
Bilkent University
Department of Computer Engineering

*CS 319 Term Project: Monopoly*

*Section 01*

*Group 1A*

# ANALYSIS REPORT

Atakan Dönmez
Elif Kurtay
Musa Ege Ünalan
Mustafa Göktan Güdükbay
Yusuf Ardahan Doğru

Instructor: Eray Tüzün
Teaching Assistant(s): Barış Ardıç, Emre Sülün, Elgun Jabrayilzade

# Table of Contents

# Table of Figures

# 1 Introduction

We are Group 1A and we will be recreating the household tabletop game Monopoly to be playable on the PC platform but with our brand-new twists. In this reimagined version of Monopoly, you will be able to play with up to 4 people or against the computer. The game will closely follow the steps of the original one however the game's main issues regarding how long it takes to finish and its monotony will be addressed with new additions such as the thief who tries to catch players and steal their money, special perks all unique to the pieces, the ability to set a turn limit or to save your game and continue later on by loading and more.

The game will be implemented in Java and will be designed and created using the principles of object-oriented programming as taught in CS 319 closely to the extent of our capabilities by utilizing techniques and principles taught in the lectures.

# 2 Overview

Digital Monopoly is a digital recreation of the classic board game Monopoly. You will be able to play with your friends, against bots, or both at the same time on your Windows operated computers. Each player takes turns rolling dice to advance on the board and perform actions like buying property, building houses, paying rent, trading, mortgaging, and more. The goal of a player is to become the richest person in the game while forcing her/his opponents to go bankrupt through managing your assets and mischievous trade deals to build your monopoly. The game will be decided when all players go bankrupt or the turn limit is reached, and the wealthiest player is crowned as the winner.

## 2.1 Start Game

Users will be able to create new games or load their saves of previous games. When creating a new game user will add players that are to be controlled by the user. If the user-controller player count is less than 4 the remaining spots will be covered by players created and controlled by the computer.

## 2.2 Gameplay

To start the game, each player should have $1500. Each player will roll the dice to determine his or her turn. The player that has the highest total after rolling the dice will have the first turn, the player that has the highest total after rolling the dice will have the second turn, and so on. To start the game, the player will choose a token and throw the dice. The player will move the token the number of spaces s/he rolled. After the player completes his or her turn the next player will choose a token and roll the dice. The game will continue like this: each player will roll the dice when it is their turn and perform an action. If a player throws doubles, they move their token as usual. The same player throws the dice again and moves their token as usual. If they throw doubles three times repeatedly, they will go to jail. The place they land on will determine their course of action.

Each time when a player's token lands on or passes over the GO spot the banker will pay them $200. Additionally, if a player passing GO on the throw of dice lands 2 spaces beyond it on Community Chest, or 7 spaces beyond it on Chance, and draws the "Advance to GO" card, s/he collects an extra $200.

When a player stands on a property that is owned by another player, s/he will pay rent to the owner. If the property is mortgaged, they will not pay any rent. If the player that holds that property owns all the properties of the same color, then a visitor will pay double rent.

When a player stands on a chance and community chest s/he will take the top card from the deck and follow the instructions on the card. The card will be returned to the bottom of the deck. The "Get Out of Jail Free" card is held to be used later or the player can sell it to another player at a price they agree.

When a player stands on the "Income Tax" s/he pays the amount specified on the card. The player can also choose to "Trade", "Build", "Mortgage" and "Redeem" at any time during their turn.

The player ends their turn once they are done with it and after all four players play their turn the turn count increases.

## 2.3 Player Pieces

Each player will choose a piece at the beginning of the game. Each piece will have positive and negative perks.

*Thimble:* + Tax multiplier of 0.8
- Rent collect multiplier of 0.8

*Wheelbarrow:* + Building cost is half
- Salary change of -M50 (Total salary is M150)

*Boot:* + Property cost multiplier 0.8
- Salary change of -M100 (Total salary is M100)

*Horse:* + Rent collect multiplier of 1.3
- Property cost multiplier of 1.1

*Racecar:* + Bonus salary is M200 (Total salary is M400)
- Jail Time of 4 rounds

*Iron:* + Building cost multiplier of 0.8
- Rent pay multiplier of 1.2

*Top hat:* + Jail Time of 2 rounds
- Tax Multiplier of 1.2

*Battleship:* + Rent pay multiplier of 0.7
- Building cost multiplier is 1.5

## 2.4 Map

The map consists of mostly property tiles and some special tiles. In addition to the classic Monopoly map, there will be different maps available. Also, the players will be given a “Custom Map” choice where they can design the map destinations.

### 2.4.1 Property

The property tiles feature a tier system in which the higher a tier the property is its cost of buying and building increases but also the rent imposed on other players increases too. This is a general trend as tiers (along with property cost, rent, and so on) increase from the “GO” position to the other end all the way back to the “GO” position again in the clockwise direction. The tiers from the lowest to the highest in colors are Brown, Light Blue, Pink, Orange, Red, Yellow, Green, and Dark Blue. When a player wants to buy a property that is not owned s/he should pay the written price. After buying they will receive the Title Deed Card. If they do not wish to buy the property, the bank will sell it at auction to the highest bidder.

### 2.4.2 Railroads Property

The opposing players that land on these tiles are rented according to the amount of railroad property cards you own. While these tiles may not be a hot commodity on their own, as a player acquires more of them, they become more of a danger and the other players may suddenly need to prevent that player from acquiring more. If the player that lands on this spot is the owner of the property and owns another railroad station, they can choose to travel to that station.

### 2.4.3 Tax

The players that land on these spots must pay the specified tax amount to the bank.

### 2.4.4 Chance and Community Card

The players that land on these spots draw a card from the deck related to the tile. These cards have commands that players are forced to follow. However, the player can choose to use the card's effect on the current round or keep it to use it next round. These cards can have effects both positive and negative like "Go to X spot", "It's your birthday everyone pays you X amount", "Go to jail" and more.

Inside "Chance" cards, there will be a "Thief" card. This card will introduce a new character/player in the game. The thief will try to catch a player by being in the same spot as another player. When the Thief catches a player, that player will get their money stolen. After stealing the Thief disappears from the game. The player who picks the Thief card can choose to use it by paying a certain amount or not to play it.

### 2.4.5 Electric Company and Water Works

The opposing players that land on this spot roll dice to determine how much they need to pay for rent. If the renter owns only one of these properties, the opposing player pays 4 times the result of the dice roll if the renter owns both of these properties the opposing player pays 10 times the result of the dice roll.

### 2.4.6 Jail

The players can get stuck in jail if they draw a card that sends them there, land on the "Go to Jail" spot, or throw a double three times in a single turn. When a player goes to jail, they cannot collect their $200 GO bonus. There are three ways to get out of jail:

1. S/he throws doubles on any of his or her next three turns.

2. S/he uses the "Get Out of Jail Free" card (they can buy it from another player if they do not have it).

3. S/he pays a fine of $50 before s/he rolls the dice on either of his or her next two turns.

If a player does not get out of jail by their third turn, they must pay the $50 fine and get out. Actions like collecting rents, buying, and selling properties are not affected when a player is in jail.

### 2.4.7 Wheel of Fortune

If a player lands on a Wheel Of Fortune Space, they will spin a wheel and gain a random property, money, a house on one of their properties if possible, a "Get Out Of Jail Free" card or they may lose money or one of their buildings on one of their properties.

### 2.4.8 Go

All players start on this spot at the beginning of the game. If the player lands on or passes this spot they get paid an M200 salary every time they do. The player can collect the salary as many times as they pass in a single round. If they are sent to jail or used a card that specifically mentions that the player should not get paid then the player cannot collect the salary.

## 2.5 Build

When a player has all the properties of the same color, they may buy houses from the bank and put them in those properties. Players cannot put more than one house on any property until they have built a house on every property of the same color. This rule continues for the second, third, and fourth row. Players should first complete that row for a color to start another row. No more than four houses can be bought for each property. When a player has four houses on a property, s/he can return the houses s/he has and buy a hotel to put it on any property of the color-group. A property can only have one hotel. These actions can only be done while the player is standing on a property of the color of the property that the player wants to build on. For example, if the player lands on a red property it can build on all red properties if the player owns them all, but no other property tier can be built upon.

## 2.6 Trade

A player can sell any unimproved properties (except buildings) to any player at any price. If buildings are standing on any property of the same color, they must first sell the buildings to the bank to sell any property of that color. A player can get half the price they paid for the houses and hotels when they sell them to the bank. Houses on a property cannot be sold at once, they should be sold one by one, the reverse of the manner they were bought. All hotels on one color-group may be sold at once, or they may be sold one house at a time (one hotel = five houses), evenly, in reverse of how they were erected. The player can make offers to others and others can accept or decline this offer.

## 2.7 Mortgage

Unimproved properties can be mortgaged through the bank at any time. The mortgage value is printed on each "Title Deed" card. To lift the mortgage, the owner must pay the bank 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 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 bank. If the mortgage is not lifted at once, you must pay the bank 10% interest when you buy the property and if you lift the mortgage later you must pay the bank an additional 10% interest as well as the amount of the mortgage.

## 2.8 Bankruptcy

You are declared bankrupt if you owe more than you can pay either to another player or to the bank. If your debt is to another player, you must turn over to that player all that you have of value and leave the game. If you own houses or hotels, these will be returned to the bank and half their value will be paid to the creditor by the bank. If you have mortgaged property you also turn this property over to your creditor, but the new owner must at once pay the bank 10% of that property. The new owner may then pay the principal or hold the property until s/he lifts the mortgage. If s/he holds the property in this way until a later turn, s/he must pay the interest again upon lifting the mortgage. You should owe the bank, instead of another player, more than you can pay (because of taxes or penalties) even by selling off buildings and mortgaging property, you must turn over all assets to the bank. In this case, the bank immediately sells by auction all property so taken, except buildings. A bankrupt player must leave the game.

## 2.9 Game End

The game will end when all but one player is bankrupt or the specified turn count is reached. The game can be restarted or saved to play again later. If the exit button is pressed the game can also be ended manually. A scoreboard showing all the players' net worth and their ranking in the game will be shown.

## 2.10 Settings

Music and game sound can be adjusted through settings.

# 3 Functional Requirements

## 3.1 Start Screen

This will be the initial page the user is presented with. From this screen, the user will have the options to go to “New Game”, “Load Game”, “Settings”, “Credits” pages or quit by clicking the respectively named buttons.

## 3.2 New Game

The user can choose which map to play on or add their own custom map.

After pressing the start game button, the users are met with a screen where they can see each avatar’s perks by hovering their mouse over it and choose their avatar by clicking on it. Chosen avatars will be disabled.

## 3.3 Load Game

The user can access “Load Game” from the “Start Screen” where they are presented a table of the game number, date, length, and the name of the players of their previously saved games. By clicking the corresponding game number, their saved game is loaded to play.

## 3.4 Gameplay Screen

- *Start of the turn:* The user is presented with a pop-up indicating it is her/his turn and the user can roll dice. The result of the roll is presented, and the piece is moved. The user is prompted to buy the tile and if they do not the tile is presented for auction. In the auction screen, the other users can offer their bids and the highest wins.

- *Chance/Community Chest:* The player can choose if they want to open the card or save it for the next round. When the card is opened the player will be displayed the contents of the card. If the card is a “Thief” card the player can choose to pay to play it or discard it.

- *Trade:* When the user presses the trade option a pop-up will appear and the user can choose what properties and how much money they want to offer and select which properties and how much money they demand from the opposing player. After clicking “Offer” the opposing player can either accept or decline the offer.

- *Build:* When the user is on a tile of which they own all the properties of the same color they can click “Build” and choose on which property to further build/start building.

- *Mortgage and Redeem:* The user can access "Mortgage" from “Gameplay Screen” and choose to mortgage one of their properties. Later, they can access “Redeem” from “Gameplay Screen” and lift the mortgage.

- *View Assets:* The player can view the assets of any player by clicking “Assets” at any moment.

- *Other Menu Choices:* The player can choose at any moment to go to the “Settings”, restart the game, save it, or exit.

### 3.5 End Game

When the game ends, a pop-up screen will display the names, net worth, and ranking according to their net worth on the screen. The user can restart or quit the game from this screen.

### 3.6 Settings

The user can access “Settings” from the “Start Screen” where they can adjust Music and Game Sound.

### 3.7 Credits

The user can access “Credits” from the “Start Screen” where they are presented with the names of the developers.

# 4 Non-functional Requirements

## 4.1 Readability and Extendibility

The source code should be organized and written in a manner that third parties looking to change, extend, or contribute to the project should be able to understand how the systems work and how they are implemented. The concepts that are learned in CS319 and CS102 regarding object-oriented design and programming etiquette should be closely followed to make bug fixes and new implementations easy to manage.

## 4.2 Intuitive and User-Friendly Interface

The graphical user interface and the way the game operates should be designed in a manner that allows the user to easily operate throughout the game and navigate through information easily. All operations should be designed to be intuitive to create a smooth experience for the end-user.

## 4.3 Performance

The algorithms and graphical operations should be coded and designed to be fast responsive. The code should be optimized to require the lowest system requirements it can, to be accessible to as many possible users as possible. The gameplay should feel fluid and responsive therefore the highest frame rate per second and the lowest response time as possible on buttons and such should be achieved.

## 4.4 Reliability

The software should be implemented in a way that would consistently work as intended and without any safety concerns. The software features saving and loading operations and such operations should handle data in a manner that compromises neither the user nor the software to possible threats. The software should also be thoroughly tested to detect and fix any possible bugs and errors that would cause it to act in unexpected ways. These could also be helped by a well-produced design of error-handling systems.

# 5 Pseudo Requirements

## 5.1 Language

The implementation language will be Java, and specifically, Version 8. This implementation will be done on a common IDE for all coders of the project which is IntelliJ IDEA. Java's Swing libraries will be utilized to create the Graphical User Interface.

## 5.2 Version Control System

The implementation will be monitored by a Git Version Control System. The implementations, changes, and reports will be pushed to our GitHub repository by using the Git plugin of the aforementioned IntelliJ IDEA IDE.

# 6 System Models

## 6.1 Use Case Model

Figure 1 provides the use case model for the monopoly game. We provide descriptions for different use cases below.

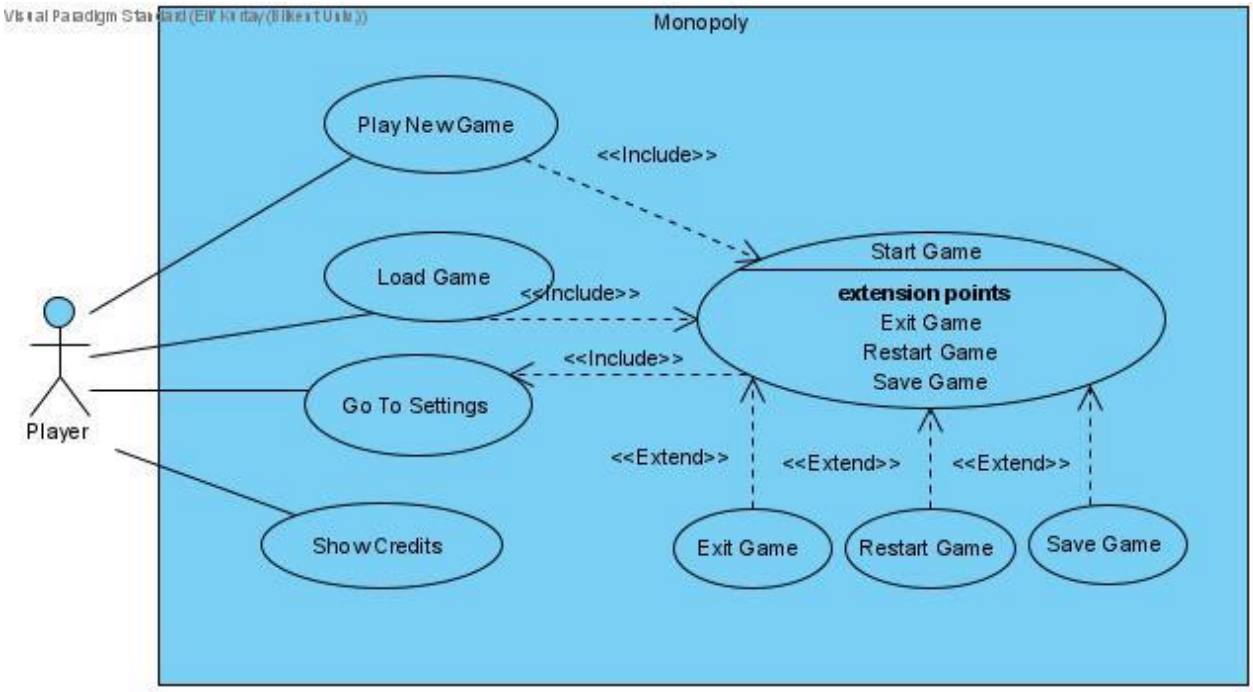


Figure 1. Use case model for the Monopoly game.

**Use Case Descriptions**

*Use Case #1:*

1. Name: Show Credits
2. Participating actor: Player
3. Entry condition:
   - Player must be on the main menu.
   - Player must click the “Credits” button on the main menu.
4. Exit condition:
   - Player clicks on the “Back” button on the credits page.
5. Flow of events:
   5.1. Player clicks the “Credits” button on the main menu.
   5.2. System renders the credits page.
   5.3. Player can read the credits.
   5.4. Player clicks the “Back” button when they want to go back to the main menu.
6. Special/quality requirements:
   - No special/ quality requirements

*Use Case #2:*

1. Name: Go To Settings
2. Participating actor: Player
3. Entry condition:
   - Player is on the main menu or playing the game.
   - Player clicks on the “Settings” button on the screen.
4. Exit condition:
   - Player clicks on the “Back” button on the settings page.
5. Flow of events:
   5.1. Player clicks on the “Settings” button.
   5.2. System renders the settings screen.
   5.3. Player changes the settings that are about adjusting the music and sound volumes.
   5.4. System adjusts the volume as requested from the Player.
   5.5. Player exits by clicking on the “Back” button on the screen.
6. Special/quality requirements:
   - System should have access to the computer's volume.

*Use Case #3:*

1. Name: Load Game
2. Participating actor: Player
3. Entry condition:
   - Player clicks on the “Load Game” button on the main menu.
   - Player should have previously played and saved games on the same computer.
4. Exit condition:
   - Player clicks on the “Back” button on the screen.
   - Player clicks on the “Yes” button on the prompt for affirmation after selecting a save and clicking on the “Start Game” button.
5. Flow of events:
   5.1. Player clicks on the “Load Game” button on the main menu.
   5.2. System displays the load game selection screen.
   5.3. Player chooses from their previously saved games.
   5.4. System loads the game settings from the chosen save data and requests affirmation from Player to start the game.
   5.5. System starts the game and renders the game screen.

6. Special/quality requirements:
    - Player cannot change the game settings of the loaded game.
    - Player needs to have at least one previously loaded and not finished game saved on their computer.
    - If Player does not have previously saved games, the system should prompt the user to select the “New Game” option in the main menu.

*Use Case #4:*

1. Name: Play New Game
2. Participating actor: Player
3. Entry condition:
    - Player clicks on the “New Game” button on the main menu.
4. Exit condition:
    - Player clicks on the “Back” button on the screen.
    - Player clicks on the “Start Game” button on the screen.
5. Flow of events:
    5.1. Player clicks on the “New Game” button on the main menu.
    5.2. System renders the game settings screen.
    5.3. Player can add other players and give usernames to all players by pressing on the keyboard. Player also can delete players.
    5.4. Player can choose which map to play on.
    5.5. Player can choose a lap limit.
    5.6. Steps 3, 4, and 5 can be repeated with any order and as many times as wanted by the Player.
    5.7. Player clicks the “Start Game” button on the screen when s/he is finished adjusting the game settings.
    5.8. System starts the game with the given game settings.
    5.9. System renders the game screen.
6. Special/quality requirements:
    - Player can add a maximum of four players including themselves.
    - The system will display a preview for the selected maps.
    - If less than four players are given, the system will auto-generate players to reach four players.
    - While adding a player, Player needs to select a character for that player.

