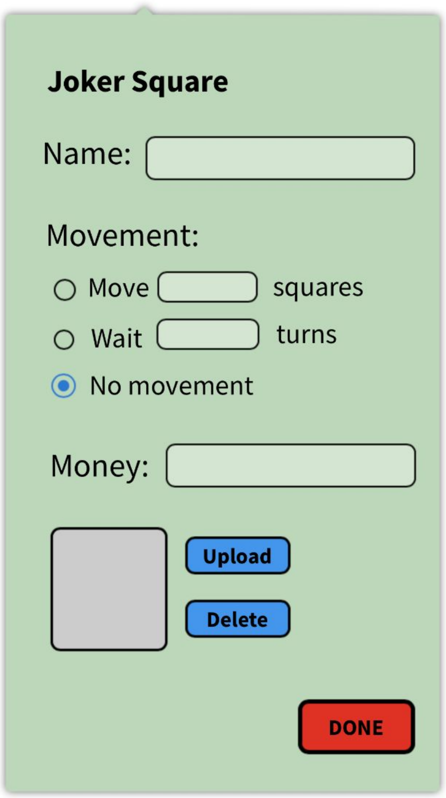
When a user selects a square to edit, a pop-up screen will appear. This screen will ask the user to select a square type. A square can be one of four square types: Property square, Joker square, Chance square and Community square. After selecting the square type, the user can click on the arrow button to further edit the selected square, according to the square type.



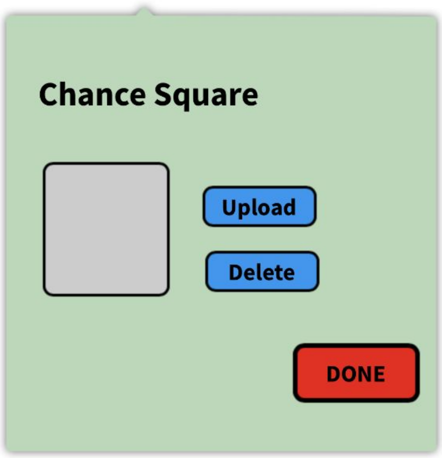


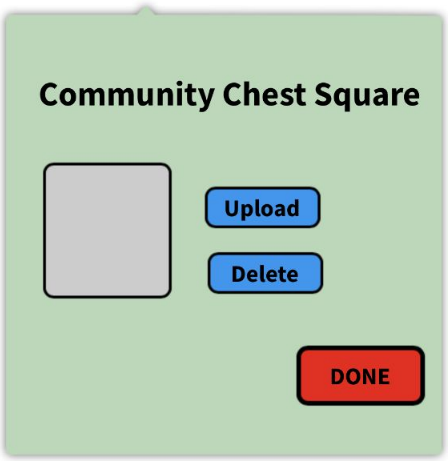


If the selected square type is a Property square, the property edit pop-up screen will come. In this screen, the user can edit the name of the property as well as the selling price and color group of that property. If the user would like to change the color group of the property, they can press the "Select" button which will take them to another pop-up screen that will display all the color-groups currently existing on the game. The user can either choose from one of those pre-existing color groups or they can create a new color group by clicking on the plus icon, which will make the new color group generation screen appear. On this screen, the user can name their new color group and select a color for the group that will be displayed on the board.



