CS 319

Object-Oriented Software Engineering

Project Analysis Report

Monopoly

Group 2-H

Aziz Ozan Azizoğlu - 21401701

Hammad Khan Musakhel - 21801175

Veli Can Mert - 21602394

Gokhan Mullaoglu - 22001086

Mert Sayar - 21601435

1. Introduction

Monopoly is a board that we will implement on a computer. In the game, players roll two six-sided dice to move around the game board, buying and trading properties, and developing them with houses and hotels. Players collect rent from their opponents, with the goal being to drive them into bankruptcy. Money can also be gained or lost through chest cards, and tax squares; players can end up in jail, which they cannot move from until they have met one of several conditions. The game has numerous rules, and hundreds of different editions exist. However, we hope you find this version to be the most fun of all.

2. Overview

Monopoly is a score-limit family game with a modern Graphical User Interface following the modern norms. The game is designed for both computers only. The game will start in full-screen mode. After starting the game the user will be shown a menu screen from where they can decide to choose how many players, see tutorials to know how to play, and quit the application and return to the desktop. The player will be able to pause anytime in-game where they will be shown a menu allowing the user quit option. The aim of the game is to become the wealthiest player through buying, renting and selling property.

2.1. GameBoard

The game board consists of a board with 2 digital dice, tokens, 32 houses and 12 hotels. There are Chance and Community Chest cards. a Title Deed card for each property and play money.

2.2. Setup

Assign players from the menu, at max it could be 8 players. Each player is assigned a number to represent him/her while manopolising around the game board, to keep track of the players. Each player is given $1500. All remaining money and other equipment go to the Bank.

2.3. Transactions

Besides the Bank's money, the Bank holds the Title Deed cards and houses and hotels prior to purchase and use by the players. The Bank pays salaries and bonuses. It sells and auctions properties and hands out their proper Title Deed cards; it sells houses and hotels to the players and loans money when required on mortgages. The Bank collects all taxes, fines, loans and interest, and the price of all properties which it sells and auctions. The Bank never "goes broke."If the Bank runs out of money, the Banker may issue as much more as may be needed by writing on any ordinary paper.

2.4. GamePlay

Each player in turn throws the dice, starting with a random player. The dice is thrown/rolled to generate a number which is the number of cells the player has to go on and stop at the maximum number collectively. After one has completed the play, the turn passes to the second player. The players remain, in the memory, on the spaces occupied and proceed from that point on the player's next turn. Two or more players may rest in the same space at the same time.
According to the space your player reaches, you may be entitled to buy real estate or other properties — or obliged to pay rent, pay taxes, draw a Chance or Community Chest card, "Go to Jail ® ," etc. If you throw doubles, you move your player as usual, the sum of the two dice, and are subject to any privileges or penalties pertaining to the space on which you land. Retaining the dice, throw again and move your token as before. If you throw doubles three times in succession, move your token immediately to the space marked "In Jail"(see JAIL).

2.5. Go

Each time a player lands on or passes over GO, whether by throwing the dice or drawing a card, the player gets himself/herself a $200 salary. The $200 is paid only once each time around the board. However, if a player passing GO on the throw of the dice lands 2 spaces beyond it on Community Chest, or 7 spaces beyond it on Chance, and draws the "Advance to GO" card, he/she collects $200 for passing GO the first time and another $200 for reaching it the second time by instructions on the card.

2.6. Buying property

Whenever a player lands on an unowned property the player may purchase that property from at its selling price. The property has the player's name then, and it's apparent that the player who bought it owns it. We have neglected the auction option in our version as it seems very uneasy to use in practice. By making purchases only, it gets more fixed and competitive.

2.7. Paying Rent

When you land on property owned by another player, the owner collects rent from you in accordance with the rent price. If the property is mortgaged, no rent can be collected.It is even more advantageous to have houses or hotels on properties because rents are much higher than for unimproved properties. The owner may not collect the rent if he/she fails to ask for it before the second player following rolling the dice.

2.8. Chance and Community Chest

When you land on either of these spaces, take the card from the deck indicated, follow the instructions. The “Get Out of Jail Free” card is held until used and then returned to the bottom of the deck. If the player who draws it does not wish to use it, he/she may sell it, at any time, to another player at a price agreeable to both.

2.9. Income Tax

If the player lands here you have two options: You may estimate your tax at $200 and lose it, or you may pay 10% of your total worth to the system. Your total worth is all your cash on hand, prices of mortgaged and unmortgaged properties and the cost price of all buildings you own. You must decide which option you will take before you add up your total worth.

2.10. Jail

Players land in Jail when: (1) the player lands on the space marked “Go to Jail”; (2) you draw a card marked “Go to Jail”; or (3) you throw doubles three times in succession.

When you are sent to Jail you cannot collect your $200 salary in that move since, regardless of where your player is on the board, you must move it directly into Jail. Yours turn ends when you are sent to Jail. If the player is not “sent” to Jail but in the ordinary course of play lands on that cell, you are “Just Visiting,” you incur no penalty, and you move ahead in the usual manner on your next turn. You get out of Jail by: (1) throwing doubles on any of your next three turns; if you succeed in doing this you immediately move forward the number of spaces shown by your doubles throw; even though you had thrown doubles, you do not take another turn; (2) using the “Get Out of Jail Free” card if you have it; (3) purchasing the “Get Out of Jail Free” card from another player and playing it; (4) paying a fine of $50 before you roll the dice on either of your next two turns. If you do not throw doubles by your third turn, you must pay the $50 fine. You then get out of Jail and immediately move forward the number of spaces shown by your throw. Even though you are in Jail, you may buy and sell property, buy and sell houses and hotels and collect rents.

2.11. Free Parking

A player landing on this place does not receive any money, property or reward of any kind. This is just a “free” resting place.

2.12. Houses

When you own all the properties in a color-group you may buy houses and erect them on those properties. If you buy one house, you may put it on any one of those properties. The next house you buy must be erected on one of the unimproved properties of this or any other complete color-group you may own. The price you must pay the system for each

house is shown on your property on which you erect the house. The owner still collects double rent from an opponent who lands on the unimproved properties of his/her complete color-group. Following the above rules, you may buy and erect at any time as many houses as your judgement and financial standing will allow. But you must build evenly, i.e., you cannot erect more than one house on any one property of any color-group until you have built one house on every property of that group. You may then begin on the second row of houses, and so on, up to a limit of four houses to a property. For example, you cannot build three houses on one property if you have only one house on another property of that group. As you build evenly, you must also break down evenly if you sell houses back to the system (see SELLING PROPERTY).

2.13. Hotels

When a player has four houses on each property of a complete color-group, he/she may buy a hotel from the Bank and erect it on any property of the color-group. He/she returns the four houses from that property to the system and pays the price for the hotel as shown on the property. Only one hotel may be erected on any one property.

2.14. Selling Property

Unimproved properties, railroads and utilities (but not buildings) may be sold to any player as a private transaction for any amount the owner can get; however, no property can be sold to another player if buildings are standing on any properties of that color-group. Any buildings so located must be sold back to the system before the owner can sell any property of that color-group. Houses and hotels may be sold back to the system at any time for one-half the price paid for them. All houses on one color-group must be sold one by one, evenly, in reverse of the manner in which they were erected. All hotels on one color-group may be sold at once, or they may be sold one house at a time (one hotel equals five houses), evenly, in reverse of the manner in which they were erected.

2.15. Mortgages

Unimproved properties can be mortgaged through the system at any time. Before an improved property can be mortgaged, all the buildings on all the properties of its color-group must be sold back to the system at half price. The mortgage value is printed on each property. No rent can be collected on mortgaged properties or utilities, but rent can be collected on unmortgaged properties in the same group. In order to lift the mortgage, the owner must pay the system the amount of the mortgage plus 10% interest. When all the properties of a color-group are no longer mortgaged, the owner may begin

to buy back houses at full price. The player who mortgages property retains possession of it and no other player may secure it by lifting the mortgage from the Bank.