*Use Case #5:*

1. Name: Start Game
2. Participating actor: Player
3. Entry condition:
   - Player clicks the “Yes” button on the loading game screen or the “Start Game” button on the game settings screen.
4. Exit condition:
   - System shows the winner when an ending condition is met in the game.
5. Flow of events:
   - 5.1. System renders the game screen.
   - 5.2. System requests all Players to roll dice before the game starts to decide on the player's turn order.
   - 5.3. System starts the first turn of the players and in their turn asks them to choose their character.
   - 5.4. System starts the turn of a player buy letting them take the actions of rolling dice, saving, restarting, and exiting the game. Player must roll the dice to start their Player-related actions.
   - 5.5. Player who rolls the dice, changes their spot on the board according to the face values of the dice. During their turn, a Player might have the opportunity to buy the property they are on, might pay tax, might have to pay rent to the owner of the property, might spin the wheel of fortune, might use their characters power, might go to jail, might draw a card, might do nothing or a combination of the aforementioned actions. The Player can trade, build buildings on, sell, and mortgage their properties in their turns as well.
   - 5.6. Player presses “Finish Turn” when they no longer want to perform an action in their turn.
   - 5.7. System starts the turn of the next Player.
   - 5.8. Players repeat Steps 4, 5, and 6 until one of the ending conditions are met.
   - 5.9. System shows the winner on the screen and prompts the user to go back to the main menu.
6. Special/quality requirements:
   - At any time during the game, the player can save their progress, quit the game, restart the game, or view the settings.
   - The ending conditions are either reaching the end of their lap limit or having three players go bankrupt.

*Use Case #6:*

1. Name: Restart Game
2. Participating actor: Player
3. Entry condition:
   - Player must be playing the game.
   - Player clicks the “Yes” button on the prompt after clicking the “Restart” button on the game screen.
4. Exit condition:
   - System renders the game screen with change on the game data.
5. Flow of events:
   5.1. Player clicks the “Yes” button on the prompt after clicking the “Restart” button on the game screen.
   5.2. System resets the game data to the starting preferences.
   5.3. Player starts playing the game with the initial settings.
6. Special/quality requirements:
   - System needs to prompt the user for affirmation after the user clicks the “Restart” button.
   - System needs to reset lap count, players’ assets, and money.

*Use Case #7:*

1. Name: Save Game
2. Participating actor: Player
3. Entry condition:
   - Player must be playing the game.
   - Player clicks the “Yes” button on the prompt after clicking the “Save” button on the game screen.
4. Exit condition:
   - System renders the game screen.
5. Flow of events:
   5.1. Player clicks the “Yes” button on the prompt after clicking the “Save” button on the game screen while playing the game.
   5.2. System updates the save data of the current game.
   5.3. System renders the main menu screen.
6. Special/quality requirements:
   - System needs to prompt the user for affirmation after the user clicks the “Save” button.
   - System names the save according to the date and time of the start of the played game.

*Use Case #8:*

1. Name: Exit Game
2. Participating actor: Player
3. Entry condition:
    - Player must be playing the game.
    - Player clicks the “Yes” button on the prompt after clicking the “Exit” button on the game screen.
4. Exit condition:
    - System renders the main menu screen.
5. Flow of events:
    5.1. Player clicks the “Yes” button on the prompt after clicking the “Exit” button on the game screen while playing the game.
    5.2. System posts the last save of the current game to the database.
    5.3. System renders the main menu screen.
6. Special/quality requirements:
    - System needs to have a connection to the database or give a warning to the Player if there is no connection.
    - System needs to prompt the user for affirmation after the user clicks the “Exit” button.

## 6.2 Dynamic Models

### 6.2.1 Activity Diagrams

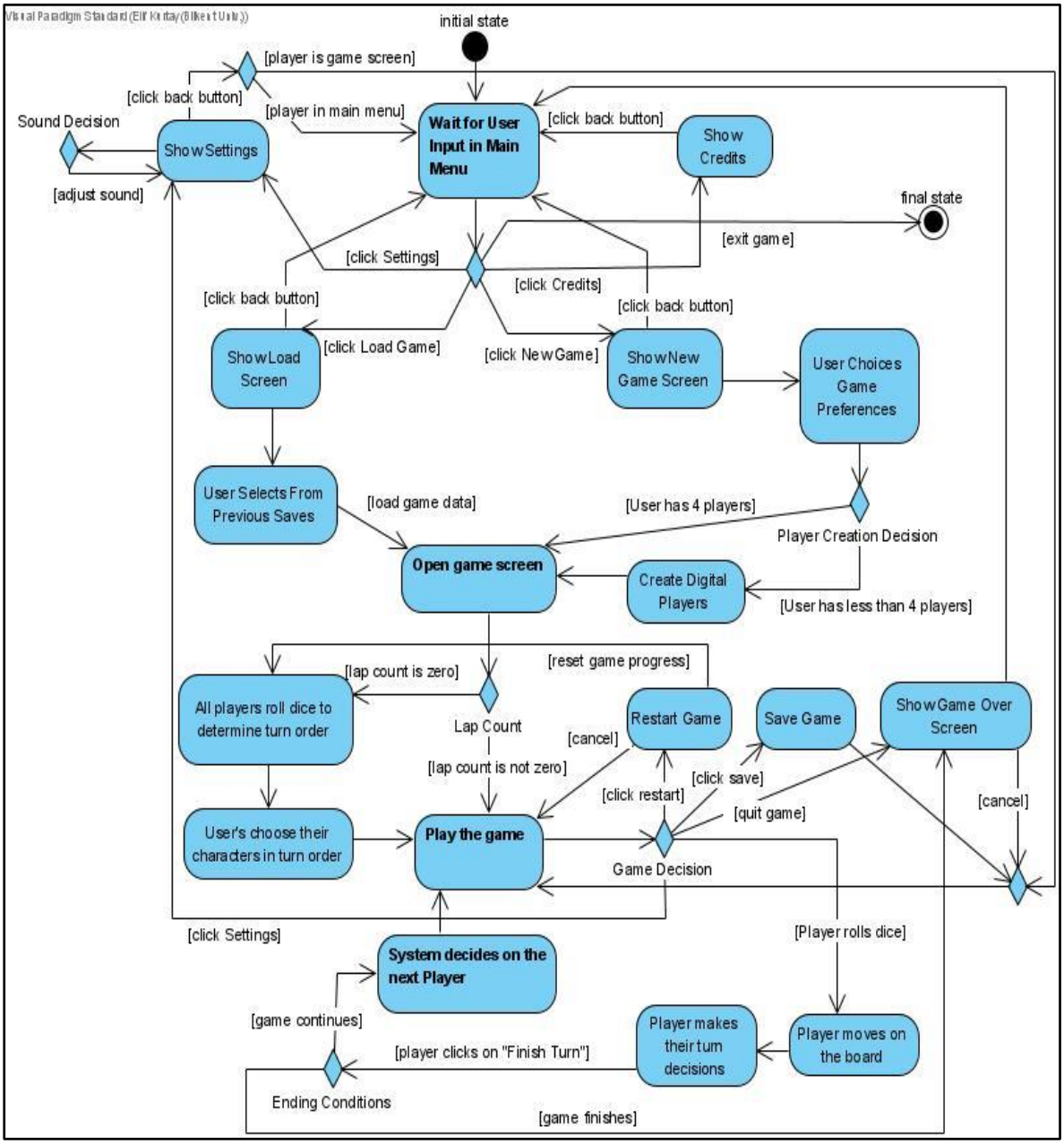


Figure 2. Activity diagram that shows how the system runs the program.

Figure 2 shows the activity diagram that shows how the game runs. The system initializes the program by opening the main menu screen and then waits for user input to give a decision. The player can choose to go to settings for sound and music adjustment, go to the credits page, exit the program, load a previously played game, or start a new game. The player can come back to the main menu after performing any of the actions by clicking the back button.

If the load game choice is selected, the system will gather data from the database and show the load screen to the user. Then the system will wait for a selection from the saved games. After the selection

of a saved game, the system will load game data from that save and open the game screen with previous data. If the new game choice is selected, the system will show the new game screen which has multiple sections for different game settings including adding players, choosing a map, and deciding on a lap limit. After the user is done with their desired game setting preferences, the system will wait for a “start game” action from the player. When this action comes, the system needs to decide regarding multiplayer or single-player game mode. If the player entered less than 4 players, the system would create digital “fake” players to complete the player number to four. Making sure there are four players, the system will render the game screen with default data where lap count is zero.

When the game screen is loaded, the system checks lap count. If the lap count is zero, the system starts a special sequence where all players need to roll the dice and then choose their characters (tokens) before the game starts. After this sequence completes, the system starts playing the game that is active if the lap count was not zero when the game screen was loaded.

During the game, each player has the same decisions in their turns. They can restart the game which resets game history and reduces the lap count to zero. Hence, after the restart, the system goes back to the special sequence for new games. They can save the game which saves game data to the database to be loaded later. They can visit the settings page. They can exit the game, which makes the system show the game over screen with players’ scores. The player can cancel their decision to exit the game and go back to playing the game or the system will render the main menu screen. In the game decision, the player can also choose to roll the dice. After this decision, the system moves the player according to the dice roll and enables the turn actions of the player. Turn actions of the player can be trading, buying property, paying rent, building on their property, using their characters’ special features, mortgaging, redeeming, drawing cards, paying tax, or playing wheel of fortune. The turn decisions can be repeated. After the player concludes their turn decisions, the system gets the “finish turn” input. When this input comes, the system checks for any game-ending condition for the player. These conditions are reaching the lap limit or going bankrupt. If one of the conditions is met, the system will show the game over screen. Otherwise, the system will decide on the next player and restart the “playing the game” loop for that player. If the next player is a “fake” player, then the system will play the default actions for this autogenerated player.

### 6.2.2 Sequence Diagrams

***Scenario 1: Changing the Settings***

The player wants to change the volume, background, and other settings. The user enters the main menu and presses the Settings Button. Then, the Settings class handles these adjustments, and the game is therefore personalized for the player (see Figure 3).

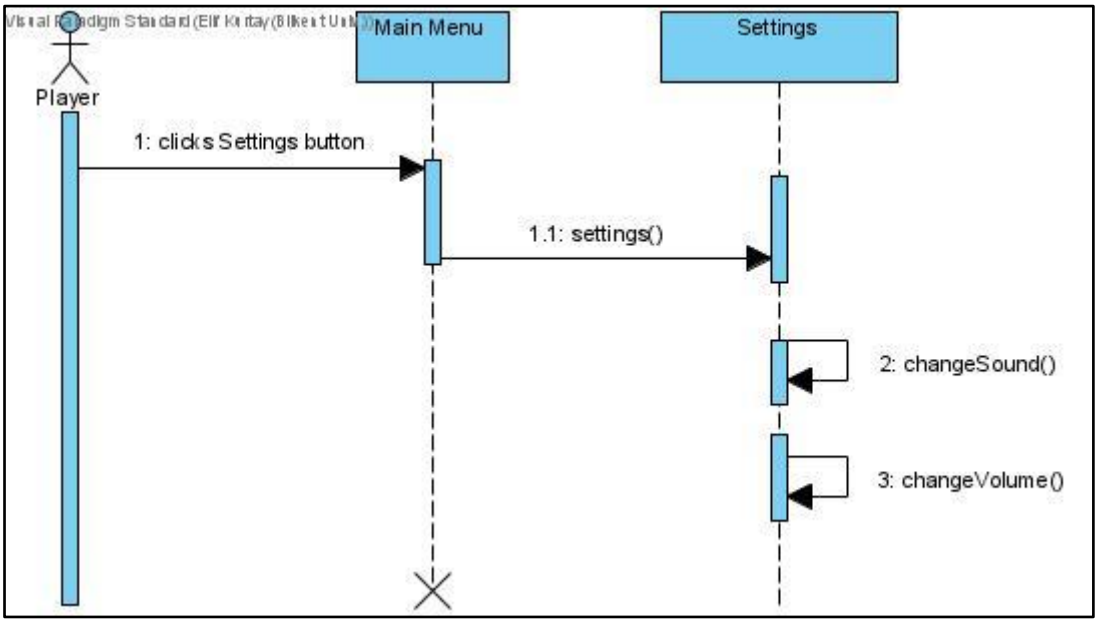


Figure 3. Sequence diagram for changing the settings.

***Scenario 2: Looking at the Credits***

The player wants to know who in the world could have written this perfect game, so he enters the main menu and clicks the cool-looking Credits Button, where he learns about the awesome and funny developers (see Figure 4).

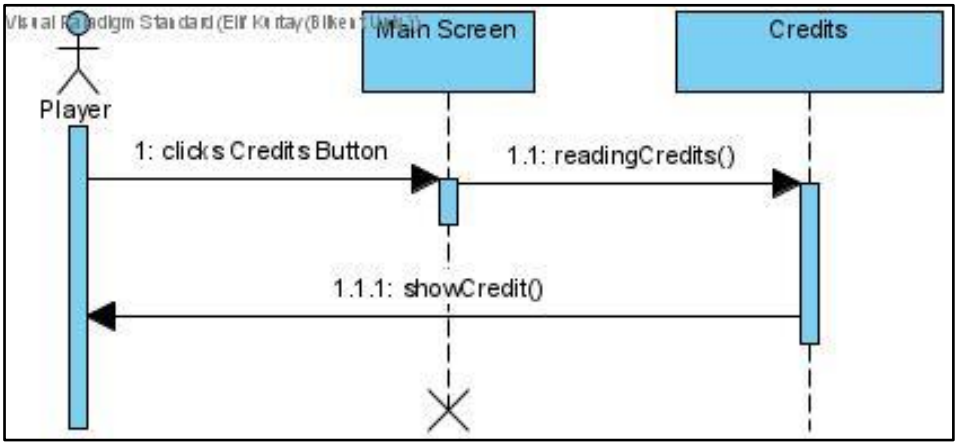


Figure 4. Sequence diagram for taking a look at the credits.

***Scenario 3: Loading a Previously Started Game***

When the player wants to load the last game that was not finished, s/he enters the main menu and clicks the Load Game button. The Main Menu calls the *loadGame* function, which opens the Load Game Screen. On this screen, the player can select the save s/he wishes to continue. The Game class activates the constructor Game (Loaded Game: Game). The constructor gives the players previously stored values instead of initializing everything, which are taken from the database**.** Then, the game loop continues from where it was left (see Figure 5).

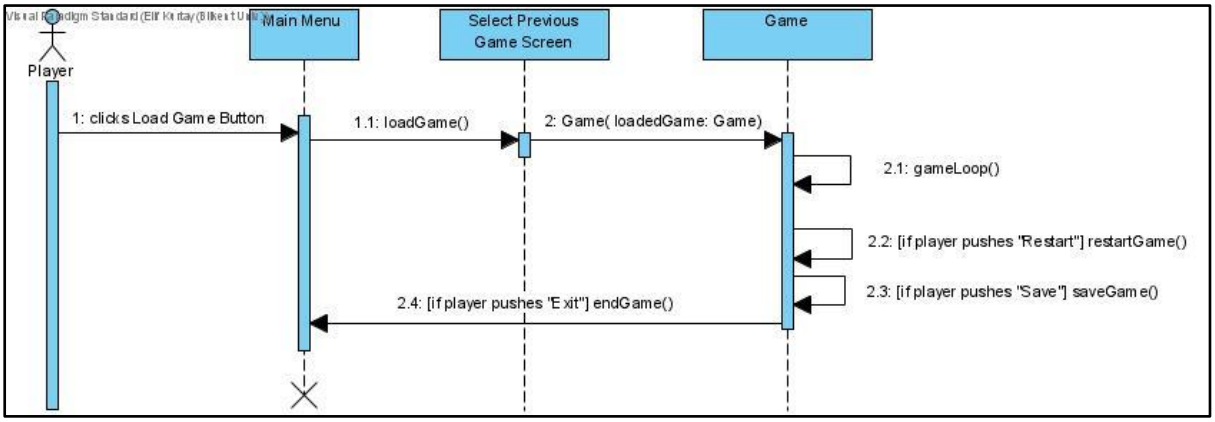


Figure 5. Sequence diagram for loading a previously started game.

***Scenario 4: Playing the Game***

The player wants to play the game. The player enters the main menu and presses the Start Game button. Then the player enters the players, turn limit, and the map of their choice in the New Game Settings screen. The number of players is saved to the *Game* class as a property (attribute) in the *playerCount* variable. After the number of players and bots are determined, the *Game* class initializes the game and creates the Board, the Bank, and The Event Handler. Then the game loop starts. The Board places the characters on the board, randomizes the drawable cards, sets the initial volume, and other settings. Players' attributes determine whether a player can roll the dice on that turn or not. Similarly, the operations of the characters that the players possess determine how the characters interact with each other. In any turn, the player can save, restart, and exit the game (see Figure 6).

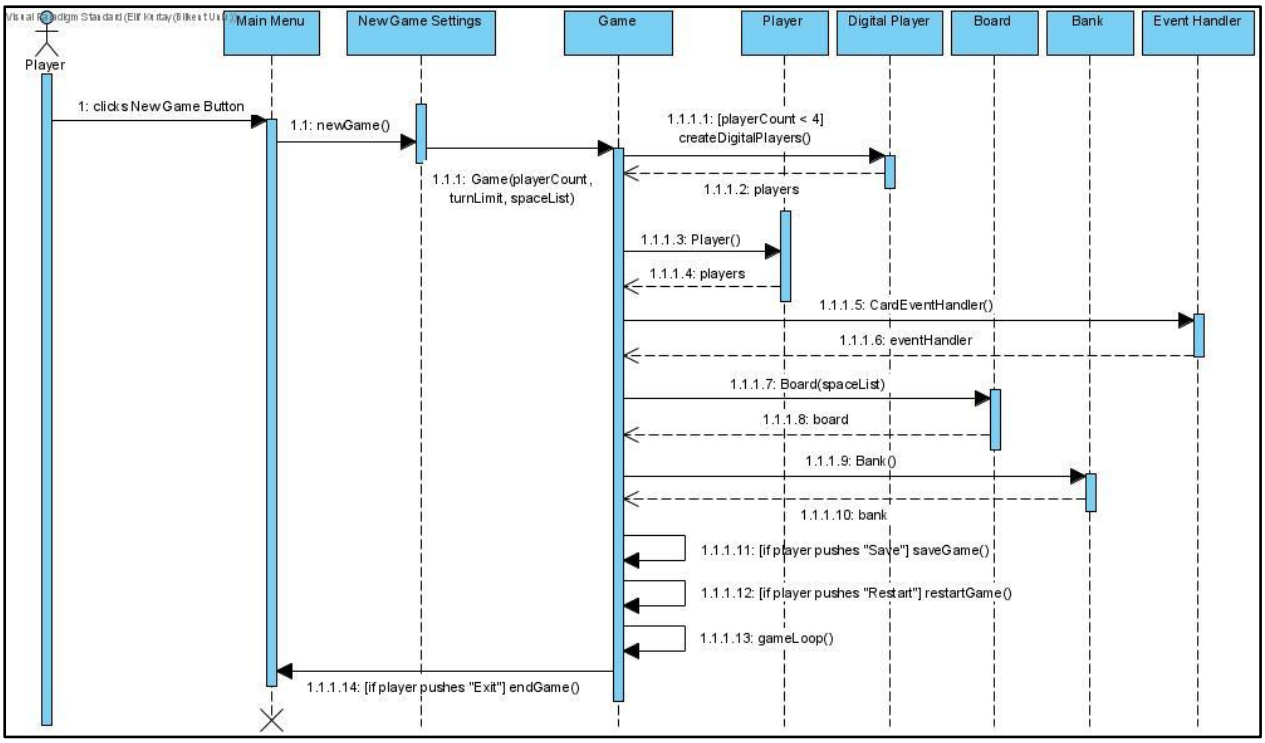


Figure 6. Sequence diagram for playing the game.