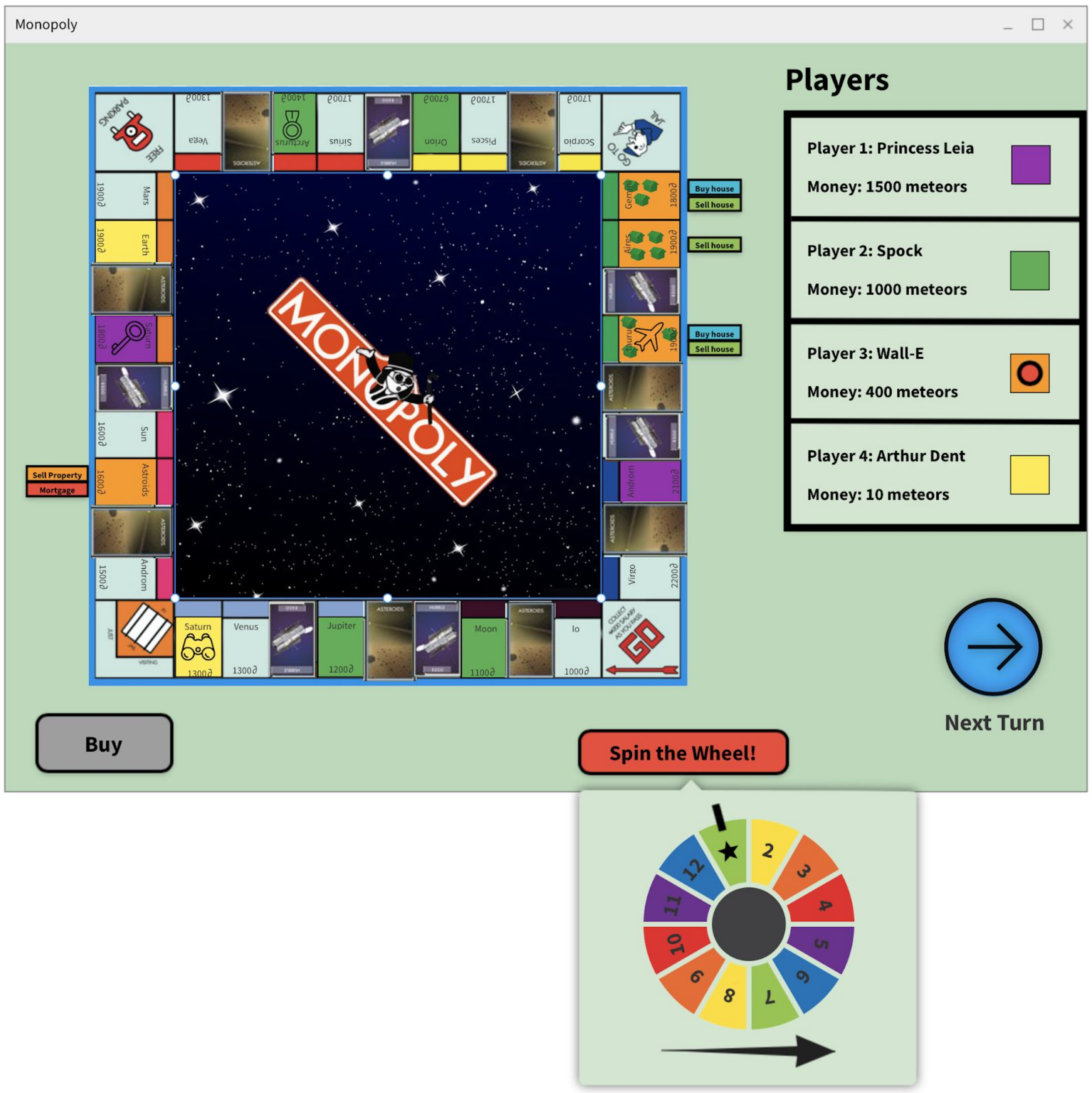
If the selected square is a Joker square, then Joker edit pop-up screen will appear. The user can edit the name and the picture of the square from this screen. In addition, they can edit the specific movement and money actions that this screen will have. For the movement, the user can pick whether they would like to make pawns that land on this square move a couple squares or make them wait for a couple turns. The user can't pick both options at the same time, however, they can choose not to pick either one of them as well. If they want to add a movement action, after deciding on which action to add, they can also change the number of squares or turns as well. The user can also add money actions to this square by typing the amount of money they want the players who land on this square to gain or lose. If the user doesn't want to add any money actions, they can leave the input box next to the money label empty.





If the selected square is a Chance or Community Chest square, the user can only edit the picture of that square. The cards on the Chance and Community Chest decks will be automatically edited according to the currency entered on the Edit Screen.