However, the owner may sell this mortgaged property to another player at any agreed price. If you are the new owner, you may lift the mortgage at once if you wish by paying off the mortgage plus 10% interest to the system. If the mortgage is not lifted at once, you must pay the system 10% interest when you buy the property and if you lift the mortgage later you must pay the system an additional 10% interest as well as the amount of the mortgage.

2.16. Bankruptcy

We are declared bankruptcy if a player has zero money and property. If your debt is to another player, you must turn over to that player all that you have of value and retire from the game. In making this settlement, if you own houses or hotels, you must return these to the system in exchange for money to the extent of one-half the amount paid for them; this cash is given to the creditor. If you have mortgaged property you also turn this property over to your creditor but the new owner must at once pay the Bank the amount of interest on the loan, which is 10% of the value of the property. The new owner who does this may then, at his/her option, pay the principal or hold the property until some later turn, then lift the mortgage. If he/she holds property in this way until a later turn, he/she must pay the interest again upon lifting the mortgage. Players can owe the Bank, instead of another player. In this case, the Bank immediately gets all property so taken, except buildings. A bankrupt player must immediately retire from the game. The last player left in the game wins.

2.17. Miscellaneous

Money can be loaned to a player only by the system and then only by mortgaging property. No player may borrow from or lend money to another player.

3. Requirements

3.1. Functional Requirements

The functional requirements of the game are explained below. After the game opened, the player can do any of the below:

3.1.1. Start Game

Players can start a game by choosing a number of players in which all the players start with the same amount of money, zero properties and buildings. They start in the same starting point cell.

3.1.2. Buy Property

Players shall be able to buy property as much as they can afford.

3.1.3. Roll Die

Players shall be able to roll die and go to another cell.

3.1.4. Pay and Collect Rent

Players shall be pay rent to cell owners and collect rent from guest players.

3.1.5. Buy Houses & Hotels

Players shall buy houses when they get second time to property they owned and when they buy the maximum number of houses for a specific property, they can buy a hotel.

3.1.6. Mortgages

Players shall mortgage their properties to lend money from the bank.

3.1.7. Pause the Game

Players shall be able to pause the game by pressing the pause button(ESC). Then, the pause menu provides 3 different options including “Continue Game”, “Main Menu”, and “Quit Game”.

3.1.8. Quit Game

Players shall be able to exit the game. If the player presses on the Quit Game button in the Main Menu, the game will quit itself. Also, when the player presses the Pause button while the game continues, the player can exit the game by pressing the Quit button.

3.2. Non-functional Requirements

3.2.1. Performance

To ensure a unique game experience the performance of the game is to be kept optimal. The frame rate will not drop below 30 frames per second, on an intel core 2 duo or higher processor machine. Rendering will be done according to the display frame, and the upper limit of crucial cells and objects visible on the screen will be set to 50.

3.2.2. Usability

Menus, titles, and a vivid screenplay will allow us to introduce a new and more engaging form of Monopoly. We have planned to keep the menu very organised and concise in terms of content. The menu will have access to tutorials where all the game rules are going to be displayed. We will try our best to include a video for guidance in the tutorials section. This will entirely help us to keep the users engaged with the game and provided with the user-friendliness, we expect them to play more than once.

3.2.3. Extension In Future

The code is to be organised in a way such that new/improvising features to the game are easily added. These include the following: Improved mechanics - showing a dice being rolled on the board with its splitting and then finally stopping , improved graphics - better animations, variety and functionality of players such as expressing their current state of wealth, enhancing the player avatars with increasing and decreasing wealth.

4. Diagrams

4.1. Use Case Diagram

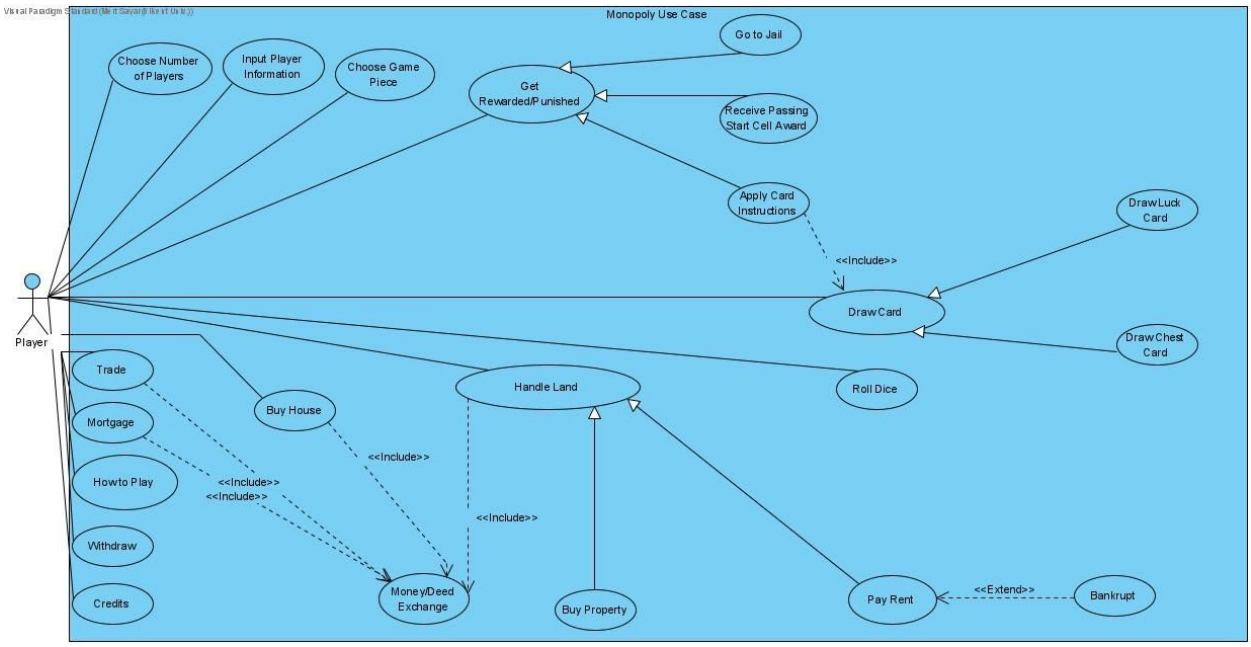

Figure 1: Use Case Diagram for Monopoly

4.1.1 Use Case Descriptions

Use Case 1:

1. Name: How to Play

2. Participating Actor: Player

3. Entry Condition:

- User presses the How to Play button.

4. Exit Condition:

- How to Play screen is shown on the screen.

5. Flow of Events:

- User presses to how to play button.
- How to Play screen is shown on the screen.

Use Case 2:

1. Name: Credits

2. Participating Actor: Player

3. Entry Condition:

- User presses the How to Play button.

4. Exit Condition:

- Credits screen is shown on the screen.

5. Flow of Events:

- User presses to credits button.
- Credits screen is shown on the screen.

Use Case 3:

1. Name: Choose Number of Players

2. Participating Actor: Player

3. Entry Condition:

- Player is on the welcome screen.

4. Exit Condition:

- Number of players is chosen.

5. Flow of Events:

- Monopoly is started.
- Number of players is chosen.

Use Case 4:

1. Name: Roll Dice

2. Participating Actor: Player

3. Entry Condition:

- Monopoly is started.

4. Exit Condition:

- Monopoly started.

5. Flow of Events:

- Player rolls dice.

Use Case 5:

1. Name: Pay Rent

2. Participating Actor: Player

3. Entry Condition:

- Land should be owned by another player.

4. Exit Condition:

- Rent is paid.

5. Flow of Events:

- Player arrives to the land.
- Land is owned by another player.
- Rent of the land is paid.

Use Case 6:

1. Name: Buy Property

2. Participating Actor: Player

3. Entry Condition:

- Land shouldn't be owned by another player.