### 6.2.3 State Diagrams

***Player Turn State Diagram***

Figure 7 gives the Play Turn state diagram. The initial state of this diagram is “Start Turn”. The “Dice State” will directly follow the initial state. In this state when the player rolls the dice the next state will be “Move State”. If the player rolled double thrice the player will be sent to the “Jail” state. If the player is already in jail, they will be sent directly to the “Operations” state. In the “Move State” the next state (“Property”, “Community Chest”, “Chance”, “Wheel of Fortune”, “Tax”, “Go to Jail”, “Go”, “Jail”) will depend on the space the player landed on.

If the player lands on the “Go to Jail” state, the player will be imprisoned and will be moved to the “Jail” state. After the “Jail” state the next state will be the “Operations” state.

In the “Property” state, if the property is owned by another player the player will pay the rent. If the player does not have enough money to pay the rent, their balance will go under zero and they must make their balance positive with certain operations before they press the “finish turn” button. If they press the “finish turn” button before they make their balance positive, they will be bankrupt. In the “Property” state if the property is unowned, then the player can buy that property. If they decide to not buy that property, an auction will be started for that property. So, the next state will be the “Auction” state.

In the “Community Chest” and “Chance” states, the users must draw a card to proceed to the “Operations” state. Similarly, in the “Wheel of Fortune” state, the users must spin the wheel to proceed to the “Operations” state.

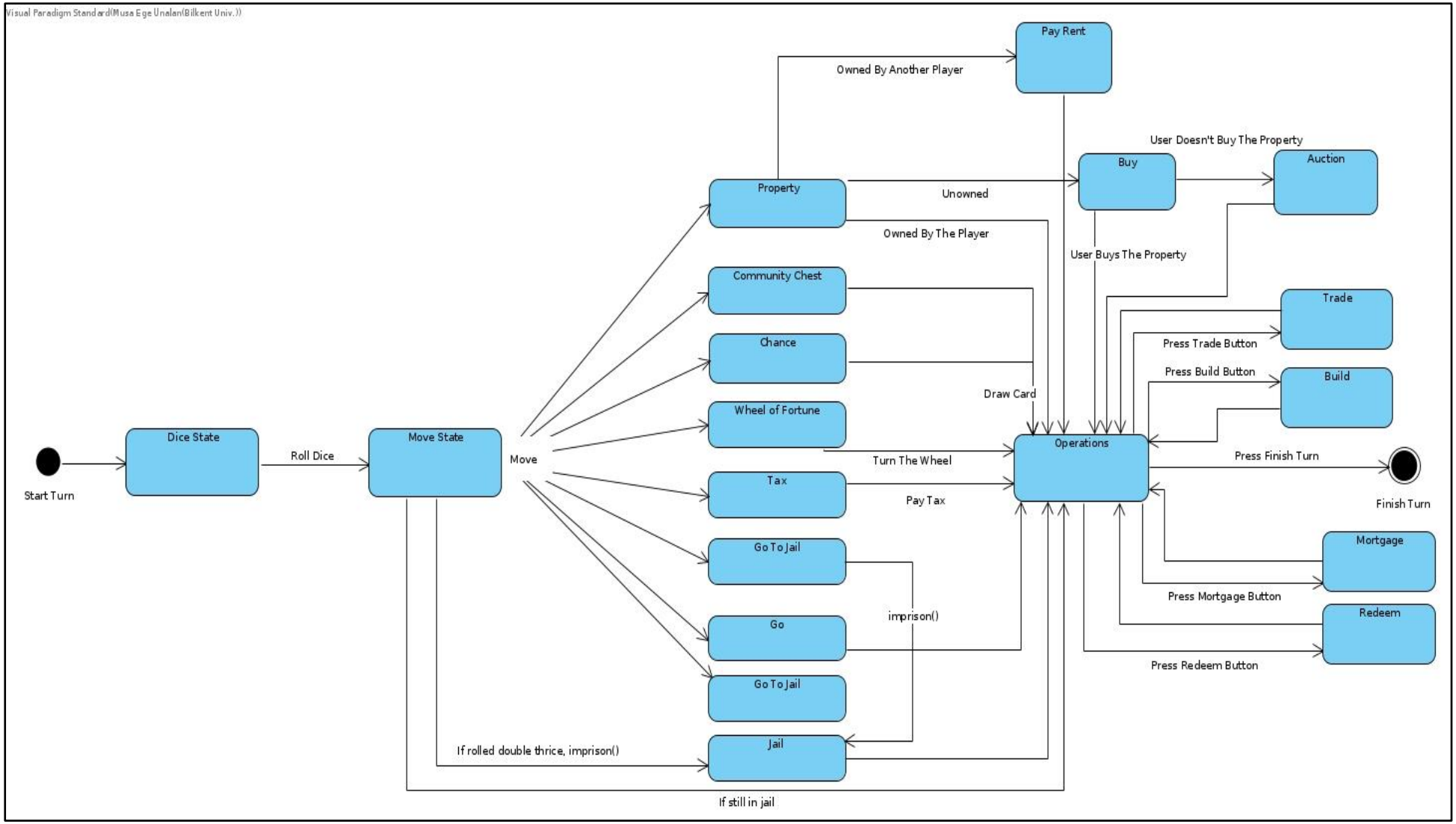


Figure 7. Player turn state diagram.

In the “Tax” state, the player must pay the tax to proceed to the “Operations” state. If the player does not have enough money to pay the tax, their balance will go under zero and they must make their balance positive with certain operations before they press the “finish turn” button. If they press the “finish turn” button before they make their balance positive, they will be bankrupt.

In the “Go” state, the player will receive their salary and proceed to the “Operations” state.

In the “Operations” state, the next state will be determined according to the player’s selection. The possible states that can be followed after the “Operations” state “Trade”, “Build”, “Mortgage”, “Redeem”, the final state “Finish Turn”.

### *Auction State Diagram*

Figure 8 shows the auction state diagram. If an unowned property is not bought, an auction for that property will take place. The initial state for the auction state diagram is “Start Auction”. Each player can bid or fold in the “Bidding State”. If all players fold or 120 seconds is over the game will proceed to the “Auction Over” state. If there is at least one bidder left when the auction is over the final state will be “Property is sold”. However, if there are no bidders left when the auction is over the final state will be “Property is not sold”.

### *Trade State Diagram*

Figure 9 shows the trade state diagram. A player can initiate a trade by pressing the trade button when it is their turn. The initial state of the trade action is “Start Trade”. The player can either select a player to send a trade offer or cancel the trade, which will move them to the “Trade Unsuccessful” final state. After the player selects a player to trade with, the state will change to the “Offer” state and they can now make their offer with the properties/money they are willing to give and the properties/money they are requesting from the other party. They can either send their offer or cancel the trade. After sending the offer the state will change to the “Review Offer” state where the other party will review the trade offer and either accept or decline it. If they accept the trade offer the final state will be the “Trade Successful” state if they decline it the final state will be the “Trade Unsuccessful” state.

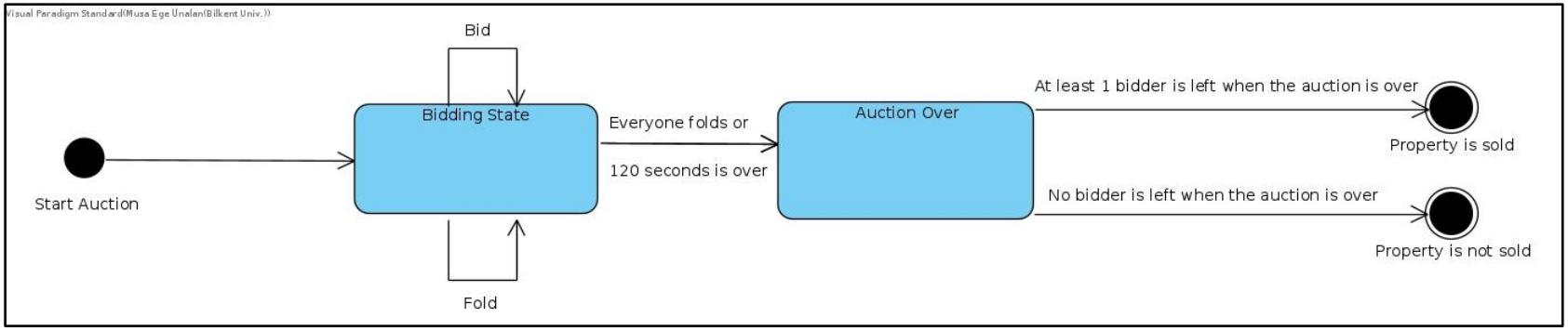


Figure 8. Auction state diagram.

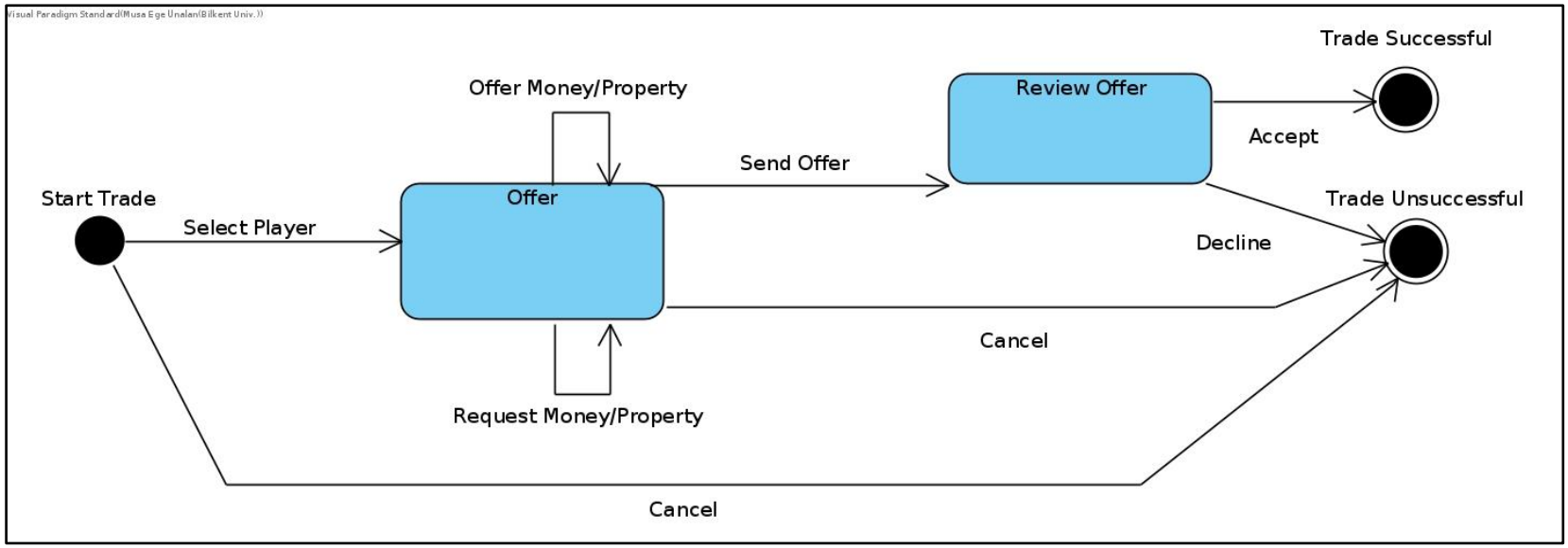


Figure 9. Trade state diagram.

***Mortgage State Diagram***

Figure 10 shows the mortgage state diagram. A player can initiate a mortgage by pressing the mortgage button when it is their turn. The initial state of the mortgage action is “Mortgage”. “Properties Selection” will automatically follow the initial state. At this stage, a player can select or deselect a property. When they press the mortgage button on the screen the state will change to the “Mortgage Successful” which is the final state. If the player decides to press the Cancel button on the “Properties Selection” the final state will be the “Mortgage Unsuccessful” state.

***Redeem State Diagram***

Figure 11 shows the redeem state diagram. A player can initiate a redeem by pressing the redeem button when it is their turn. The initial state of the redeem action is “Redeem”. “Properties Selection” will automatically follow the initial state. At this stage, a player can select or deselect a property. When they press the redeem button on the screen the state will change to the “Redeem Successful” which is the final state. If the player decides to press the Cancel button on the “Properties Selection” the final state will be the “Redeem Unsuccessful” state.

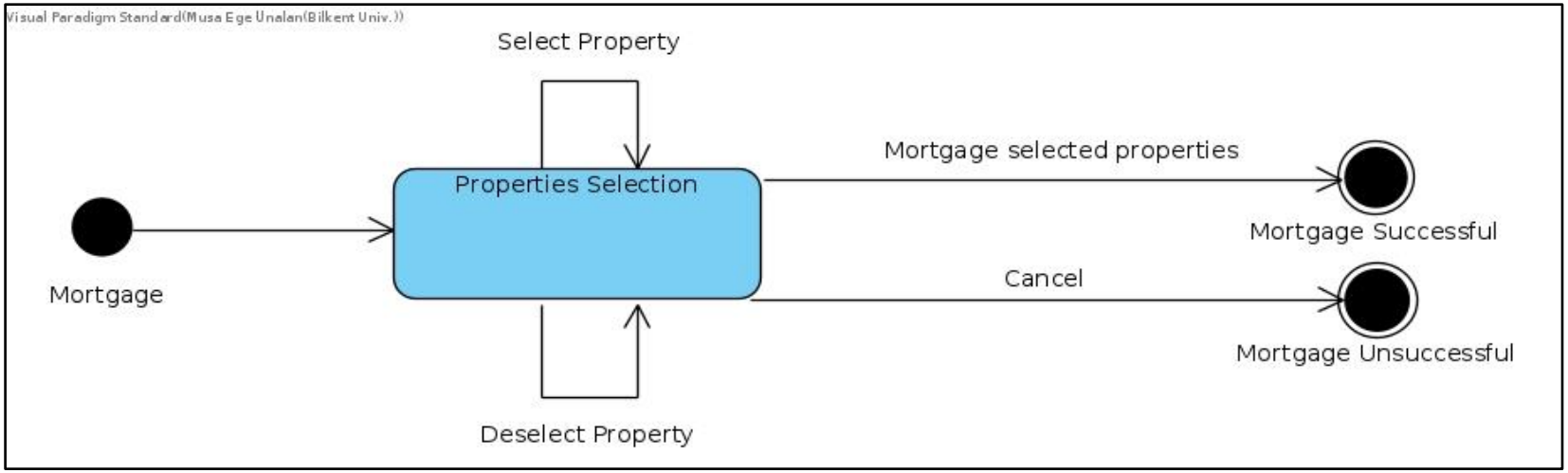


Figure 10. Mortgage state diagram.

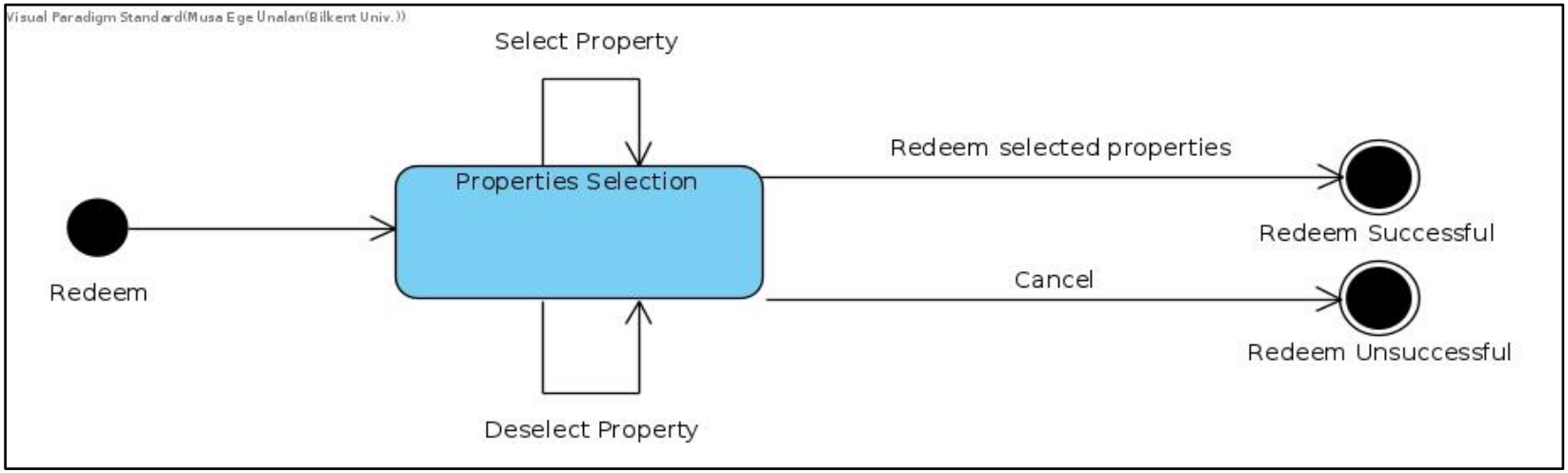


Figure 11. Redeem state diagram.

## 6.3 Object and Class Model

Object and Class Model without the UI classes are provided in Figure 12. The descriptions of the classes are as follows:

- *MainMenu* Class: The MainMenu class is our main UI class that is used for displaying the options New Game, Load Game, Settings, Credits, Quit, their subsequent screens, the board, and the UI of the game after the game starts.

- *Game* Class: The Game class is the main class of our game logic. It stores the current player, the current turn number, the number of players, the turn limit if there is a turn limit and consists of the game board, the players, the bank, and the card event handler.

    - The Game object can be constructed from a *playerCount*, *turnLimit*, and a *spaceList* as a new game, or from a Game object that was saved before.

    - Our game logic will be implemented within the *gameLoop*() function. The dice rolls, turns, bank operations, player interactions with the board and the other players and ending conditions of the game will be handled within this function.

- *Player* Class: The Player class represents each of the players of the game. It holds values such as the player name, their current money, the space they are currently in, if they are bankrupt or not, the properties they own, how many "Get Out Of Jail Free" cards they own, if they are jailed or not, how many Chance or Community Chest cards they have postponed and haven't opened yet and which player token they are using.

    - *Digital Player* Class: The DigitalPlayer class inherits from the Player class and represents the players that are controlled by the computer. All their actions and responses to activities such as trades and auctions are determined by the computer.

- *Token* Class: The Token class represents each of the tokens the players can choose when starting the game. Each token has different advantages and disadvantages. The token class stores the values that are used for the calculations of these advantages and disadvantages.

- *Property* Class: The Property class represents each property a player can own. Each property has its own *TitleDeedCard* that stores its rent, or the values used in calculating its rent and the cost of the building. It stores values such as the number of houses built on the property, if there is a hotel on it, and if the property is mortgaged or not.

- *TitleDeedCard* Class: Each property in the game has its own *TitleDeedCard*. A *TitleDeedCard* must have the name and the mortgage value of the property. There are 3 child classes that inherit from the *TitleDeedCard* class, which are *UtilityTitleDeedCard*, *TransportTitleDeedCard*, and *LandTitleDeedCard* classes.