5.4.2.8 Play Screen


Monopoly

Players

Player 1: Princess Leia
Money: 1500 meteors

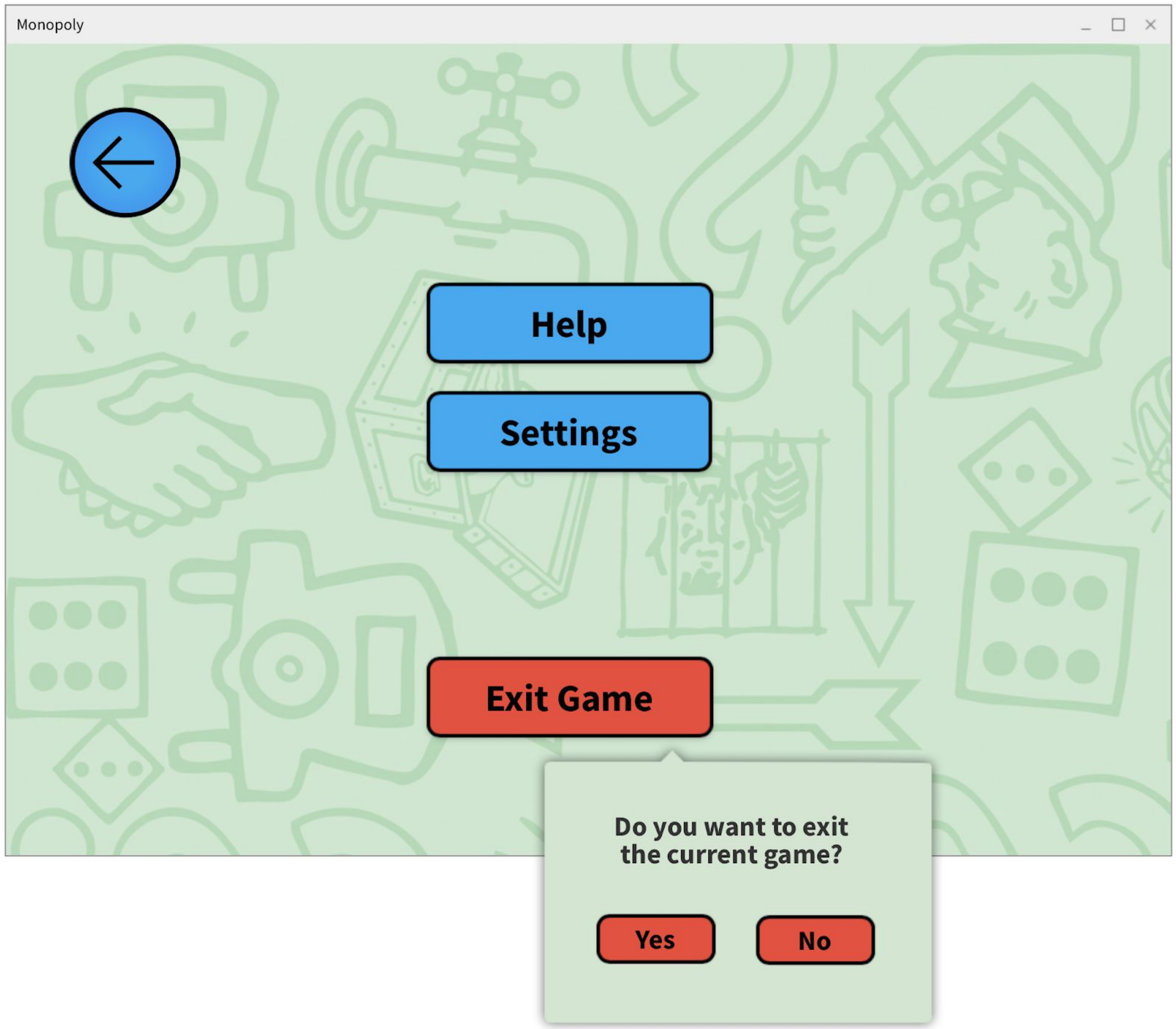
Player 2: Spock
Money: 1000 meteors

Player 3: Wall-E
Money: 400 meteors

Player 4: Arthur Dent
Money: 10 meteors

This screen shows a game in progress. On the board, there are some colored properties which represent the owners of those properties. For example, the yellow properties belong to the player named “Arthur Dent” which can be seen from the player list on the right side of the screen. This list includes the names of the players, their money, and the color they use to mark their properties. Additionally, one can see whose turn it is by looking at the red circle on the colors of players. In this case, since the red is on the orange color, the turn is Wall-e’s. Looking at the board, we can see the green properties which are marked orange. They have plenty of houses on them and since it is Wall-E's turn, next to its properties there are some options such as sell/buy houses or mortgage the property. Since there are 4 houses on the middle red property, the buy house option is not present as a property can have at most 4 houses. Using the "Buy" button on the down-left of the screen, the player can buy the property he/she landed on. Since Wall-E owns the property and therefore cannot buy it, the buy button is inactive. To spin the wheel which has all the outcomes of the sum of 2 dice, the player can click on the "Spin the Wheel!" button, which will open a pop-up that has the spinning wheel. When the player finishes his/her turn, the player will click on the "Next Turn" button.

5.4.2.9 In-game Menu



This menu can be reached while playing the game or editting the board by pressing Esc button ın keyboard. This screen contains "Help" and "Settings" buttons just like the main menu screen.

The screen also has an “Exit game” button which will be “Exit Editing” when this screen is reached from the editing screen. This button will exit the current action (either play game or edit board) after displaying a confirmation window. Additionally, by pressing the arrow on the top left corner, the player can exit the in-game menu.

5.5 Priorities of Requirements

In our project, we decided to group the requirements according to their priorities. Implementing gameplay has higher priority than the editing feature since editing feature is only possible with a working game and without a playable game, no user would prefer our game. In addition, while implementing editing features, implementing editing features to Joker squares will have higher priority than implementing editing features to other squares. Joker places offer a more innovative perspective to the Monopoly game since players will be able to edit both game logic and user interface while editing Joker places. Following the Joker places, implementing editable features to Property squares will have the second highest priority. This includes, editable property square names, editable selling prices and editable color groups. At last, implementing editable features to Chance and Community Chest squares will have the least priority since there are not many editable features that Chance, and Community Chest squares offer.

5.6 References

All the rules referred in our report were based on the official Monopoly rules in "http://www.mtholyoke.edu/~blerner/cs315/Monopoly/MonopolyRules.pdf"