4. Exit Condition:

- Land is bought.

5. Flow of Events:

- Player arrives to the land.
- Player buys the land.

Use Case 7:

1. Name: Draw Card

2. Participating Actor: Player

3. Entry Condition:

- Player should arrive to luck or chest square.

4. Exit Condition:

- Card is drawn by player.

5. Flow of Events:

- Player arrives to either luck or chest square.
- Player draws the card.

Use Case 8:

1. Name: Go to Jail

2. Participating Actor: Player

3. Entry Condition:

- Player arrives to Go to Jail square.

4. Exit Condition:

- Player goes to the Jail.

5. Flow of Events:

- Player arrives to Go to Jail square
- Player goes to the jail.

Use Case 9:

1. Name: Receive Passing Start Cell Award

2. Participating Actor: Player

3. Entry Condition:

- Player should pass the starting point.

4. Exit Condition:

- Player receives the award.

5. Flow of Events:

- Player passes the starting point.
- Player receives the award.

Use Case 10:

1. Name: Money/Deed Exchange

2. Participating Actor: Player

3. Entry Condition:

- Player should either Pay Rent, Buy Property, Buy House, Trade or Mortgage property

4. Exit Condition:

- Money/Deed Exchange is completed.

5. Flow of Events:

- Player either Pays Rent, Buy Property, Buy House, Trade or Mortgage property.
- Player completes the exchange.

Use Case 11:

1. Name: Buy House

2. Participating Actor: Player

3. Entry Condition:

- Player should have completed a monopoly(Player should own all properties from a color group
- Player has enough money to buy a house.
- Player presses Buy House button.

4. Exit Condition:

- House(s) bought.

5. Flow of Events:

- Player decides to buy a house.
- Buy house button is enabled if player has enough money and a monopoly.
- Player chooses number of house he/she wants to buy.
- House(s) bought.

Use Case 12:

1. Name: Trade

2. Participating Actor: Player

3. Entry Condition:

- Player presses the Trade button.
- Player offers a deal to the Player.

4. Exit Condition:

- Offer is rejected by the other player or the trade is completed.

5. Flow of Events:

- Player presses the trade button.
- Player chooses the player who he/she wants to offer a trade.
- Player either offers money or a deed for a property owned by the other player.
- Offer is either rejected by the other player or the trade is completed.

Use Case 13:

1. Name: Mortgage

2. Participating Actor: Player

3. Entry Condition:

- Player should own a property.
- Player presses withdraw button.

4. Exit Condition:

- Deed chosen by the player gets mortgaged.

5. Flow of Events:

- Player presses the mortgage button.
- Player chooses the deed he/she wants to mortgage.
- Deed chosen by the player gets mortgaged.

Use Case 14:

1. Name: Bankrupt

2. Participating Actor: Player

3. Entry Condition:

- Player's dept is more than total values of player's money and deeds.

4. Exit Condition:

- Player is out of the game.

5. Flow of Events:

- Whenever player pays money, player is checked by game manager if he/she bankrupt.
- If Player's dept is more than total values of player's money and deeds, player is bankrupt.
- All money and the deeds owned by the player are transferred to the player which bankrupt player has dept.
- Player is out of the game.

Use Case 15:

1. Name: Withdraw

2. Participating Actor: Player

3. Entry Condition:

- Withdraw button is pressed.

4. Exit Condition:

- Player is out of the game.

5. Flow of Events:

- Player presses withdraw button
- All money and the deeds owned by the player are transferred to the bank.
- Player is out of the game.

4.2. State Diagrams

State diagrams represent the states according to the activities made for specific components like property cells or users. Our state diagrams represent one life-cycle of turn of a player and the states of a property cell along the monopoly game below.

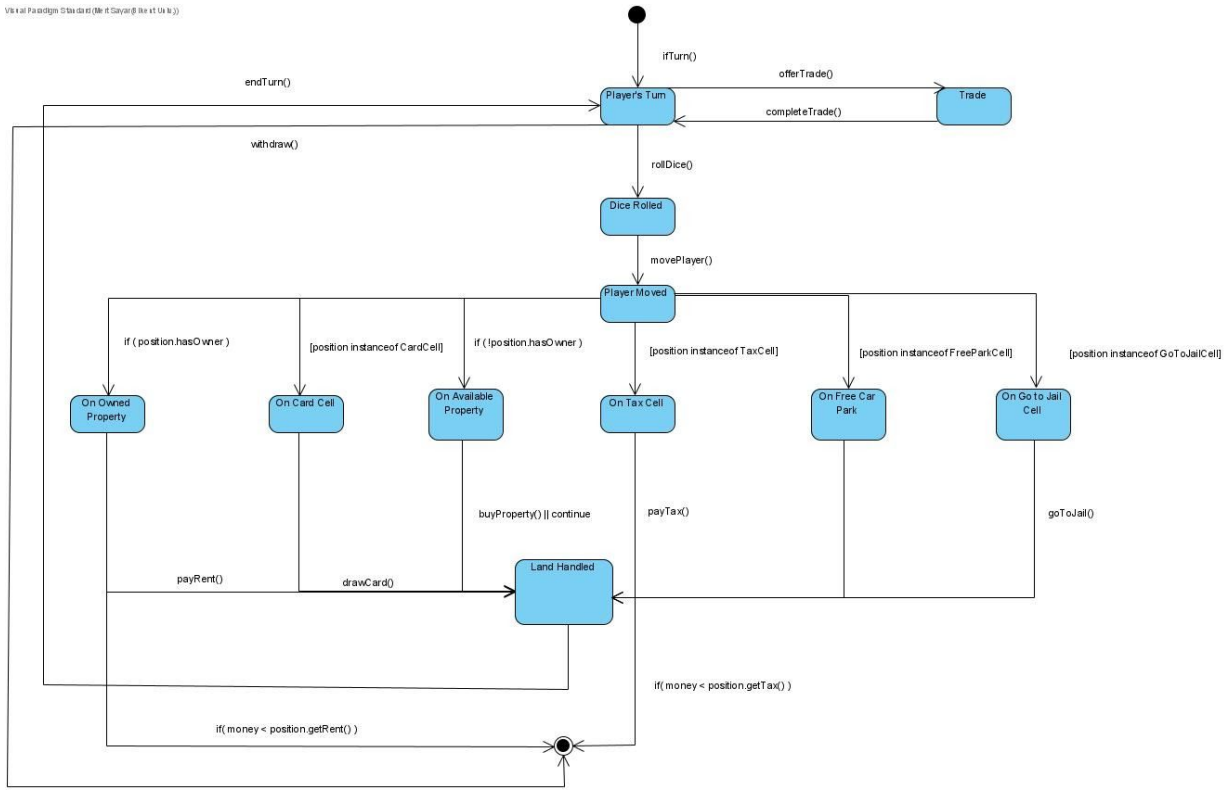

Figure 2: State Diagram for one turn for each player

Explanation:

This diagram provides the possible states for a player to land in his/her move. Before rolling the dice, the player is able to offer a trade to another player. After rolling the dice, player moves according to the dice and handles the land accordingly. Land handling consists of paying rent, buying the property, pay tax fee, going to jell, drawing a card or doing nothing at the related land. A player will reach the final state of the diagram if he/she is bankrupt or if the player withdraws the game. If a player reaches the final state, there will be no other turn for that player since the game will be end for him/her.