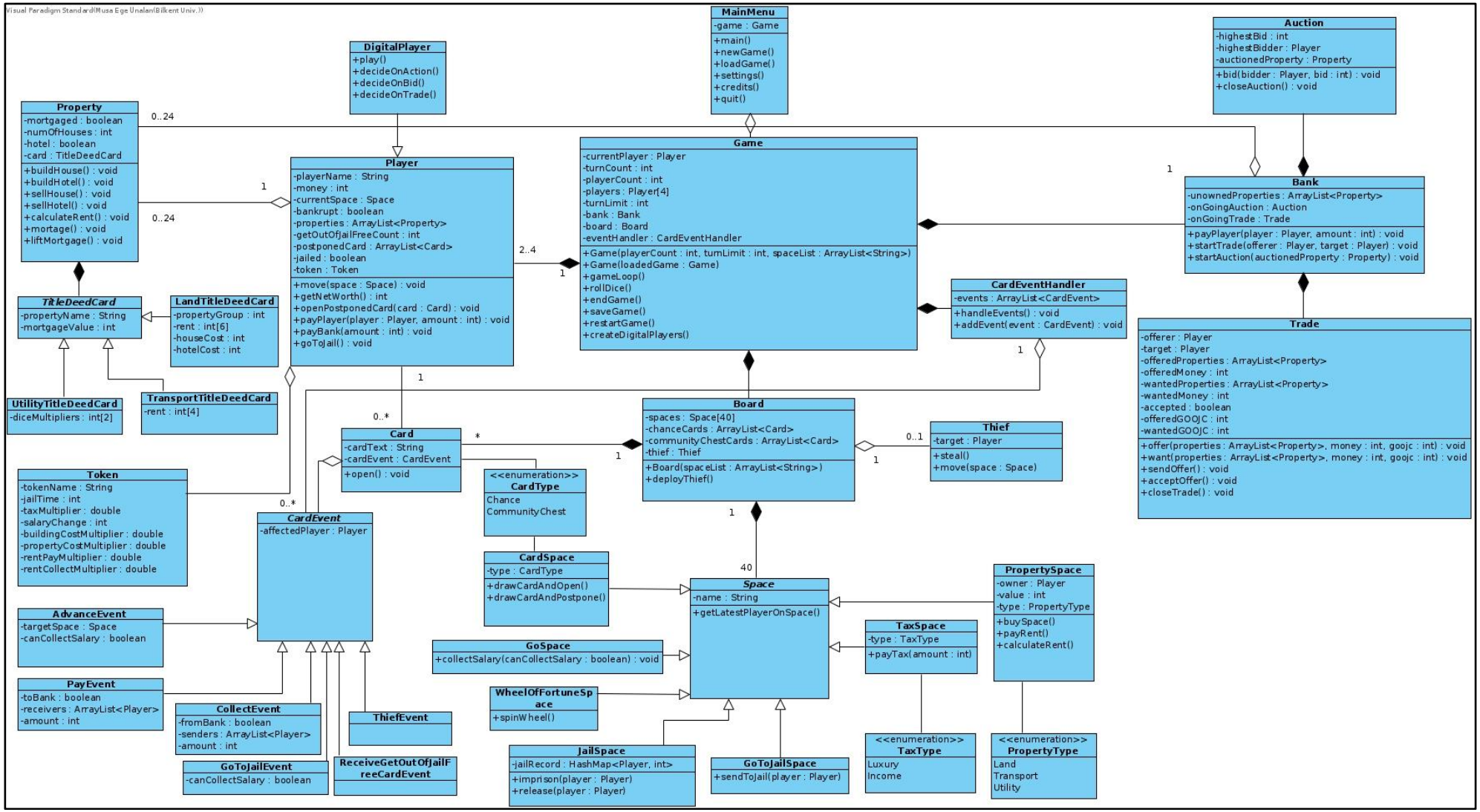


Figure 12. Object and Class Model without the UI classes.

- *LandTitleDeedCard* Class: The LandTitleDeedCard class inherits from the TitleDeedCard class. It holds set values about a land property such as the property group (color group) it belongs to, the rent for each possible state (empty property to having a hotel on it.) of the property, the cost of building houses, and a hotel on it.

- *TransportTitleDeedCard* Class: The TransportTitleDeedCard class inherits from the TitleDeedCard class and holds the rent values for a transport property.

- *UtilityTitleDeedCard* Class: The UtiliyTitleDeedCard class inherits from the TitleDeedCard class and holds the dice multipliers for calculating the rent values for a utility property.

– *Card* Class: The Card class represents the Chance and Community Chest cards from the classic Monopoly game. Each card has a card event, a card text that explains the card event, and has a type which is an enumeration that is either Chance or CommunityChest.

– *CardEvent* Class: The abstract CardEvent class represents events that can occur as a result of opening Chance or Community Chest cards. Each CardEvent has an affected player who is the one who opened the card. The *AdvanceEvent*, *PayEvent*, *CollectEvent*, *GoToJailEvent*, *ReceiveGetOutOfJailFreeCardEvent* classes are the subclasses of the CardEvent class.

- *AdvanceEvent* Class: An AdvanceEvent advances the affected player to a target space and specifies if the player can collect their salary if they pass through the GO space (the starting space) while advancing to the target space.

- *PayEvent* Class: A PayEvent makes the affected player pay a certain amount of money to the bank, to a player or players.

- *CollectEvent* Class: A CollectEvent makes the bank, a player or players pay a certain amount of money to the affected player.

- *GoToJailEvent* Class: A GoToJailEvent sends the affected player to the jail and specifies if the player can collect their salary if they pass through the GO space (the starting space) while advancing to the jail space.

- *ReceiveGetOutOfJailFreeCardEvent* Class: A ReceiveGetOutOfJailFreeCardEvent increases the amount of “Get Out Of Jail Free” cards the affected player has by 1.

- *ThiefEvent* Class: A ThiefEvent deploys a thief to the board, that randomly selects a target player to steal from.

– *CardEventHandler* Class: The CardEventHandler handles the random events that can occur as a result of opening Chance or Community Chest cards. It stores the list events that are not yet handled to handle the case of multiple events happening in one turn, and handles them in the same order they happened.

- *Bank* Class: The Bank class pays the players, holds the unowned properties, manages trades between the players, and conducts an auction if a player decides not to buy a previously unowned space.

- *Auction* Class: The Auction class holds the highest bidder, the highest bid, the auctioned property, and allows the players to bid on the property. When the auction ends the transfer of property is handled by the bank.

- *Trade* Class: The Trade class allows the players to make trades involving money, properties, and "Get Out Of Jail Free" cards. A trade is initiated when a player sends their offer, composed of what they are willing to give and what they want from the opposite side. If the opposite side agrees, the trade is closed and the transfer of assets is handled by the bank.

- *Board* Class: The Board class consists of 40 spaces, has a list of Chance cards, a list of Community Chest cards, and may have a thief. The board is constructed at the beginning of the game from a list of spaces, which may be the default space set of classic Monopoly or a user-defined custom space set.

- *Space* Class: The abstract Space class represents each of the 40 spaces on the classic Monopoly board. *GoSpace*, *WheelOfFortuneSpace*, *GoToJailSpace*, *JailSpace*, *PropertySpace*, *TaxSpace*, and *CardSpace* are all subclasses of the abstract Space class.

    - *GoSpace* Class: The GoSpace class inherits from the Space class and pays salary to players when they go over it if they are not prohibited from receiving a salary.

    - *WheelOfFortuneSpace*: The WheelOfFortuneSpace class inherits from the Space class. It replaces the "Free Parking" space from the Monopoly board. If a player lands on a WheelOfFortuneSpace they will spin a wheel and gain a random property, money, a house on one of their properties if possible, a "Get Out Of Jail Free" card or they may lose money or one of their buildings on one of their properties.

    - *CardSpace*: The CardSpace class inherits from the Space class and has a type which is an enumeration that is either Chance or CommunityChest. If a player lands on a CardSpace they can either draw and open a card from the given type or draw and postpone opening the card.

    - *TaxSpace*: The TaxSpace class inherits from the Space class and has a type which is an enumeration that is either Luxury or Income. If a player lands on a TaxSpace they will have to pay a tax of an amount decided from the type of the TaxSpace.

    - *PropertySpace*: The PropertySpace class inherits from the Space class and has an owner, value, and a type, which is an enumeration that is either Land, Transport, or Utility. If the player lands on an unowned PropertySpace they will have the chance to buy the property associated with that PropertySpace, if they decide not to, the property will go on auction through the bank. If the player lands on a PropertySpace that is not theirs and is owned, they will have to pay rent to the owner of the associated Property.

- *GoToJailSpace* Class: The GoToJailSpace class inherits from the Space class. If a player lands on the GoToJailSpace, they will be immediately sent to jail and will not be able to collect their salary even if they pass through the GoSpace while advancing to the JailSpace.

- *JailSpace* Class: The JailSpace class inherits from the Space class. The players that are sent to jail via either the GoToJailSpace, a Chance card, or by rolling double 3 times are sent to this space. It holds a HashMap<Player, int>, which stores the number of turns that each player currently in jail has spent in jail.

– *Thief* Class: The thief will be deployed on the board after the handling of a ThiefEvent from a card. The thief will select a random player to target upon its deployment and will move towards them every turn. When the thief catches its target, it will steal a random amount of money from the target.

## 6.4 User Interface - Navigational Paths and Screen Mock-ups

In this section, we provide the screen mock-ups for the Monopoly game. Please see Section 3. Functional Requirements for the navigational paths between these screen mock-ups, Section 6.1 Use Case Models for various use-case scenarios based on these screen-mockups, Section 6.2.1 Activity Diagrams that shows the flow of the monopoly game in terms of the game activities, Section 6.2.2 Sequence Diagrams the events that players generate, the order of these events, possible system events for specific use-case scenarios, Section 6.2.3 State Diagrams for representing the players' behaviors using finite state transitions. The list of screen mock-ups are as follows.

– Figure 13 gives the start screen for the Monopoly game.

– Figure 14 gives the screen for loading a game from the existing ones.

– Figure 15 gives the screen for settings.

– Figure 16 gives the screen for adding players offline to the game lobby.

– Figure 17 gives the screen for displaying players offline in the game lobby.

– Figure 18 gives the screen for deleting players offline from the game lobby.

– Figure 19 gives the starting screen for the gameplay

– Figure 20 gives the restarting screen for the gameplay.

– Figure 21 gives the screen for saving the gameplay.

- Figure 22 gives the screen for the end of a game.
- Figure 23 gives the screen for choosing an avatar for a player in the game.
- Figure 24 gives the screen for rolling dice.
- Figure 25 gives the screen for rolling a dice and playing the game according to the dice value.
- Figure 26 gives the screen for buying an asset.
- Figure 27 gives the screen for an auction.
- Figure 28 gives the screen for player choosing.
- Figure 29 gives the screen for a trade offer.
- Figure 30 gives the screen for approving a trade offer.
- Figure 31 gives the screen for a build selection.
- Figure 32 gives the screen for a build approval.
- Figure 33 gives the screen for a mortgage.
- Figure 34 gives the screen for a redeem.
- Figure 35 gives the screen for displaying the assets of a player.
- Figure 36 gives the wheel-of-fortune screen.
- Figure 37 gives the credits screen.

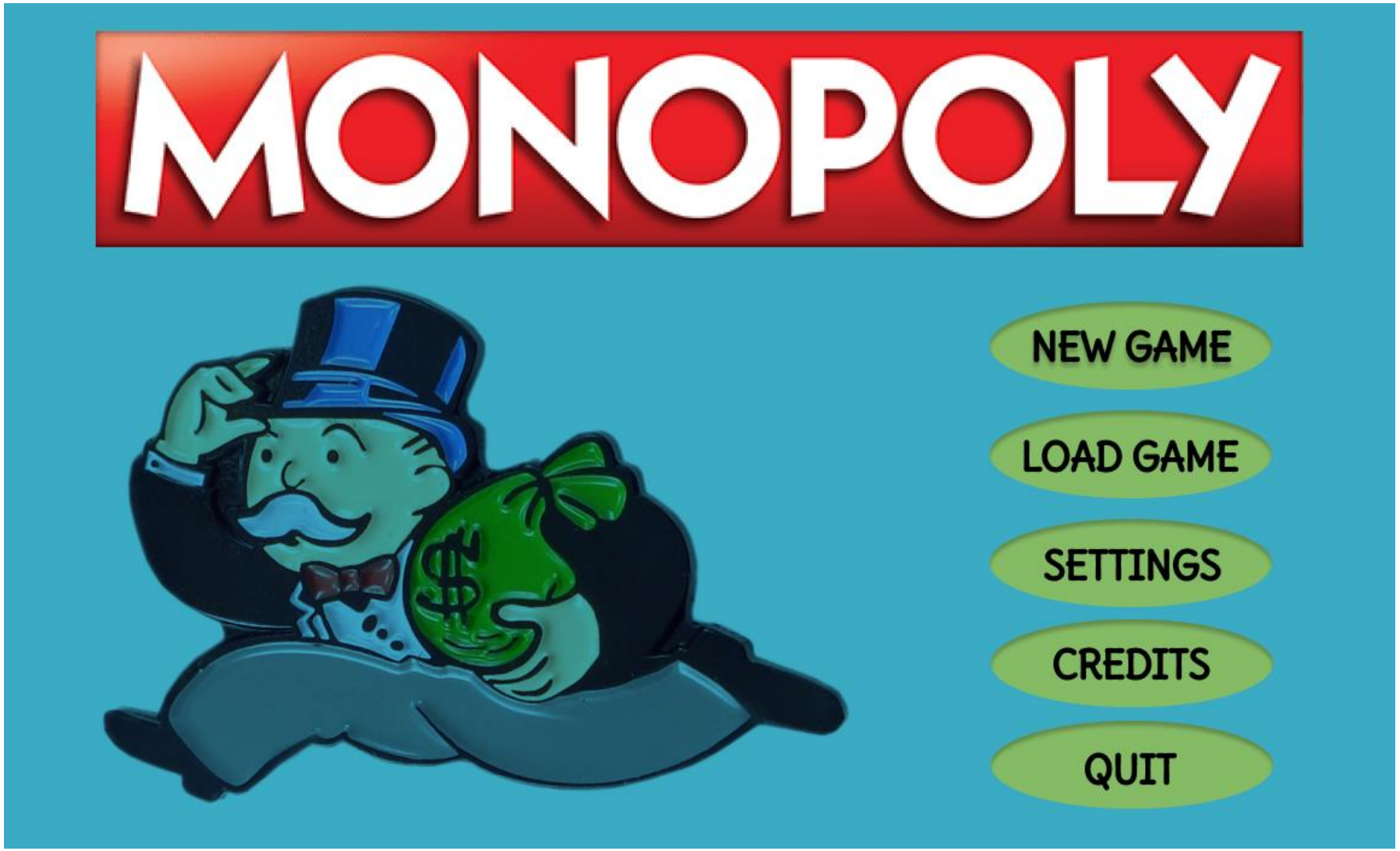


Figure 13. Monopoly game start screen.


Figure 14. Load game screen.

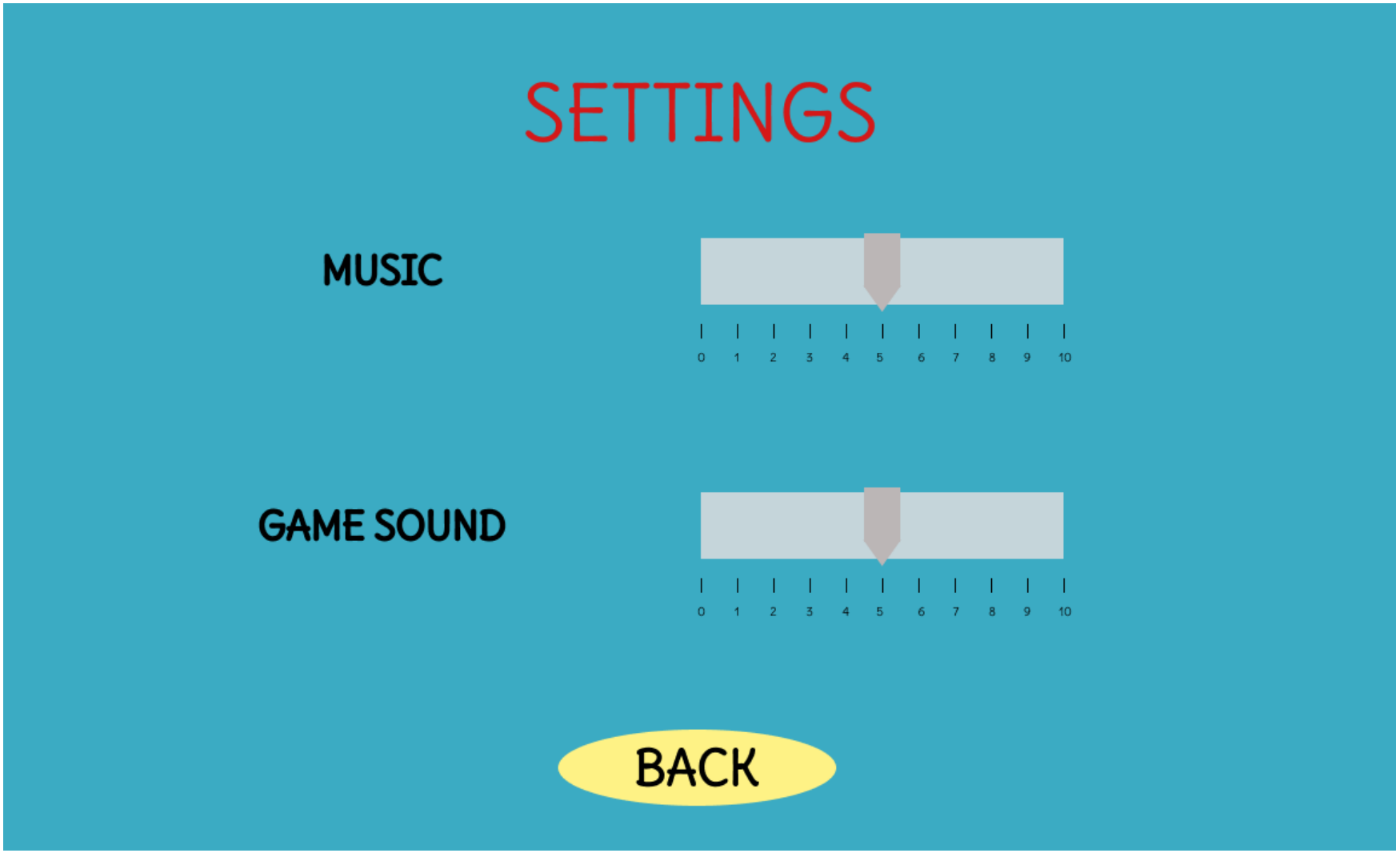


Figure 15. Settings screen.