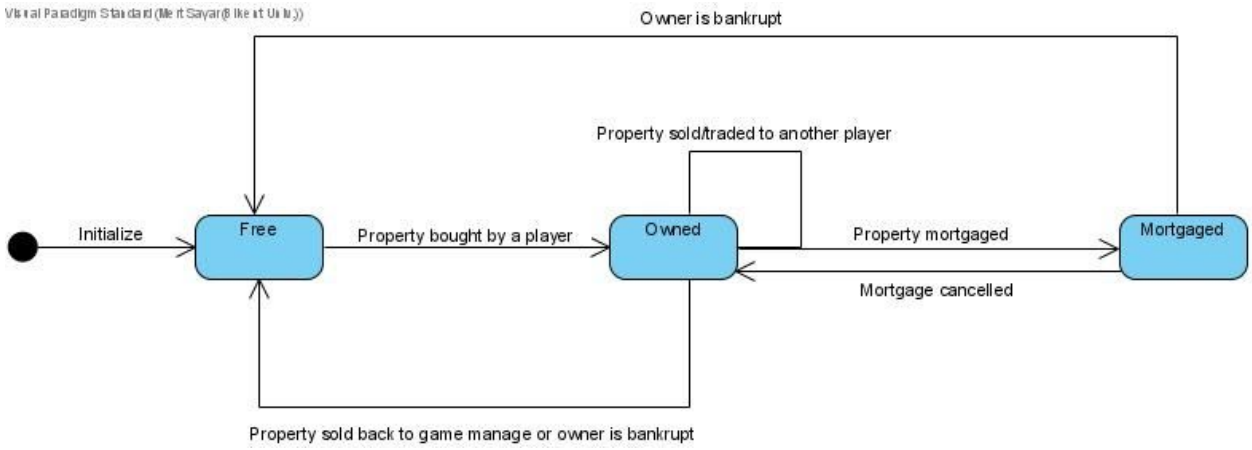

Figure 3: State Diagram for property cells.

Explanation:

This state diagram provides the possible states for a property cell along the game. A property cell will be initialized as free which means there is no owner for that cell. If a player buys the related cell, cells state will be “Owned” which means no any other player can buy this cell at this state. State of the cell will not change if the cell is traded or sold to any other player because the cell will still have an owner. Player can decide to sell its cell back to the bank, and the cell will be “Free” again in this situation. A player can mortgage his/her property and the state will be “Mortgaged” in that case. While in mortgage state, player will not be able to sell/trade its property. Cell will shift to owned again if the mortgage is cancelled. No matter which state is current, the state will be free if the owner of the cell withdraws from the game.

4.3. Sequence Diagrams

In the Sequence Diagrams we depict how the classes are organized and utilised interchangeably. These diagrams show how classes depend on one another for a goal to be achieved, hence making Object Oriented Programming a special feature to use.

4.3.1. Sequence Diagram for going to Jail

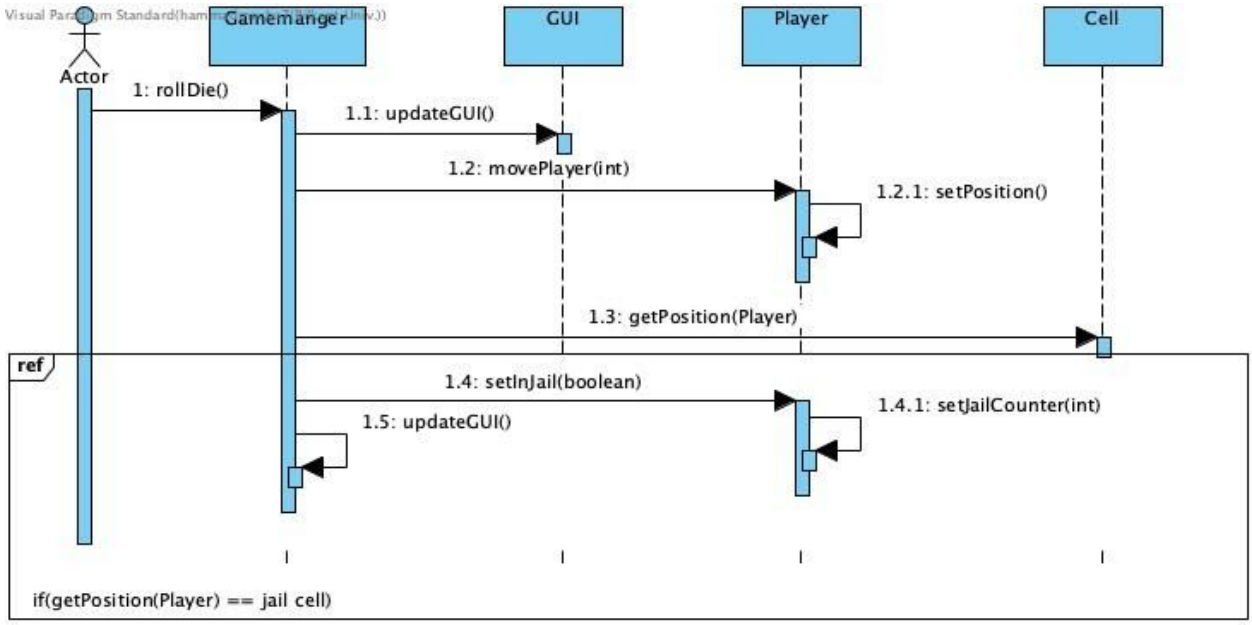

Figure 4: Sequence Diagram for going to jail

Scenario:

In this sequence diagram we depict how the classes feature into taking a player to a jail, in essence making the jail feature work, we assume that the die rolled produces the number that will make the player land on the GO TO JAIL cell, and hence will be jailed. We request the reader that our most of the functions/methods are still raw in nature and require modifications during implementation. This is a general idea of the sequence of events for this particular scenario.

Explanation:

Initially, after clicking on the rolling button for the dice, the method rollDie() is called in GameManager which results in the generation of a random number for the dice. The die's number is stored in an integer which is then used to move the player by calling the movePlayer(int) function on the player object. Then the setPosition() is called on the player object as it is a function defined in the player class; this gives the player its new position in terms of index.

Then the getPosition(Player) function is called on the cell instance of its class. This assesses the type of cell the player is on, in our case we have assumed it is a jail cell. A condition is developed, since it is a jail cell, if both indices are equal, we go into a conditional statement which calls the setInJail(boolean) function on the player object, and then the player calls onto itself the setInJailCounter(int) function which then adds the

times the player has been to jail; lastly updateGUI() is called which then updates the GUI with relevant information to the player(s). The main idea is yet to be implemented and tested in practice but this would change the variable in the player's object of inJail to a boolean "true." This is what we need to do to restrict the player into jail.

4.3.2. Sequence Diagram for purchasing property

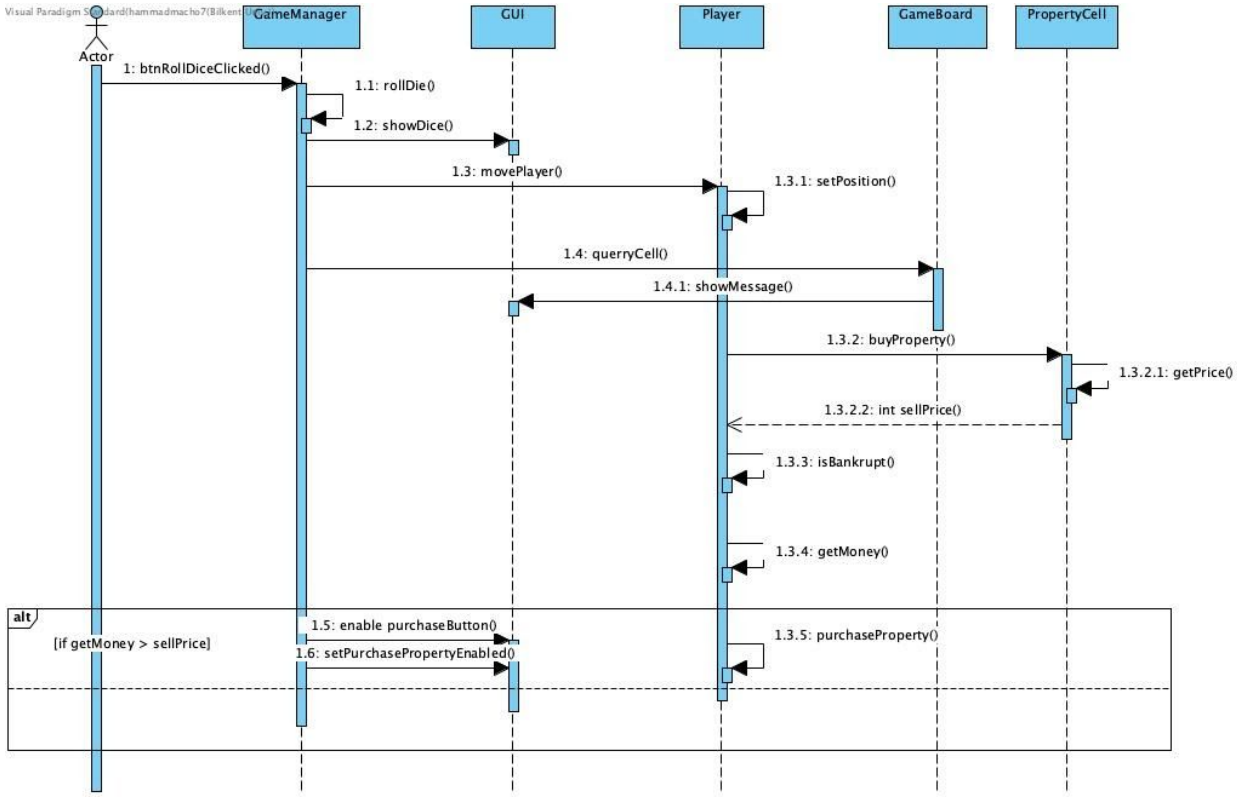

Figure 5: Sequence Diagram for purchasing property

Scenario:

In this sequence diagram we depict how the classes feature into enabling a player to purchase property, in essence making the purchase feature work, we assume that the die rolled produces the number that will make the player land on a property cell, and hence will be able to purchase. We request the reader that our most of the functions/methods are still raw in nature and require modifications during implementation. This is a general idea of the sequence of events for this particular scenario.

Explanation:

Initially, after clicking on the rolling button for the dice, the method btnRollDiceClicked() is called in GameManager which then calls rollDie() which results in the generation of a random number for the dice. The die's number is stored in an integer which is then used to move the player by calling the movePlayer() function on the player object. Then the setPosition() is called on the player object as it is a function defined in the player class; this gives the player its new position in terms of index.

Then the querryCell() function is called on the GameBoard instance of its class. This assesses the type of cell the player is on, in our case we have assumed it is a property cell. After knowing it is a property cell, the software would automatically assess the player to BUY this by calling the buyProperty() on the cell instance inside the player class. This would result in the returning of the sellingPrice of the property and the player would call its isBankrupt() and getMoney() functions to know if they can purchase that property. Again, a condition proceeds and if getMoney() > sellingPrice, then purchase button is enabled and after clicking on that, player is able purchase the property by calling the purchaseProperty() on player and then setPurchaseEnabled() button appears, depicting that the property has been purchased. The player can then further go and set rent, etc etc.

4.3.3 Sequence Diagram for getting out of jail

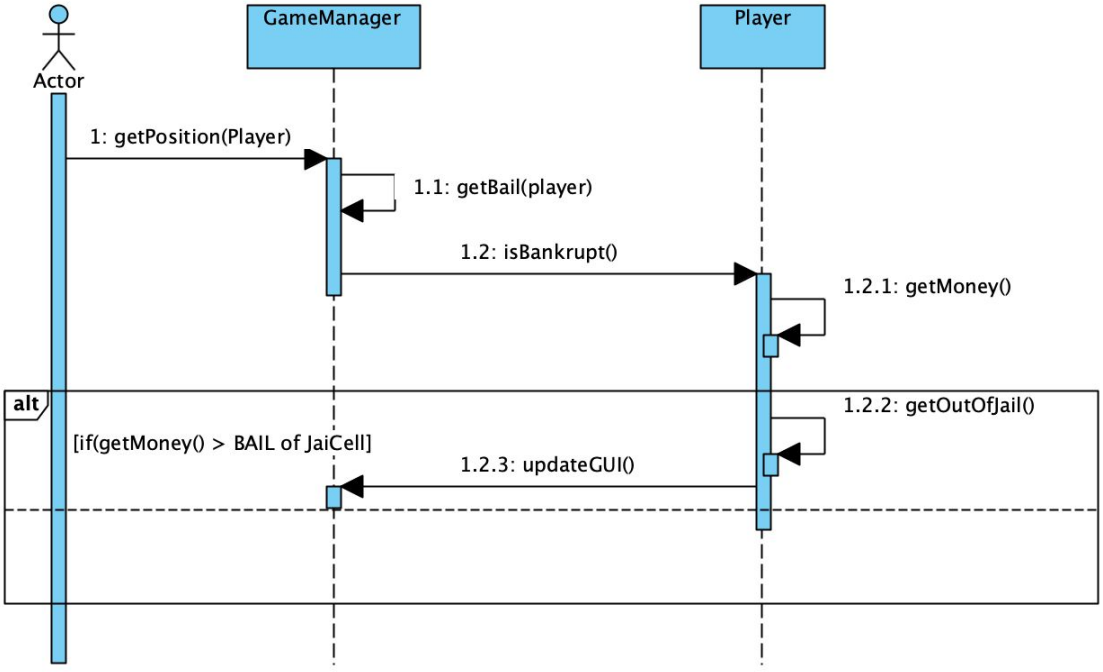

Figure 6: Sequence Diagram for Getting out of jail

Scenario:

In this sequence diagram we have depicted the sequence of class interactions to get a player out of the jail in the most simple of ways and that is by paying bail. Here, the reader must be informed that there are other ways of getting a player out of jail that is by drawing cards and one can get the card to get out of jail as well. However, to keep it simple for basic implementation and notion we have concluded to introduce the bailing way of getting out of jail.

Explanation:

In this sequence diagram, we assume that the player is already in jail, that is the player attribute inJail is true. We aim to make this false and get the player out of jail. Initially, the GameManger gets the player position by calling the getPosition(Player) function which returns the type of cell the player is at/restricted to at that particular moment. Then, GameManger calls the getBail(Player) function with Player in jail as the parameter; this function would only further commence if the player is actually in jail and thus its attribute is checked. Furthermore, inBankrupt() function is called on the Player in Jail to see if the player is Bankrupt or not; assuming the player is not, we proceed with checking the amount of money the player has by calling the getMoney() function on the player. Then, a conditional statement arises where the bail price and amount of money player has to be equal or greater; if it is so, then getOuOfJail() is called on the player and the attribute is updated accordingly.

4.4. Activity Diagrams

4.4.1. Main Menu Activity

When the game starts, the main menu screen will show up. This system can take inputs to display How to Play screen, the game screen and the quit button.