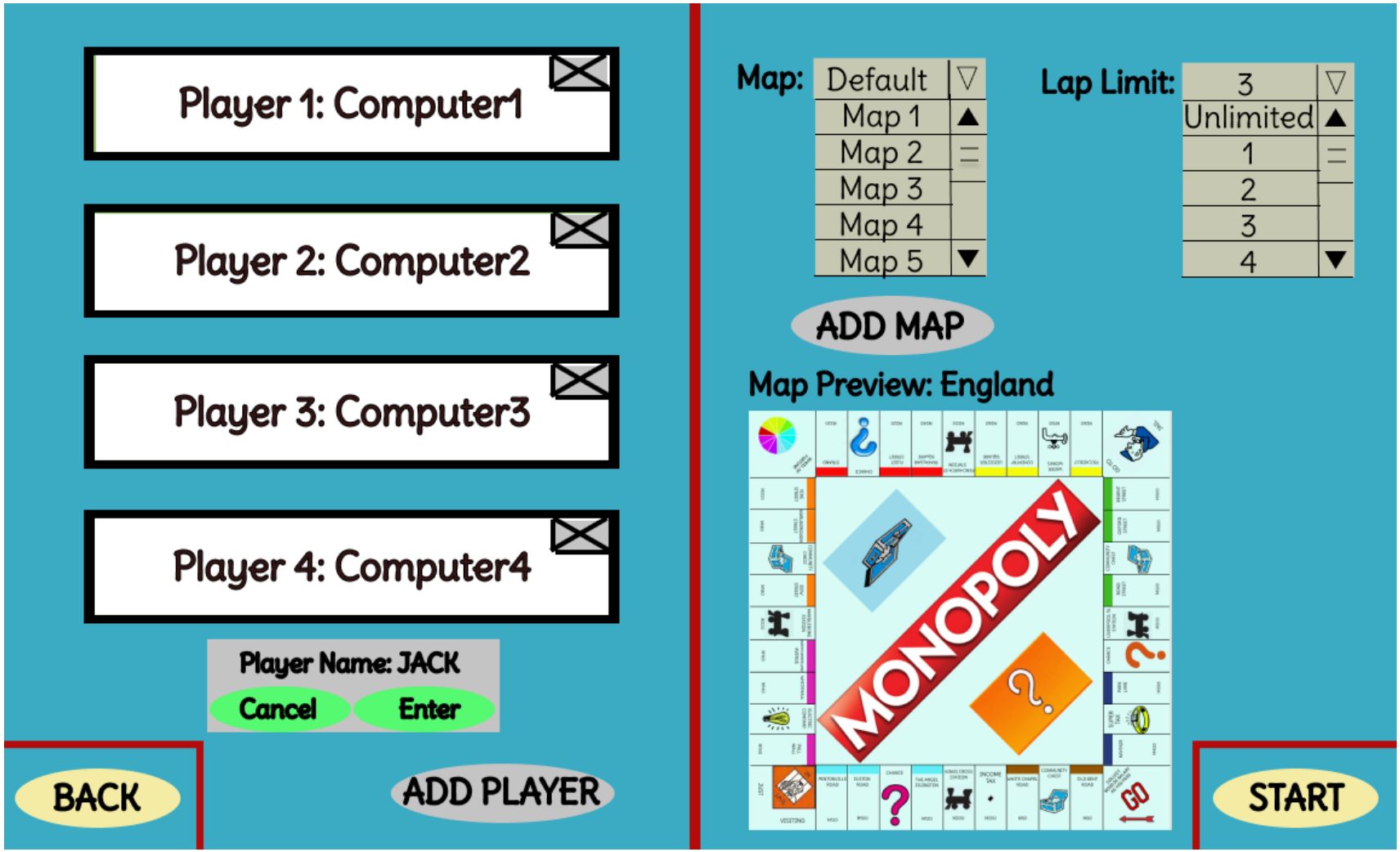


Figure 16. Game lobby offline add player screen.

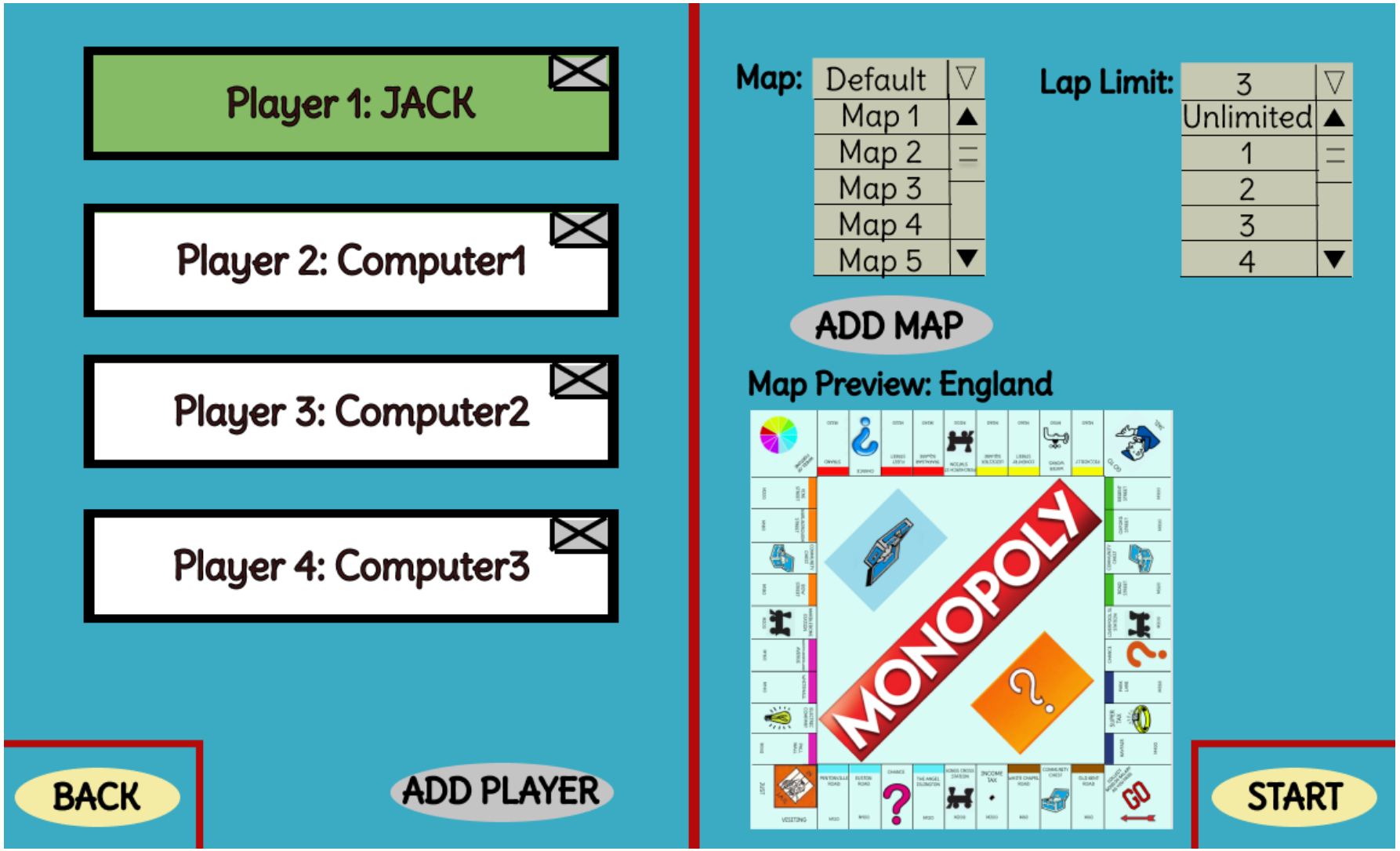


Figure 17. Game lobby offline with players.

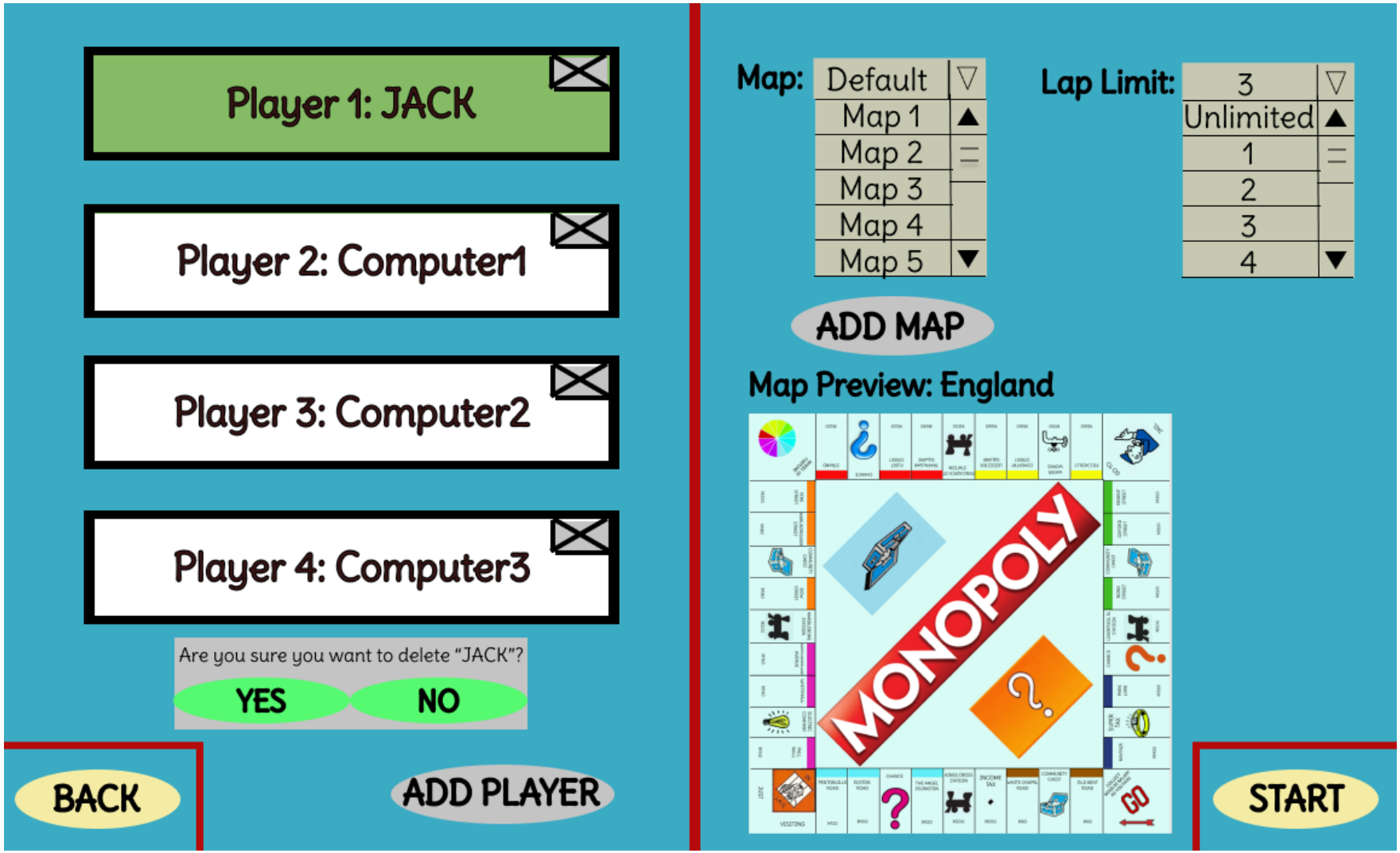


Figure 18. Game lobby offline delete player screen.

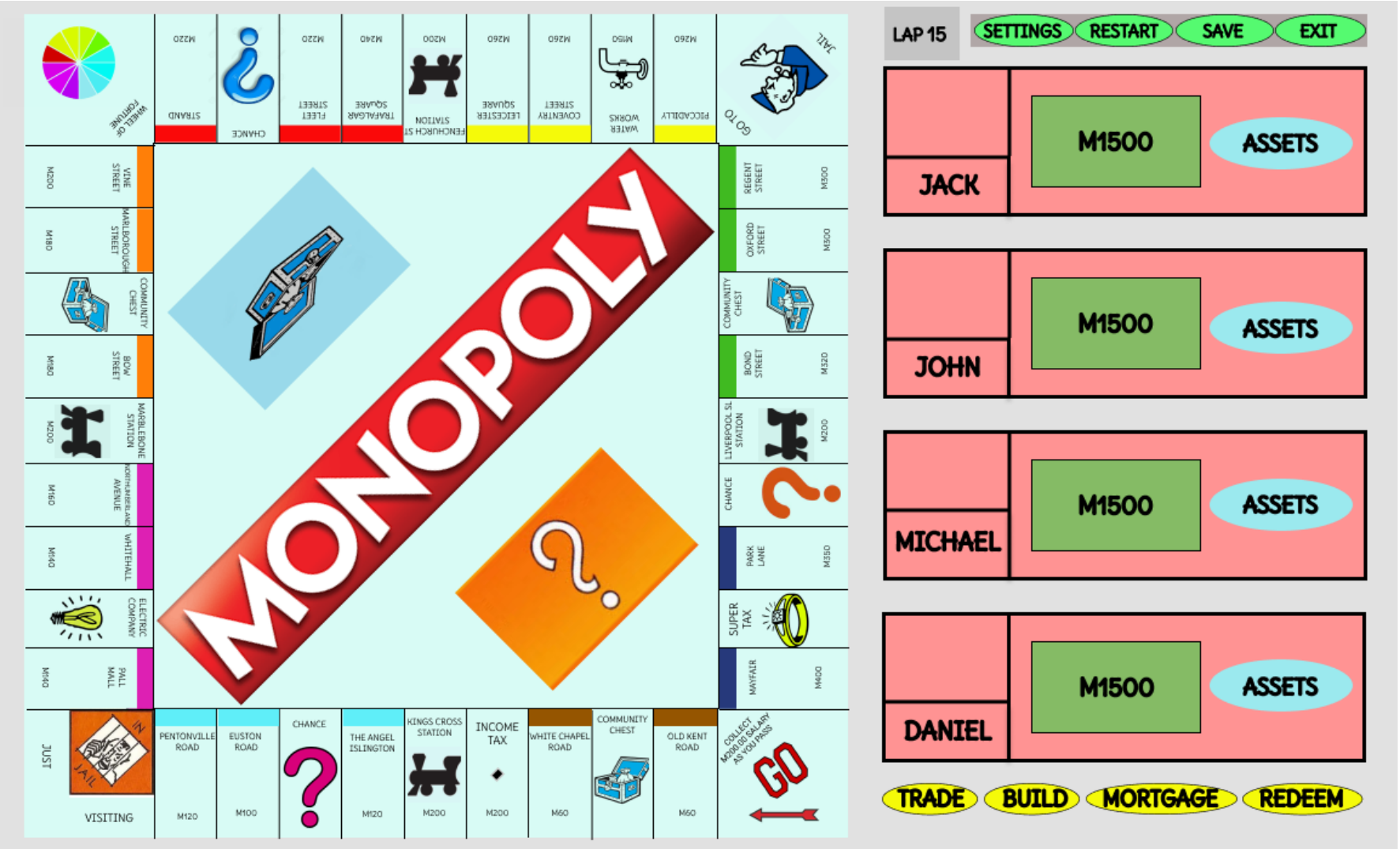


Figure 19. Gameplay starting screen.

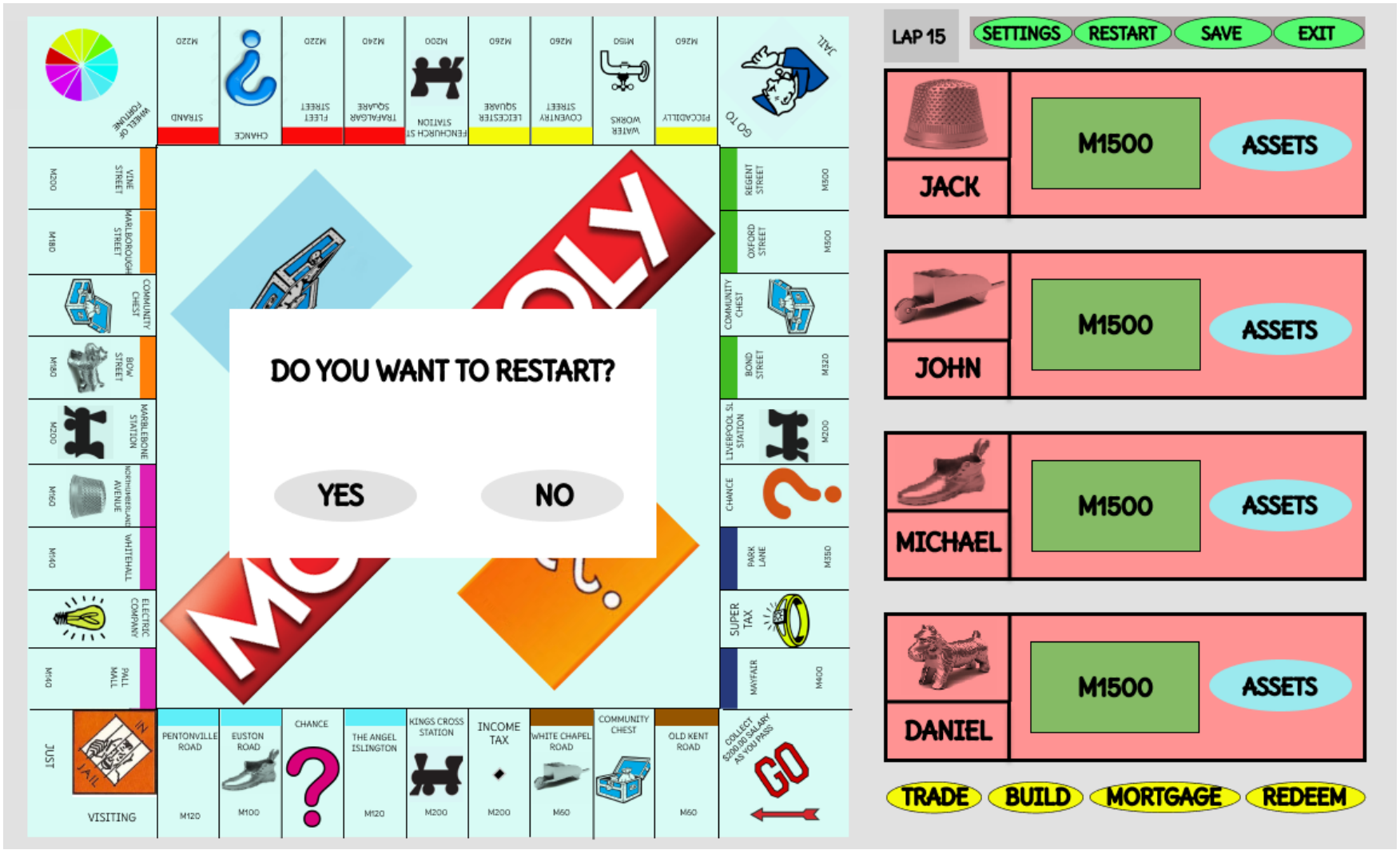


Figure 20. Gameplay restart screen.

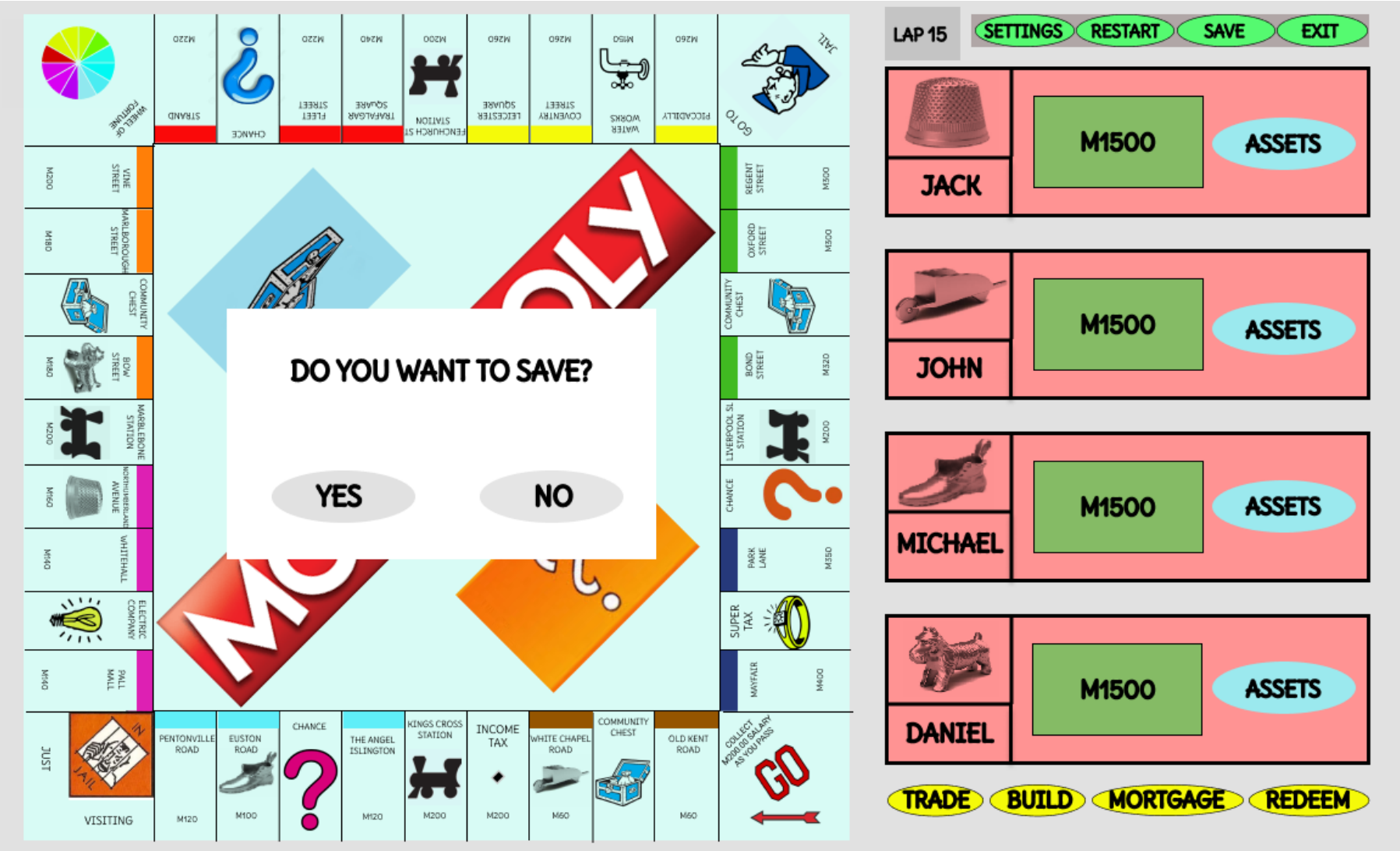


Figure 21. Gameplay save screen.