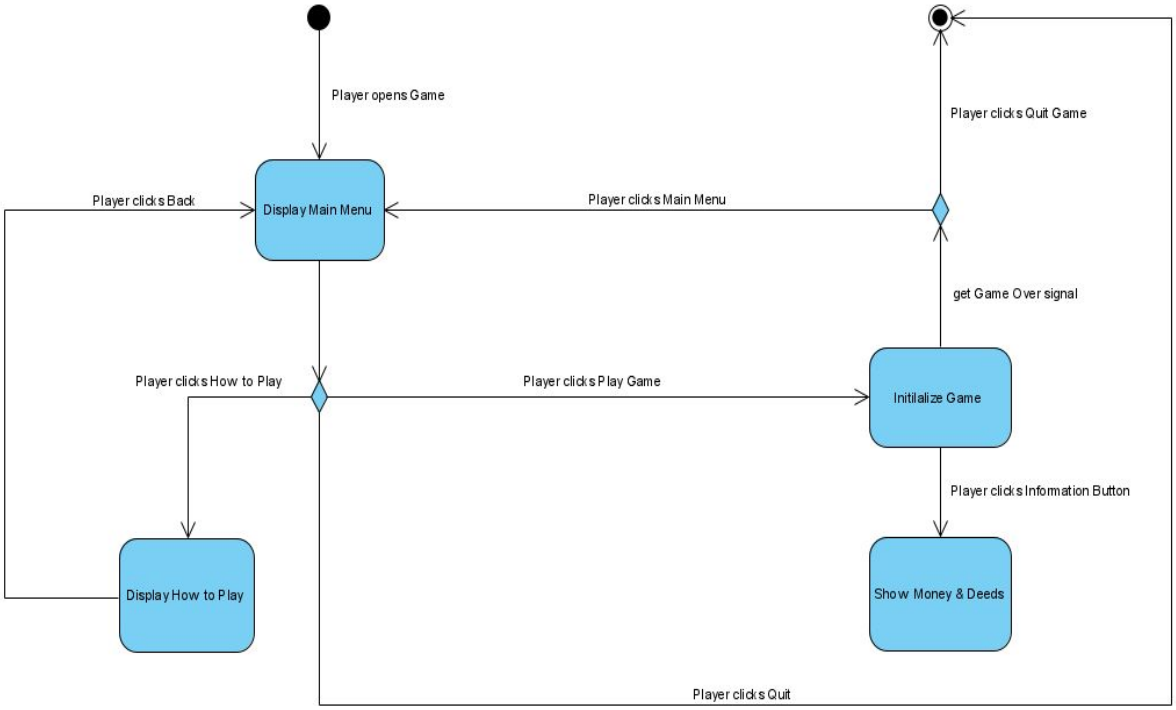

Figure 7: Activity Diagram for main menu

4.4.2. Turn Activity

This figure shows how players can move. This activity mostly depends on the player's choices.

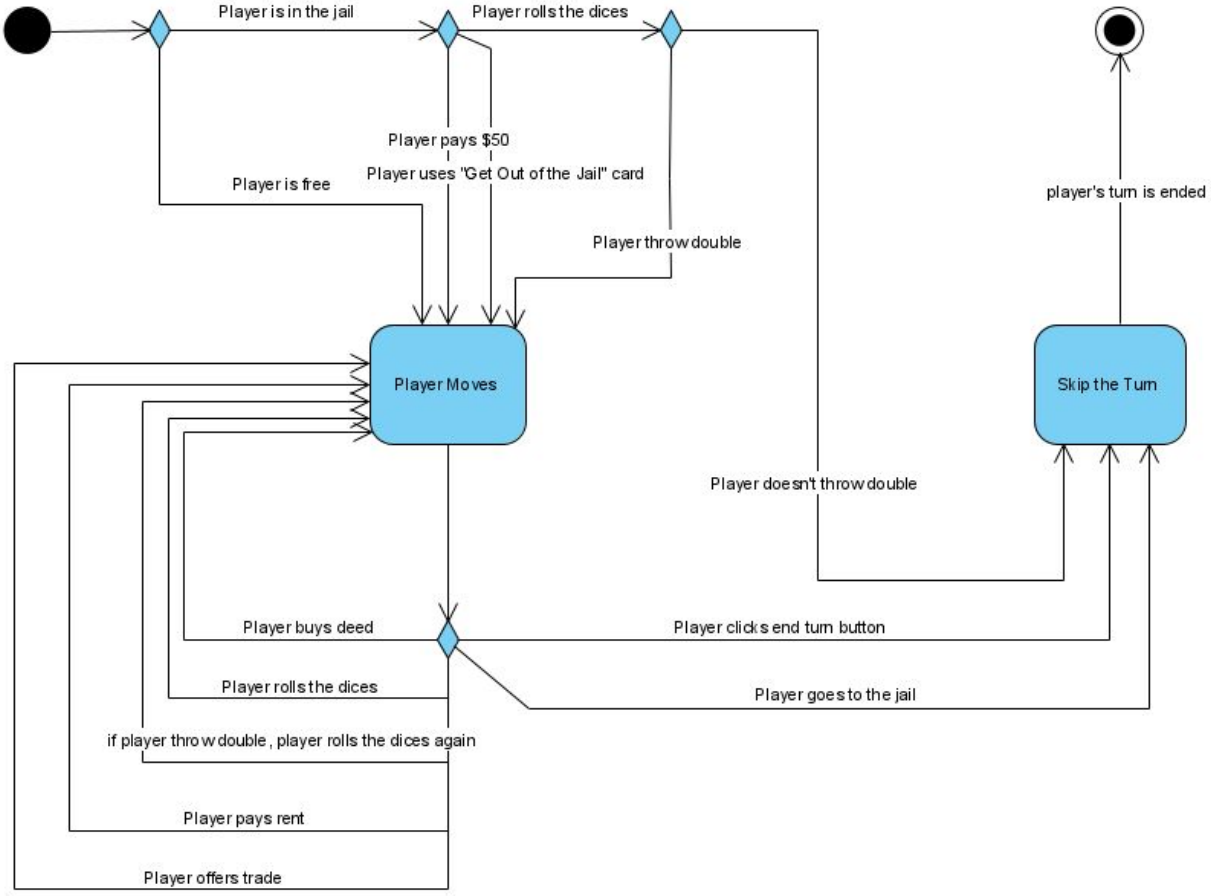

Figure 8: Activity Diagram for turns

4.4.3. Final Total Money Calculation Activity

This figure shows how the system calculates the final money of players. When the game is over, the system starts to calculate total money for each player by checking their deeds, their buildings and their money.