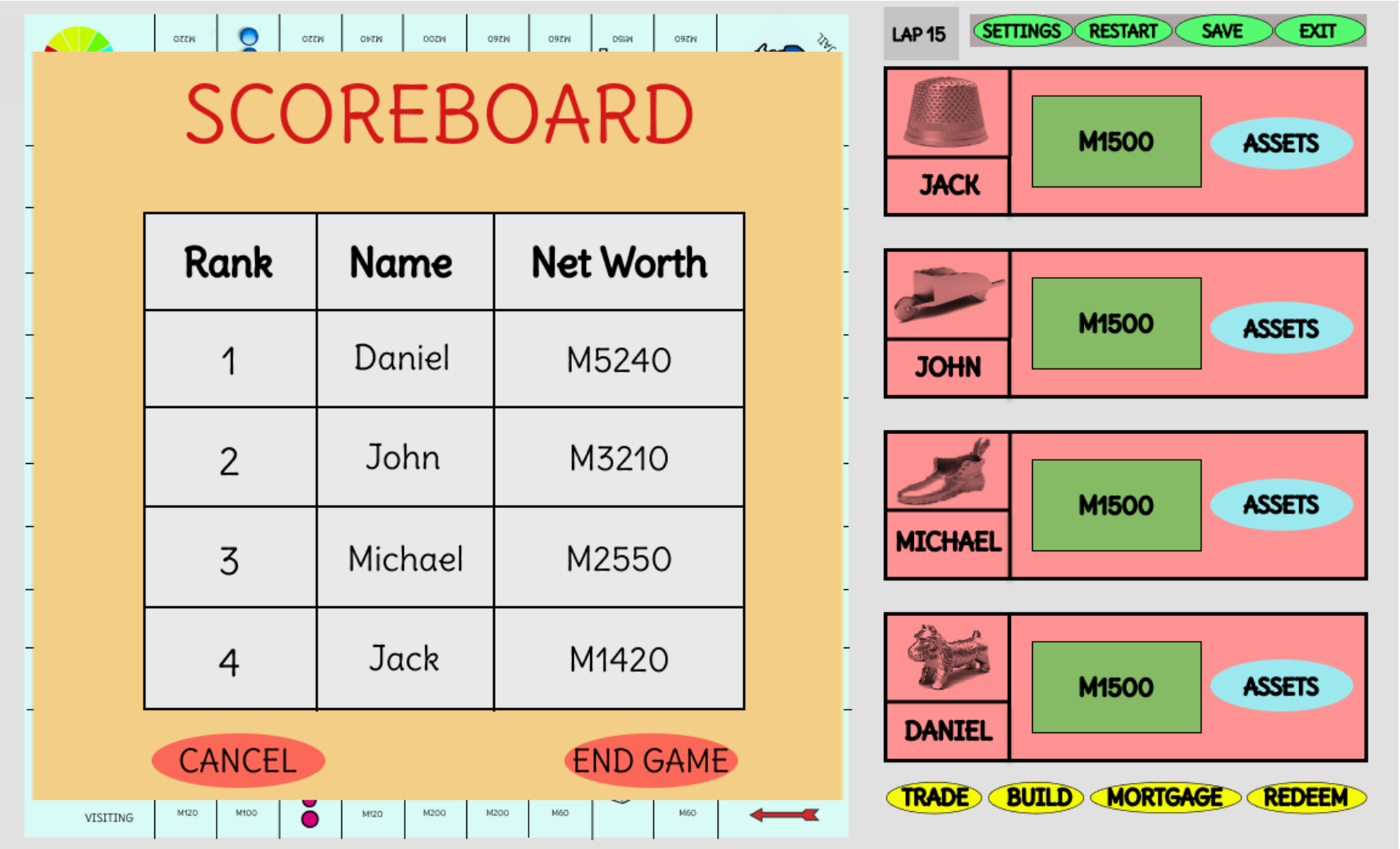


Figure 22. Gameplay end of the game screen.

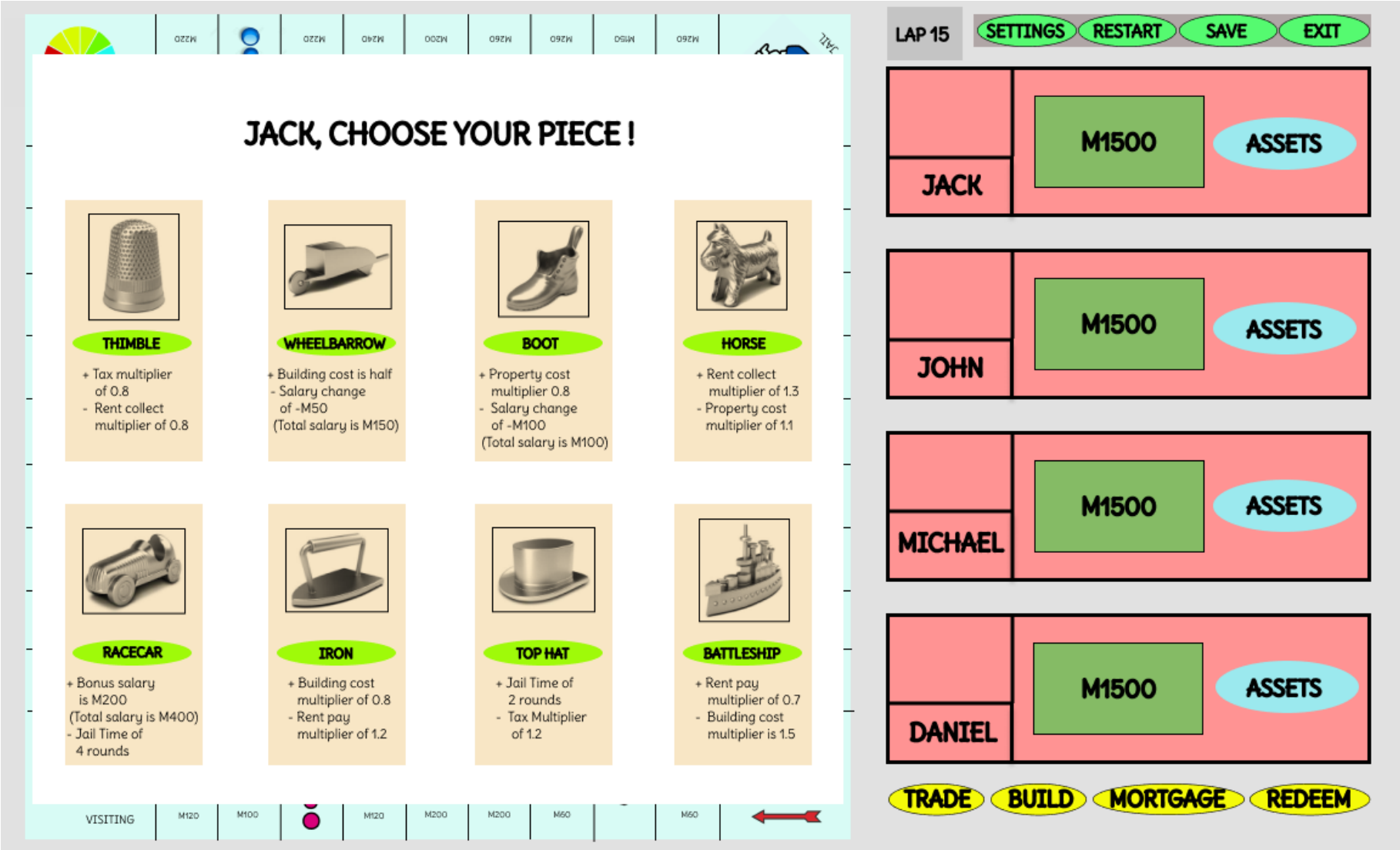


Figure 23. Gameplay choose avatar screen.

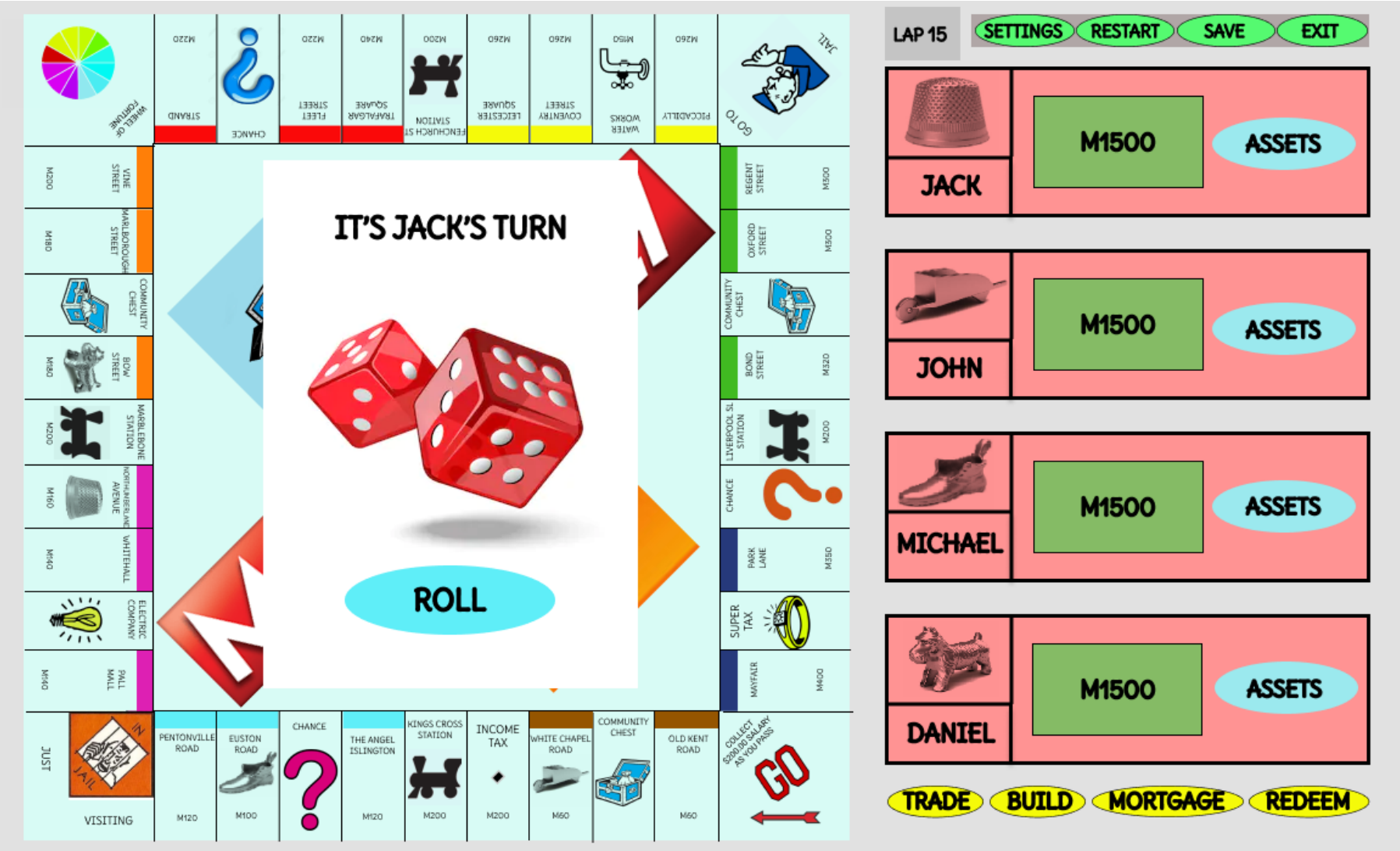


Figure 24. Gameplay roll dice screen.

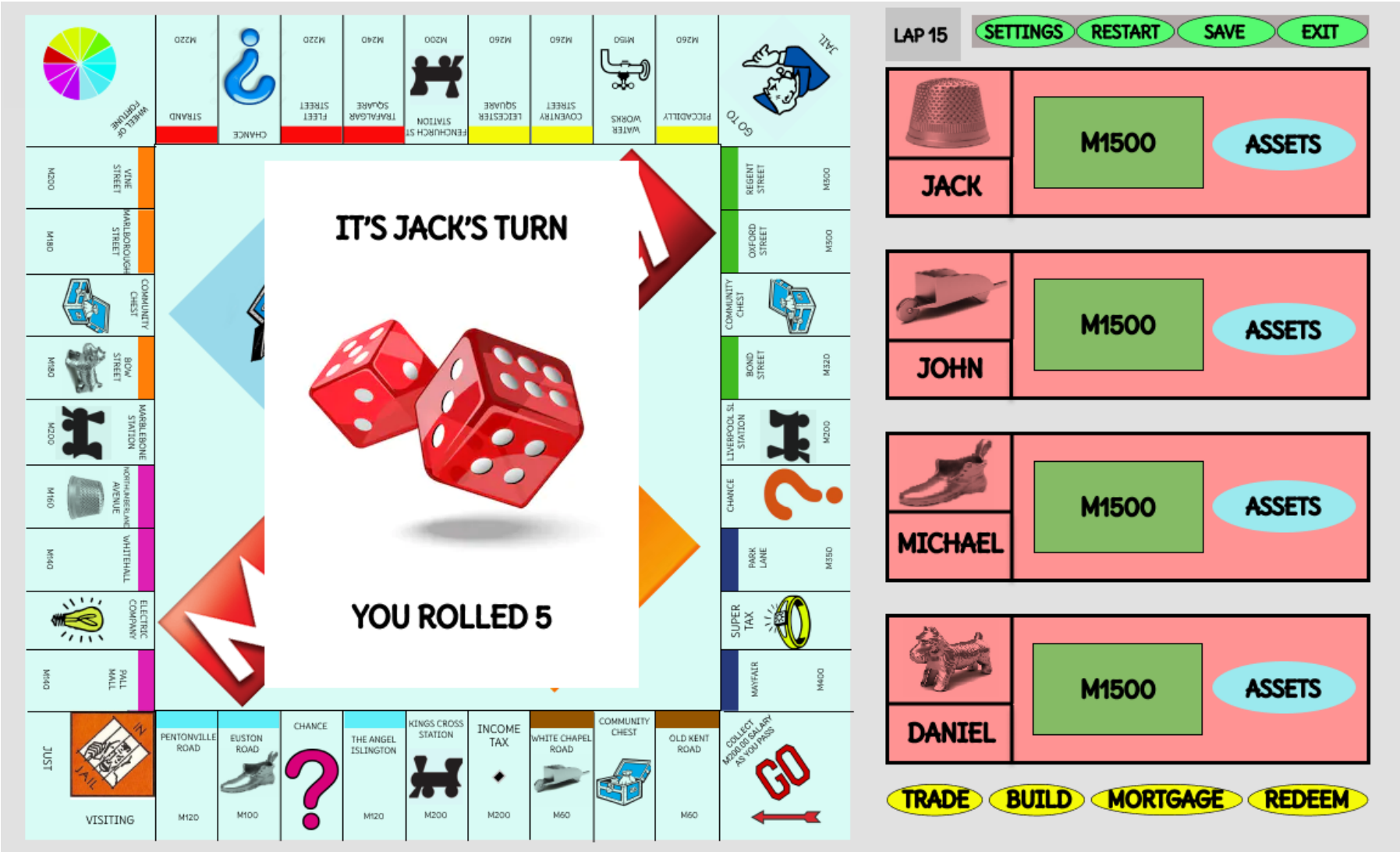


Figure 25. Gameplay roll dice and play screen.

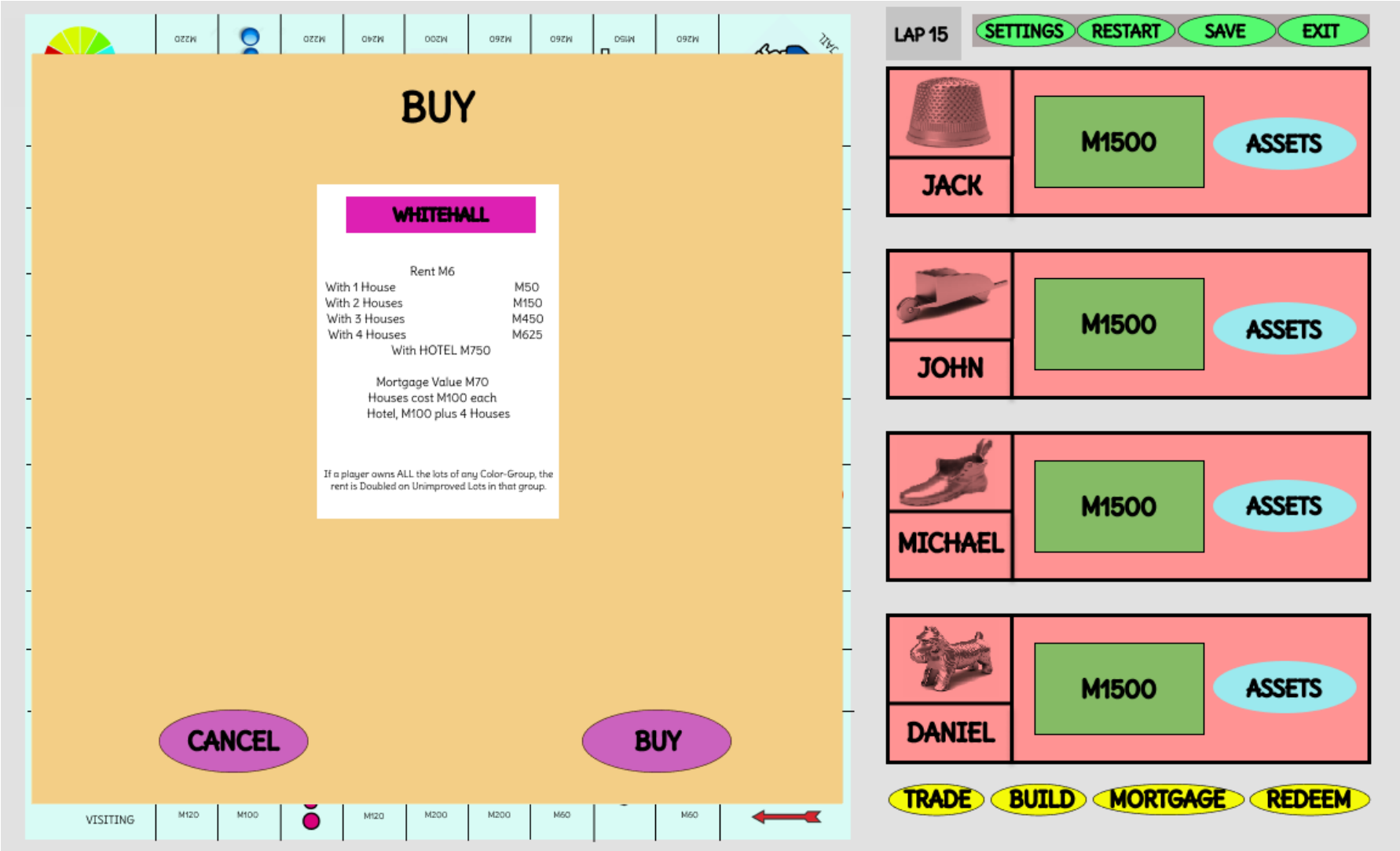


Figure 26. Gameplay buy screen.