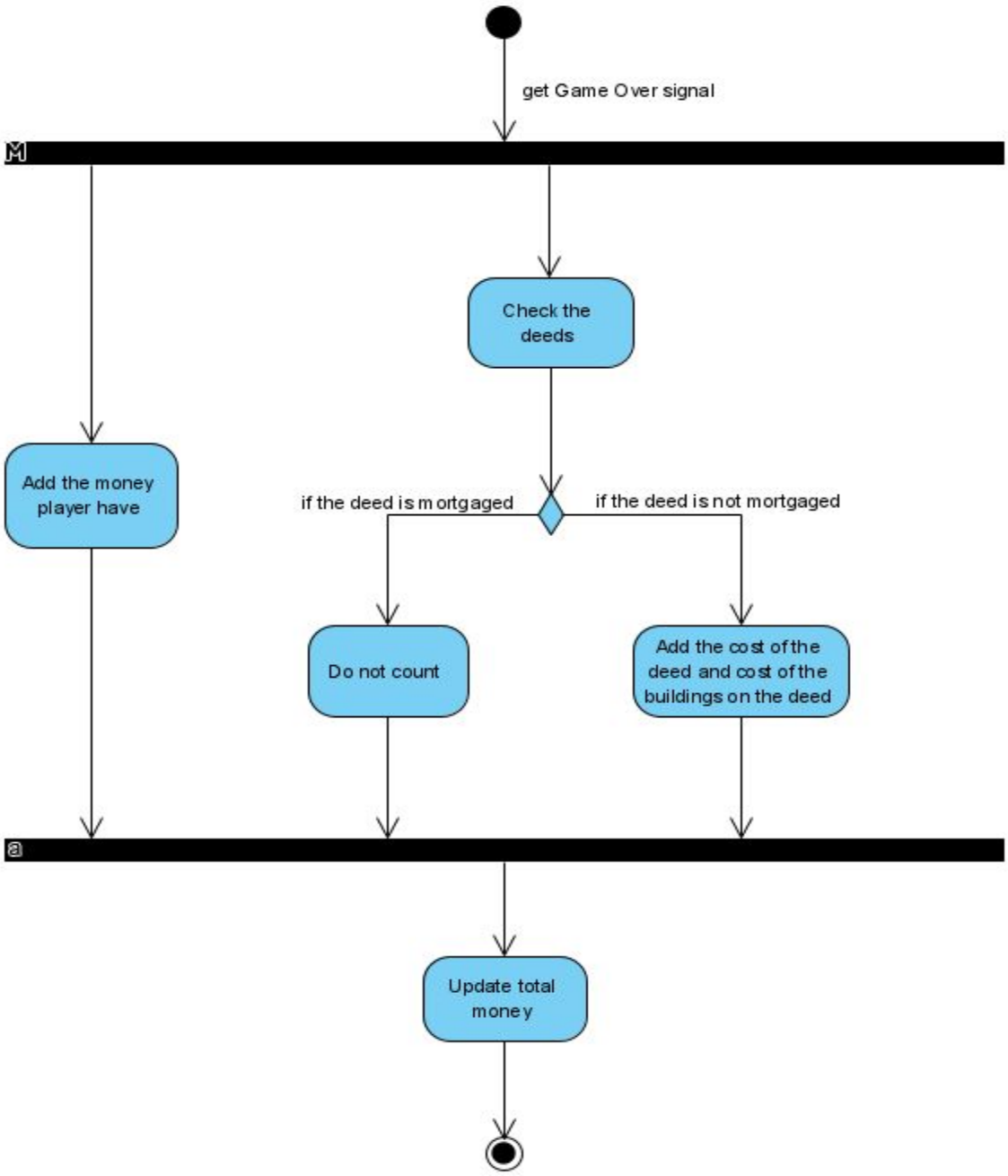

Figure 9: Activity Diagram for money calculation

4.5. Class Model

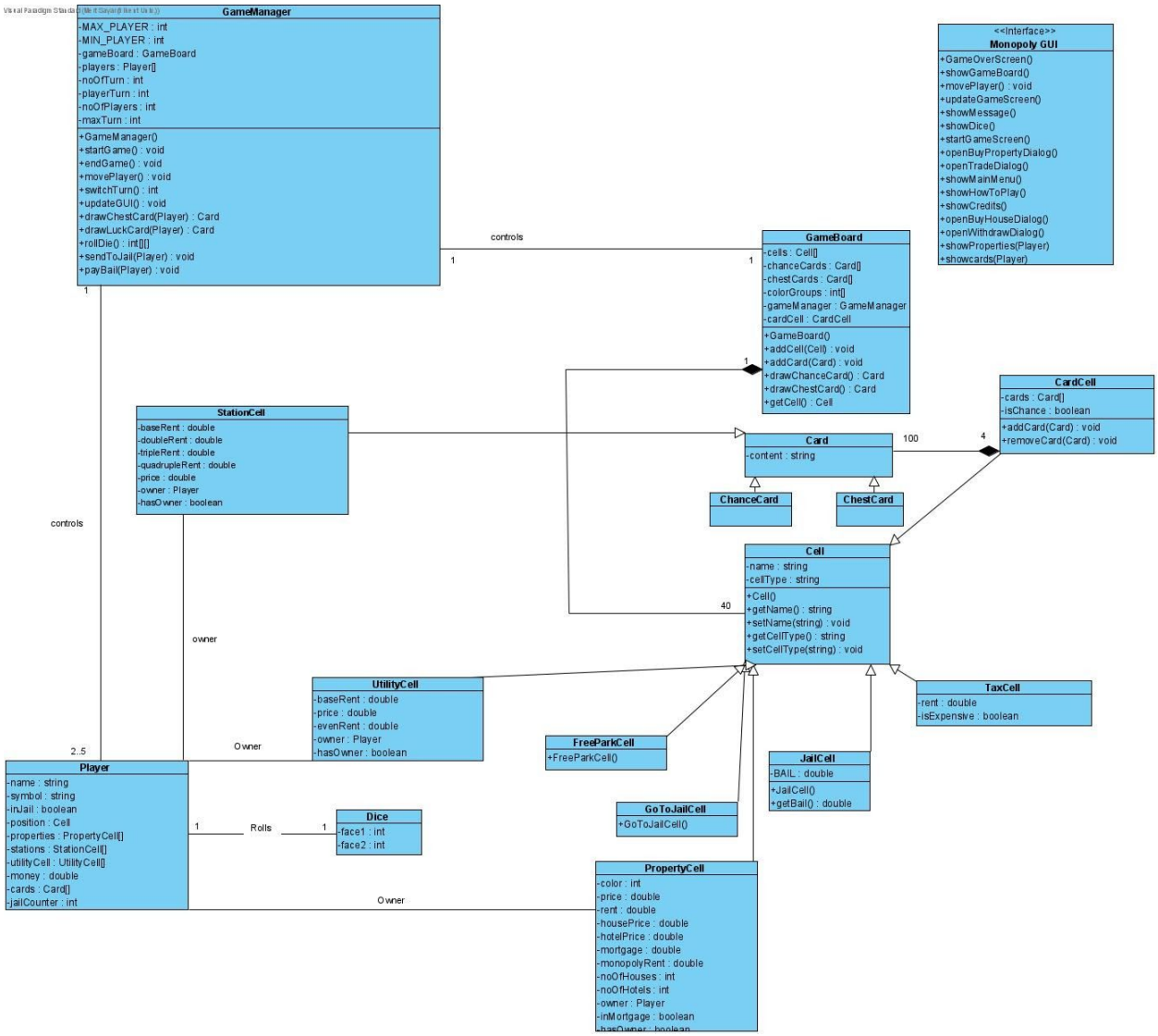

Figure 10: Class Diagram of Monopoly

4.6. User Interface

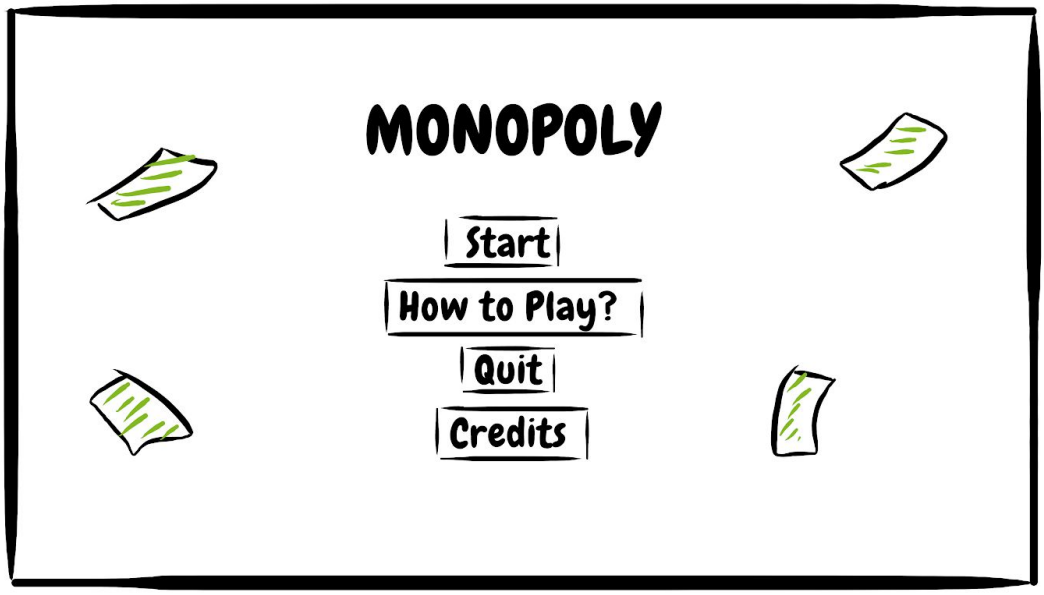

Figure 8: Mockup for the main menu

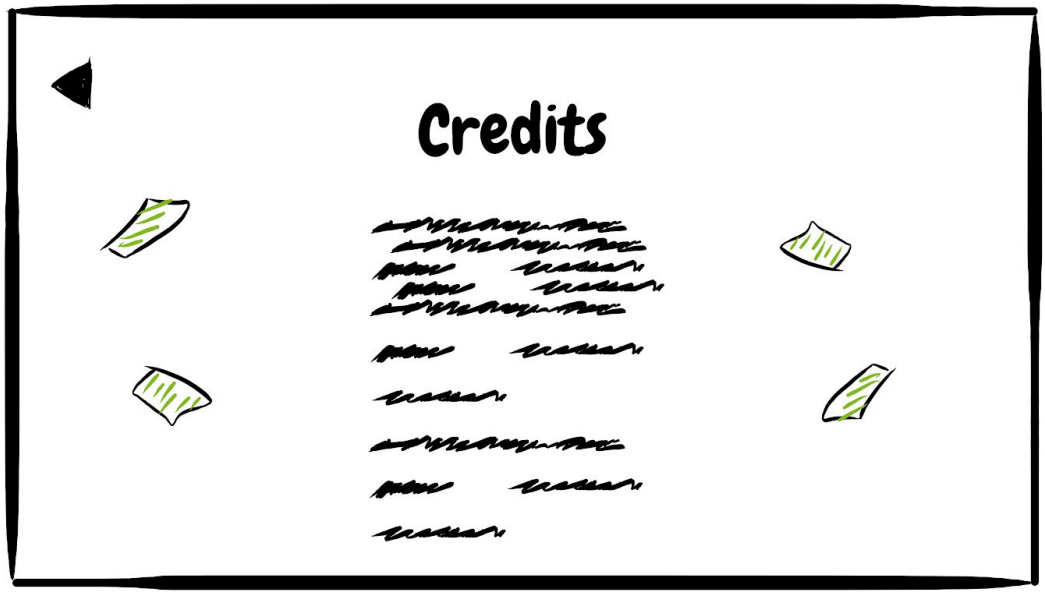

Figure 11: Mockup for the credits