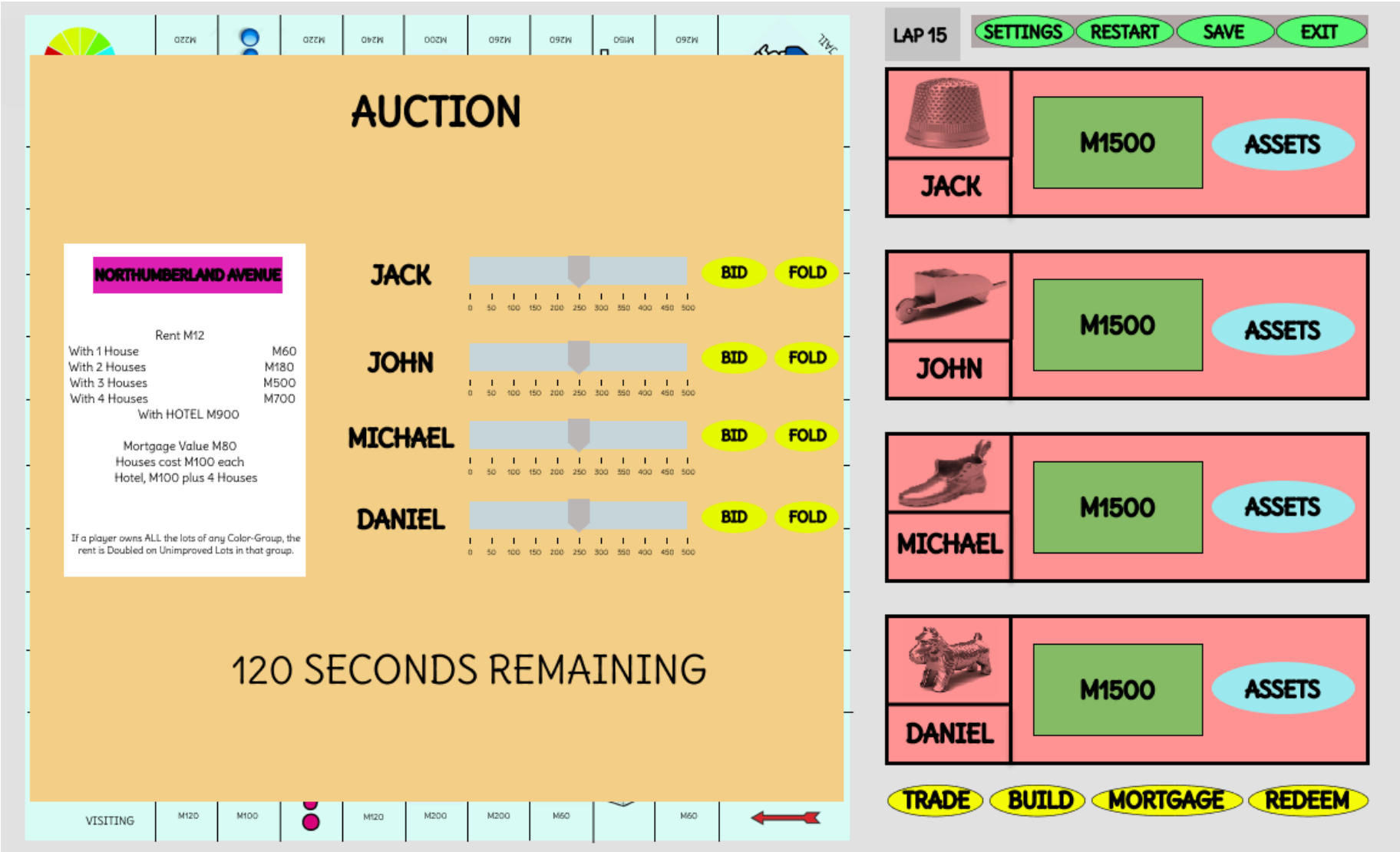


Figure 27. Gameplay auction screen.

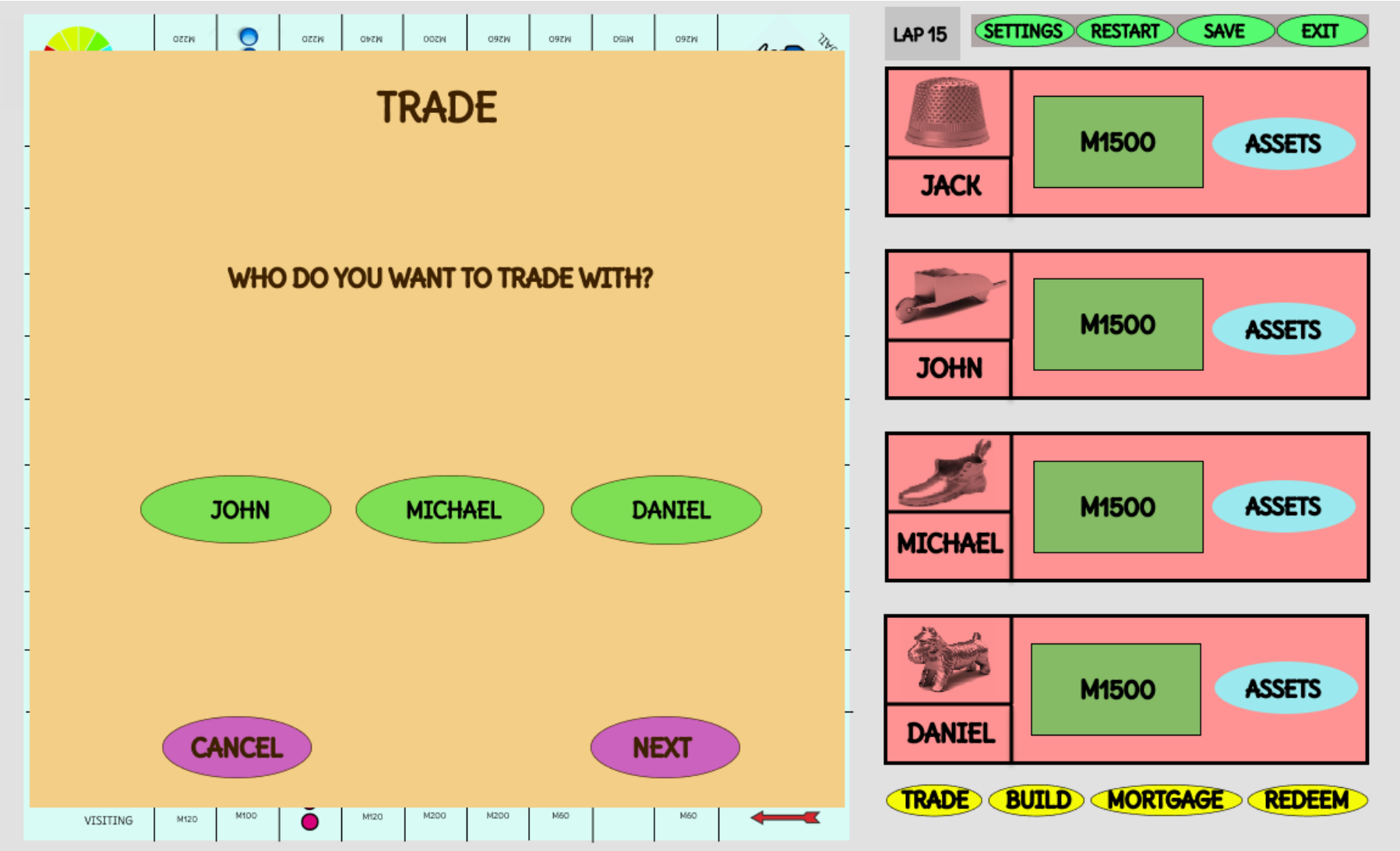


Figure 28. Gameplay trade player choosing screen.

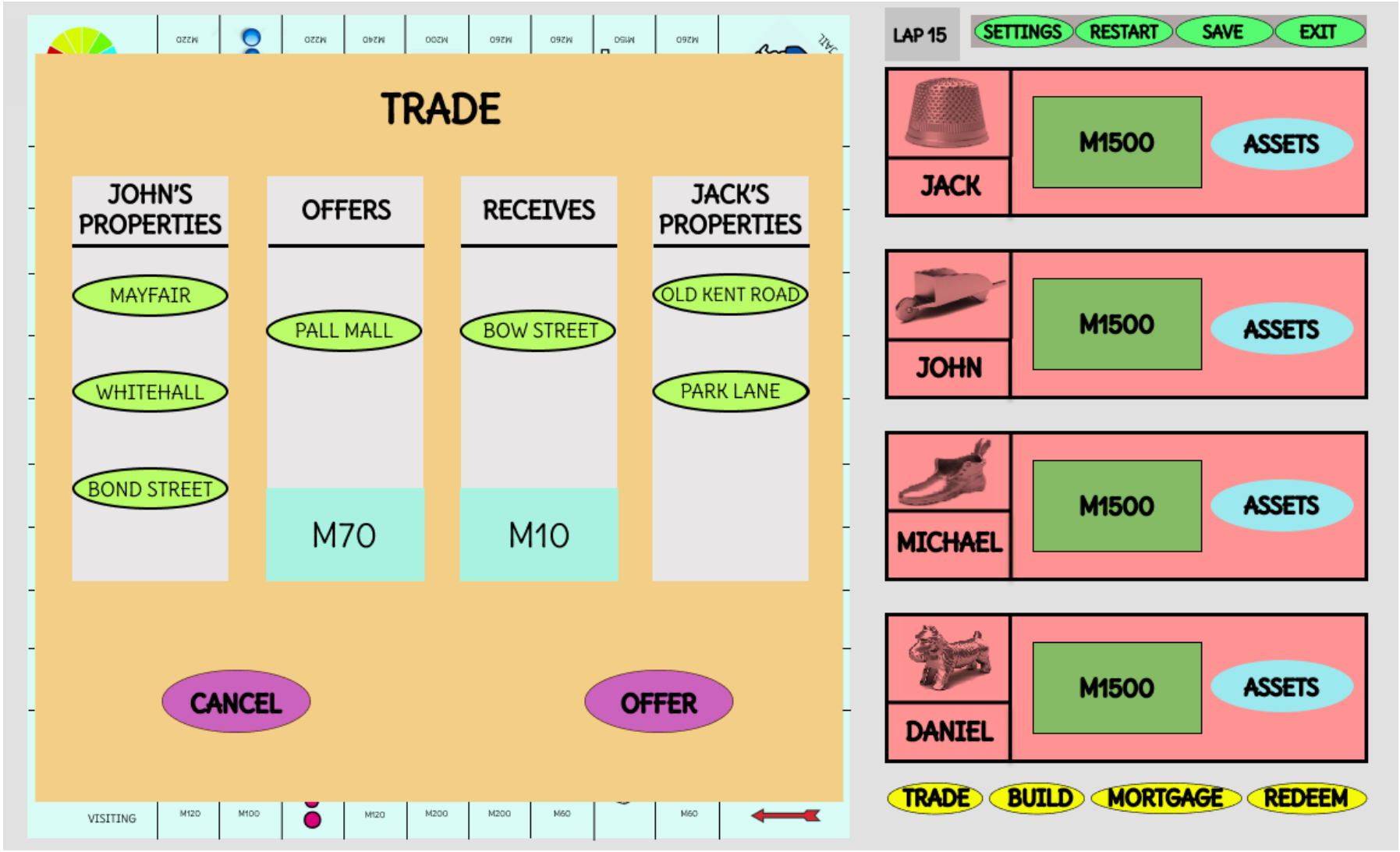


Figure 29. Gameplay trade offer screen.

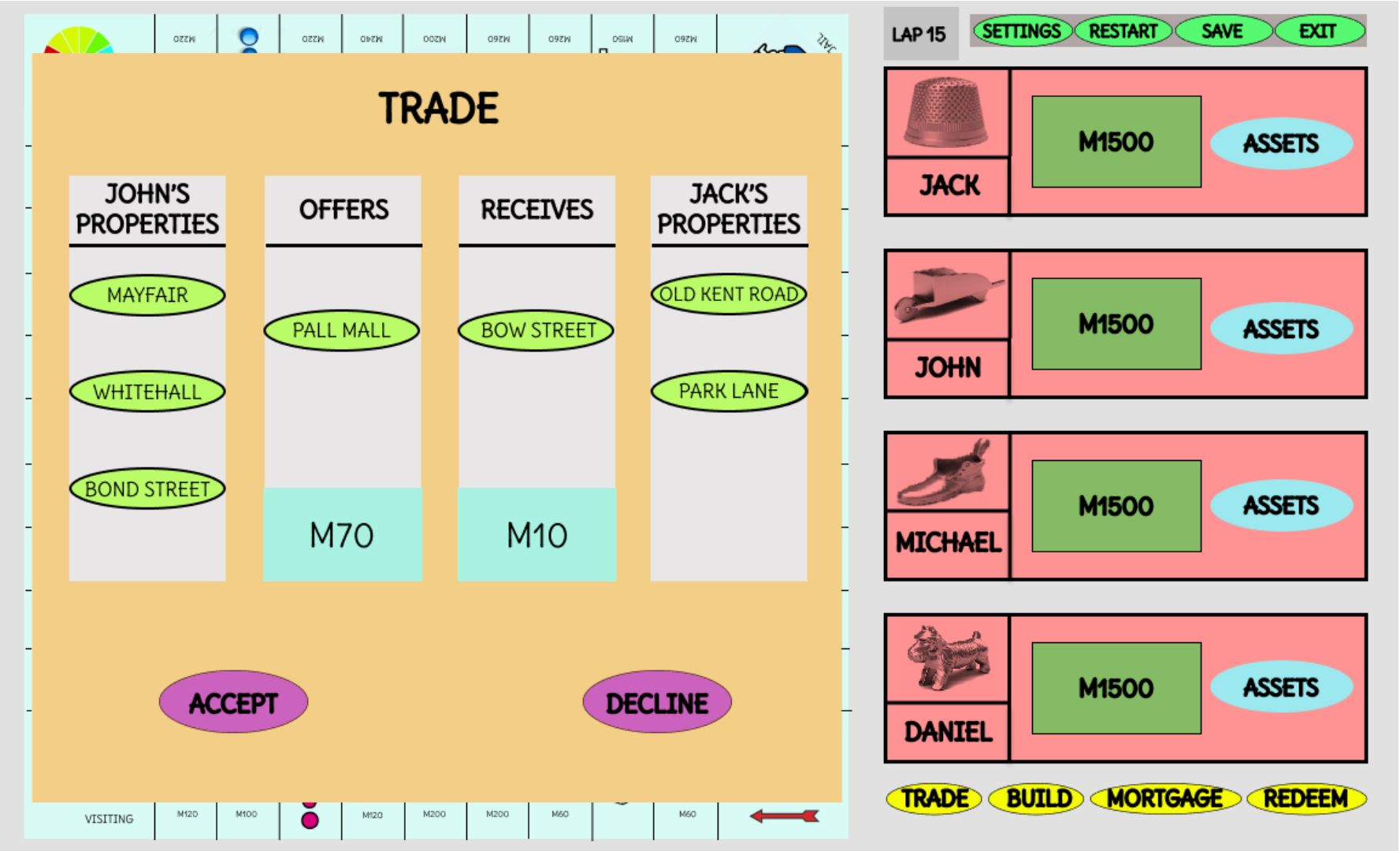


Figure 30. Gameplay trade approval screen.

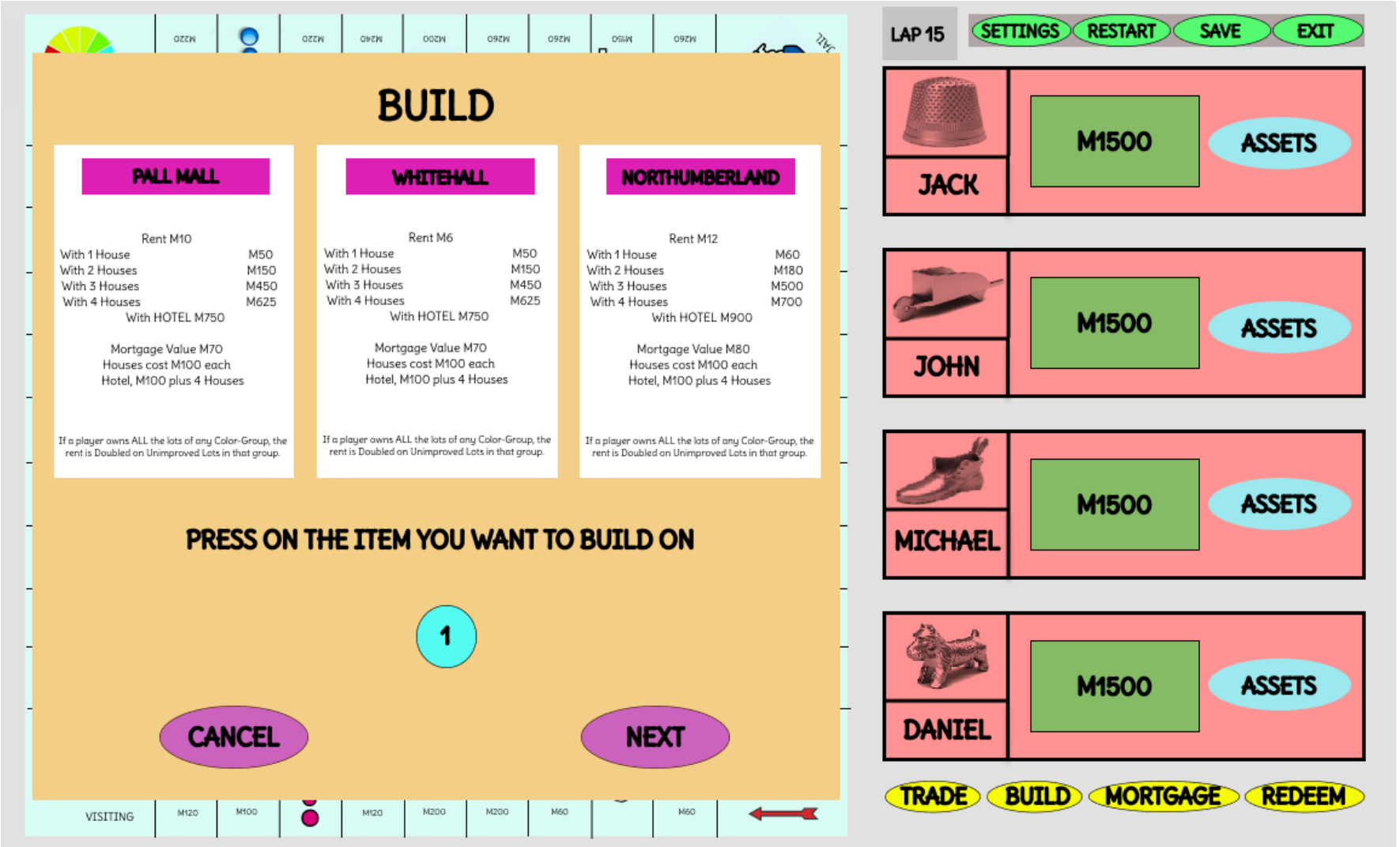


Figure 31. Gameplay build selection screen.

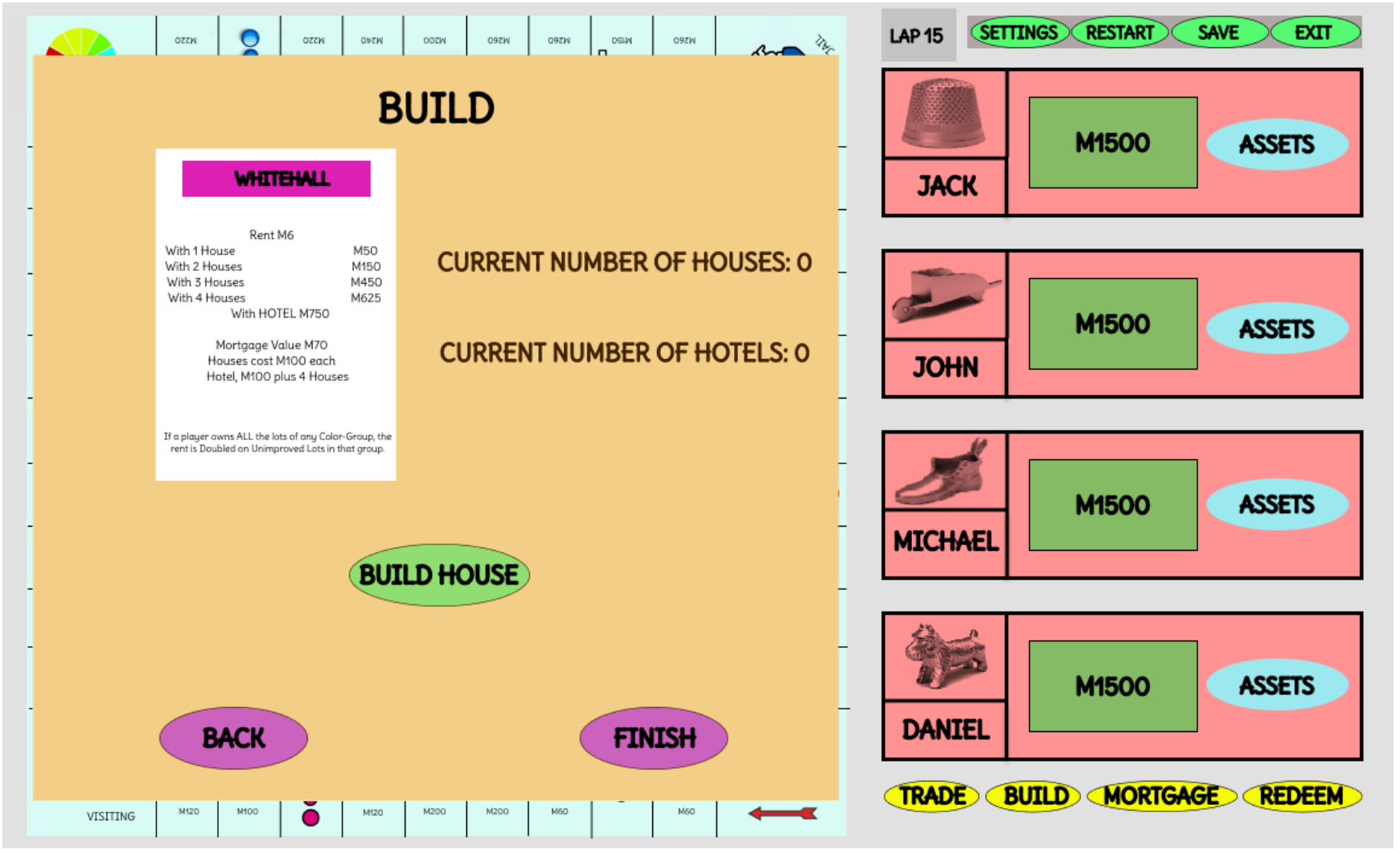


Figure 32. Gameplay build approval screen.

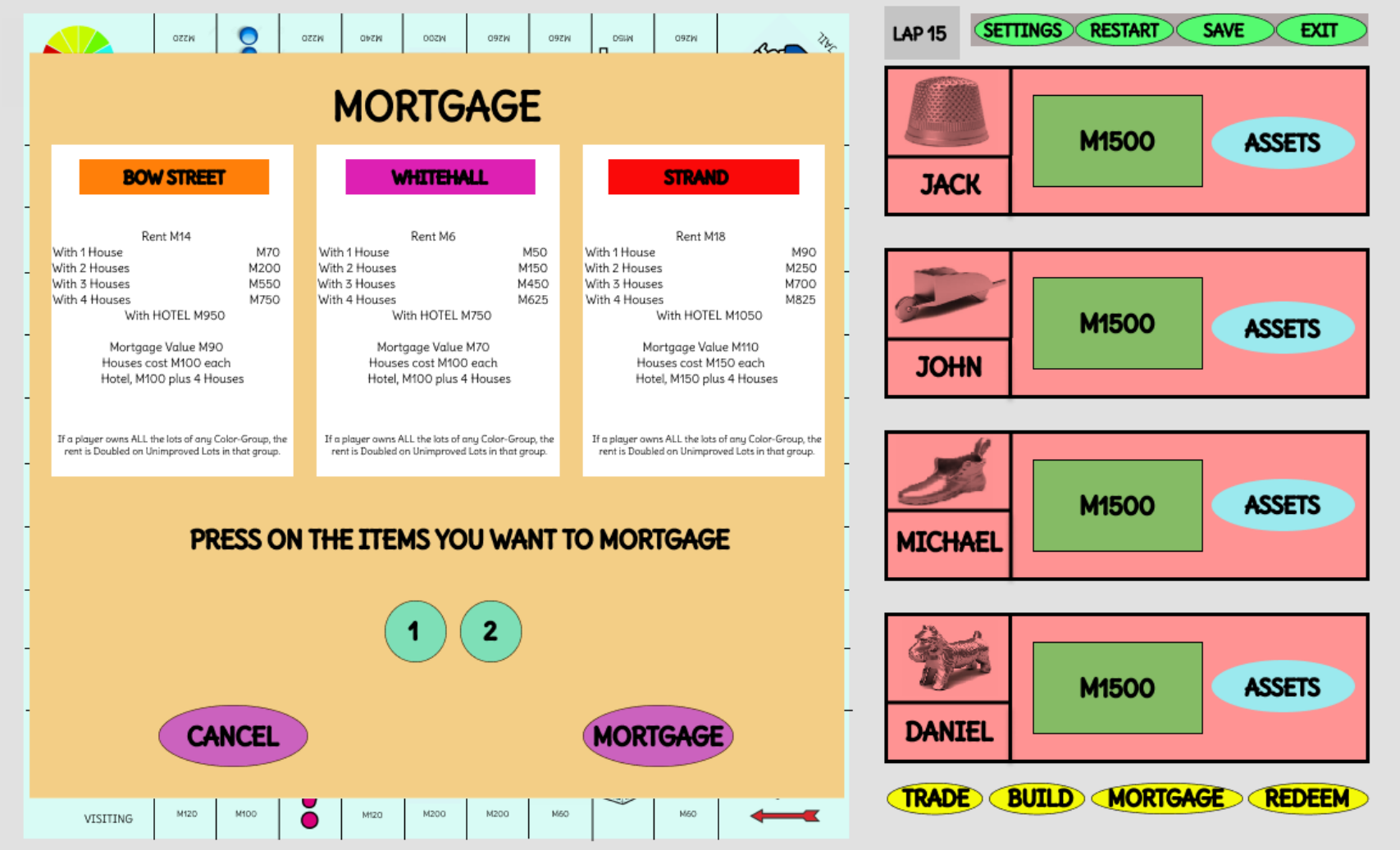


Figure 33. Gameplay mortgage screen.

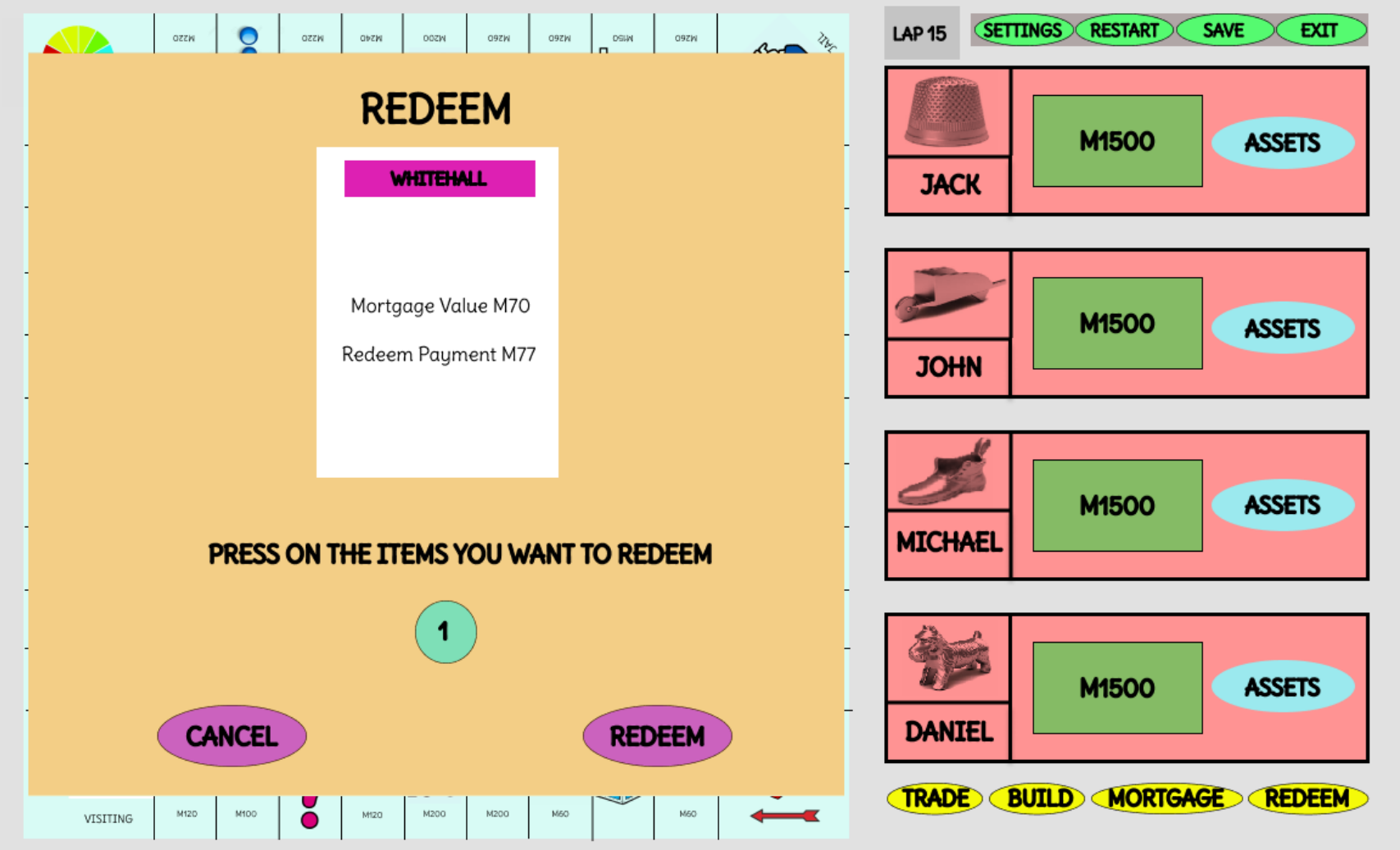


Figure 34. Gameplay redeem screen.

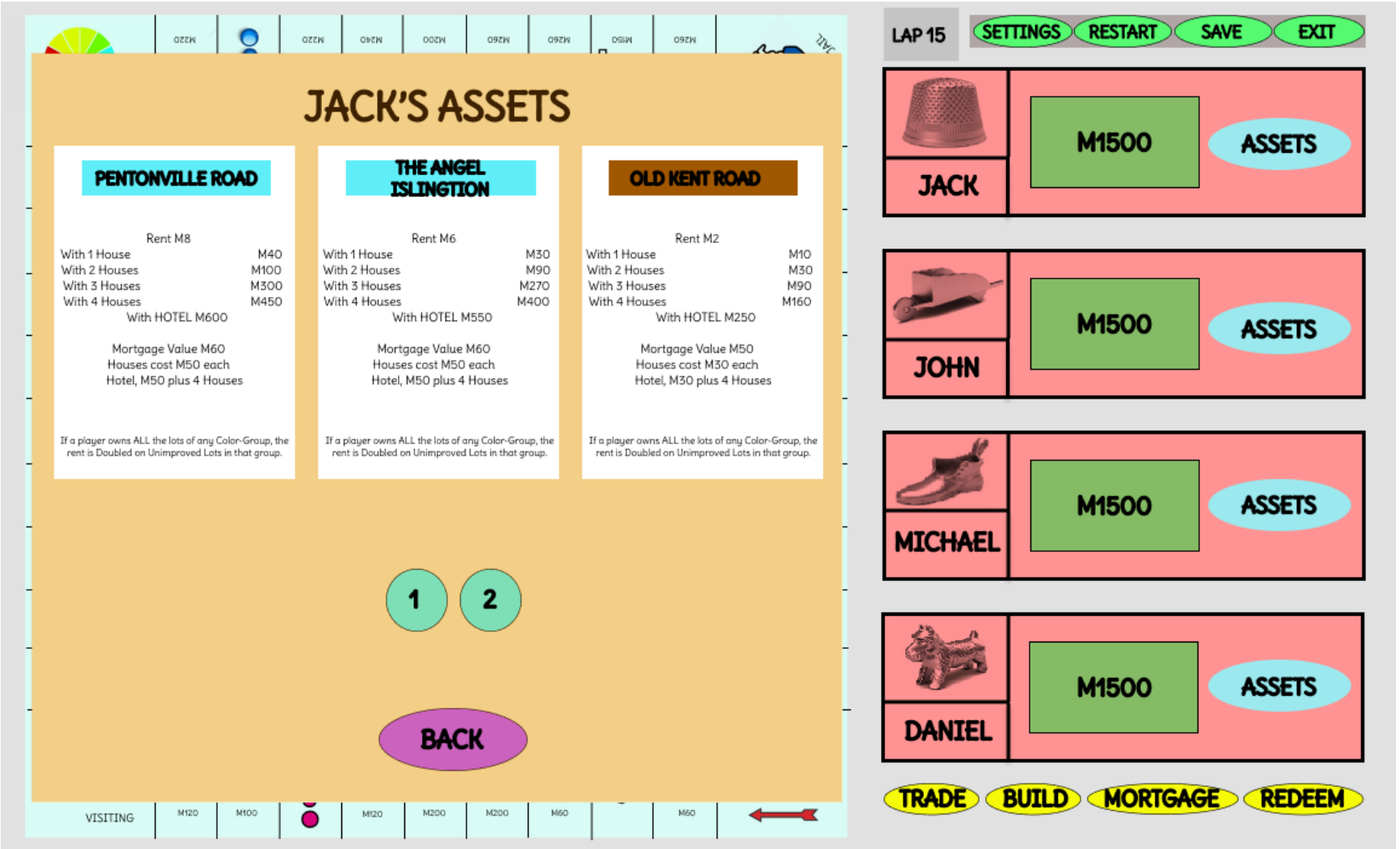


Figure 35. Gameplay screen for displaying the assets of a player.

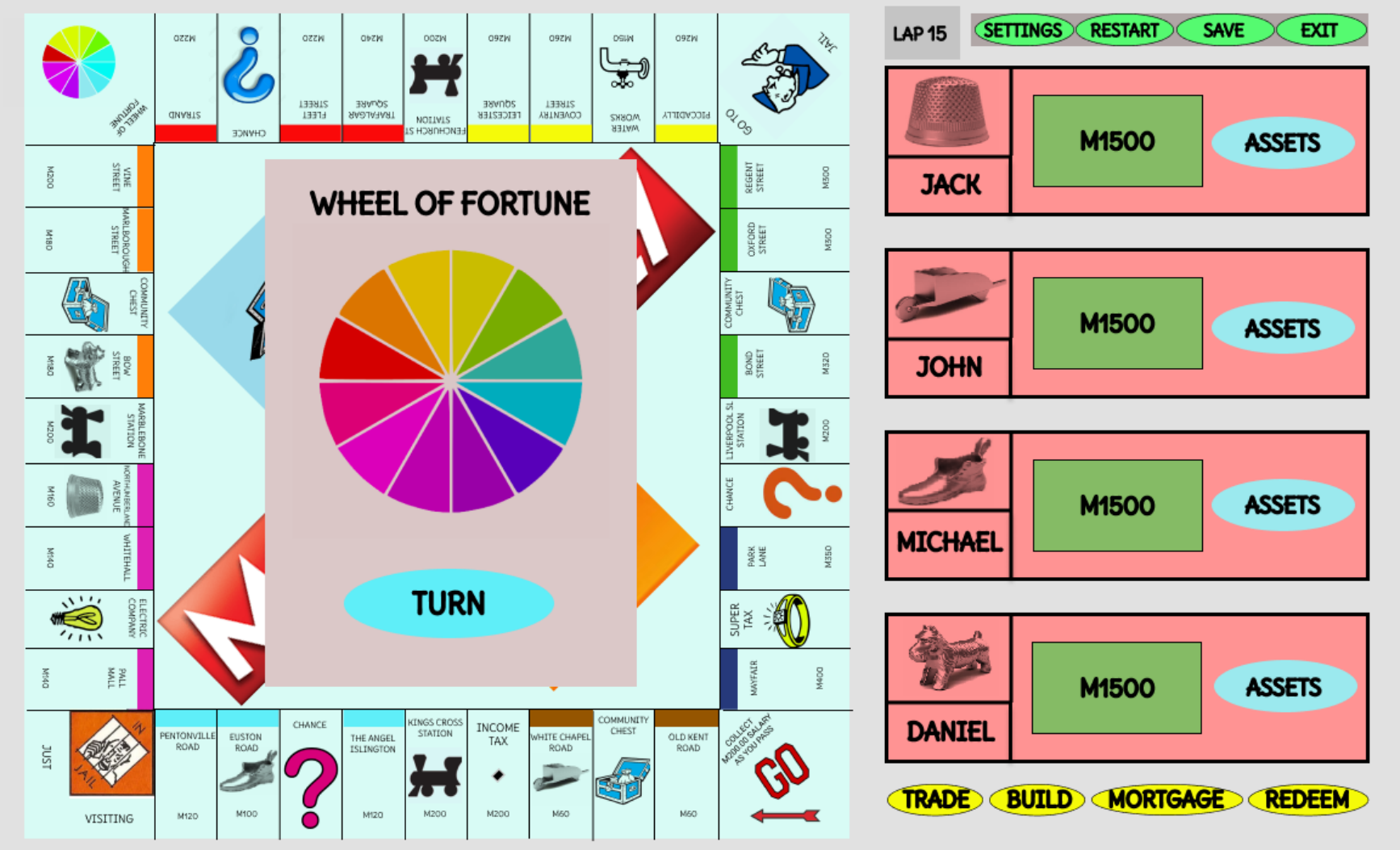


Figure 36. Gameplay wheel of fortune screen.


Figure 37. Monopoly game credits screen.

# 7 Bibliography

[1] *Object-Oriented Software Engineering, Using UML, Patterns, and Java, 3rd Edition*, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2010, ISBN-10: 0136066836.

[2] "Monopoly Rules." Hasbro, *www.hasbro.com/common/instruct/00009.pdf*. Accessed on September 29, 2020.

[3] Monopoly | Ubisoft (US), *www.ubisoft.com*, Accessed on September 29, 2020.

[4] IntelliJ IDEA Integrated Development Environment, *https://www.jetbrains.com/idea/*, Accessed on September 29, 2020.