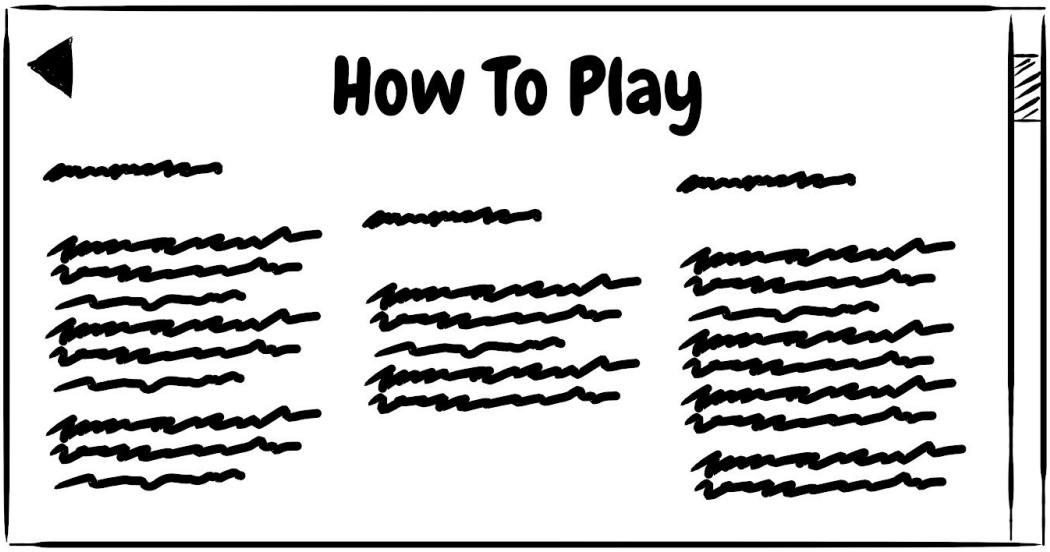

Figure 12: Mockup for the “How to Play” screen

Figure 13: Mockup for the screen that wants player input from the players

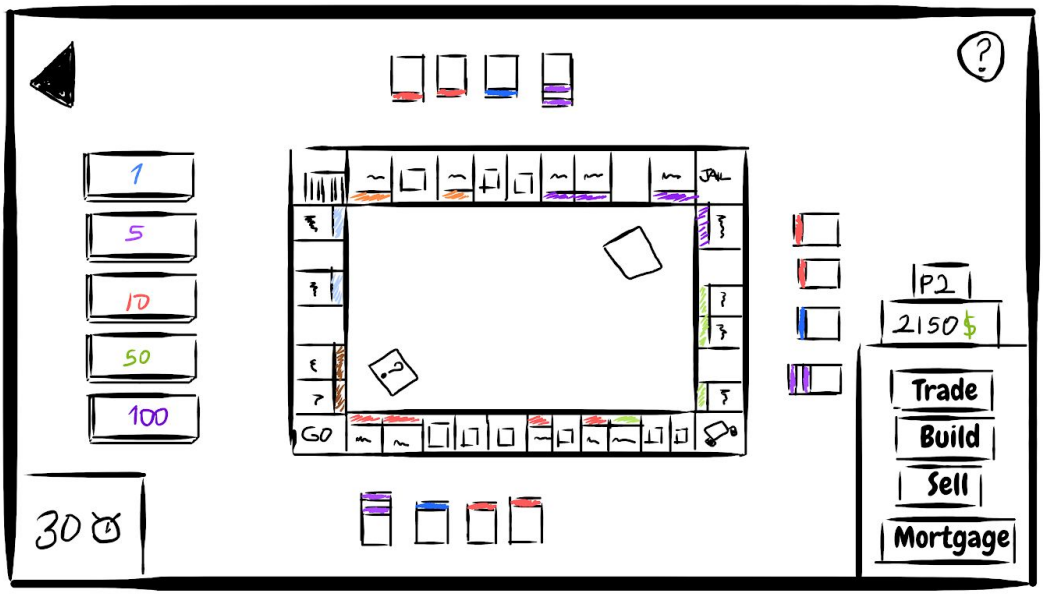

Figure 14: Mockup for the game screen

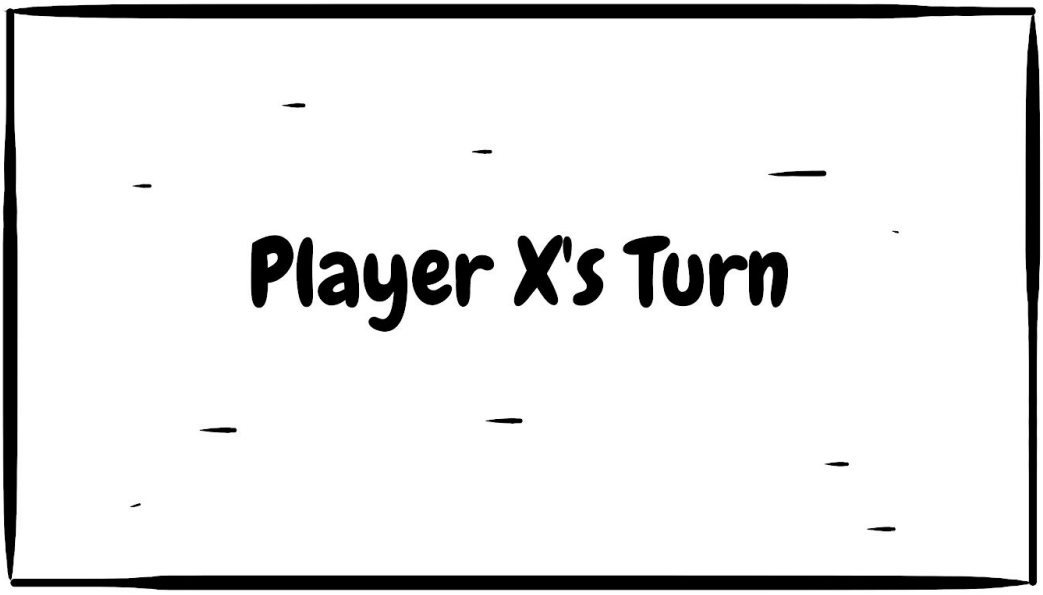

Figure 15: Mockup for the transition scene between player turns