System Report for M0N0P0LY

Brad Curtis

Bryan Rogers

Katherine Wilsdon

Dr. Isaac Griffith

Idaho State University

April 25, 2019

System Report for M0N0P0LY

Executive Summary

Team IF-Beta has produced a Monopoly-like game called *M0n0p0ly*. This game allows users to roll dice, advance around the board, and interact with the tile spaces that they land upon. The goal for players is to accumulate more money than any of their competitors, within the time limit of twenty turns, after which the player with the most money will be declared the winner. Our target audience is individuals ages 8 and up; particularly, small groups of family and friends. Within the scope of the class, Team IF-Beta's goal is to achieve the single group grade "A" that will be given out. The members of our team have split experience between the different aspects of planning, coding, and graphical user interface (GUI)-design which has allowed us create a well-rounded product.

The *Requirements Definition* section of this document details the user stories we established before project construction, how our group met these objectives, and the feature list of our resulting product. The *Functional Models* section lays out the relationships between different actions during the game. The *Structural Model* section depicts the desired functionality of the project that has been implemented in the code. The *Behavioral Models* section describes the change of state of certain objects within the game and actions occurring within the game, from a behavioral instead of functional focus. The *Appendix* details different parts of the GUI.

Some risks that we faced during the course of the project were: narrowing down our feature list to contain achievable, realistic goals and adding features that wouldn't cause scope creep. We worked on the project during our own time with a four month deadline, so that activity scheduling and maintaining a workable minimum viable product (MVP) throughout the

development phase were essential. These objectives were also key in our compliance with the key aspect of the Agile software development approach: iteration. This project will benefit our group members by creating a system that will serve as a template furthering each of our future careers.

Requirements Definition

Prior to starting the implementation of the project, we gathered some user stories to help us figure out what features to put into the game, as listed below in Figure 1.

User Stories	Implementation
As a user, I want to have a game so that I can have fun.	● MVP maintained throughout the project to minimize the risk of project-failure
As a user, I want to select a token to represent my character in the game so I can be easily identified.	● Custom player creation ● Game piece selection
As a user, I want to move around the board so that I can interact with the different tiles.	● Piece moving ● Board tile interactions ○ Properties ○ Rent/Fees ○ Passing Go ○ Going to Jail ○ Community/Chance cards ○ Income Tax/Luxury Tax
As a user, I want to be able to interact with the tiles I land on so that I can enjoy the rules of the game.	● Board tile interactions ○ Properties ○ Rent/Fees ○ Passing Go ○ Going to Jail ○ Community/Chance cards ○ Income Tax/Luxury Tax
As a user, I want this game to be automated so that I only have to enjoy playing and not paying attention to the tedious details.	As opposed to the physical version of Monopoly, M0n0p0ly has the game logic implemented behind-the-scenes so the players do not have to deal with it.
As a developer, I want the user to enjoy the game so they will play it often and have others play as well.	We created a nice project with numerous features that users are sure to enjoy. ● User friendly GUI

As a developer, I want to create this game so that the user does not have to remember the rules, but can just play.	As opposed to the physical version of Monopoly, M0n0p0ly has the game logic implemented behind-the-scenes so the players do not have to deal with it.
As a developer, I want to add features so that the user can have an even better experience than just using the basic MVP.	● 2-4 players ● Save/Load game functionality ● Community and Chance cards ● Income Tax/Luxury Tax

Figure 1: User stories and their resulting implementations in M0n0p0ly.

Feature List

Additionally, we have laid out the features that our MVP and final product had in Figure 2. This gives a more complete view of the game's features, some of which may not have appeared as user stories (or were not specifically mentioned).

MVP	● 2 players ● Visual of gameboard (just the background) ● Main game logic implemented ● 20-turn win condition with victory screen ● Dice Rolling ● 2 icons; 1 for each player; no customizability ● Player pieces move around the board ● Player details display ● Board tile interactions ○ Properties (purchase and pay rent): basic properties, railroads, and utilities ○ Go (and passing Go) ○ Jail ○ Go to Jail ○ Free Parking
Final Product	● MVP features ● 2-4 players ● Title Screen ● Save/Load Game to/from file ● Player customization (custom names and can choose from selection of 8 game pieces) ● Gameboard has tile names and purchase/rent costs displayed ● Board tile interactions (in addition to MVP) ○ Community and Chance Cards ○ Luxury/Income Tax ● Community and Chance cards can be customized by editing their .txt files ● (Internal: Singleton implemented instead of static class)

Figure 2: Full feature list for the MVP and final versions of M0n0p0ly.

Functional Models

Each player in M0n0p0ly will take turns in a set order, and during the course of said turns, will roll dice, get their piece moved to the new location, and perform some action based on that new location. Alternatively, the player can go to jail by rolling 3 doubles in a row, or by being sent there by some special means (landing on the Go to Jail space or drawing a Community/Chance card that sends them to jail). If a player is in Jail, they must either escape by rolling doubles, or wait until three turns have passed. Finally, once 20 turns have passed for all players, the game ends. This is detailed below, visually in Figure 3. Note that the act of taking a turn is meant to be one turn of one player.

We chose not to implement Get-out-of-jail-free cards or payment to escape Jail due to the complexity involved in implementation (particularly in the GUI) and the lack of time needed to complete this feature. Additionally, for much the same reason, we did not implement houses or hotels and their associated community chest and chance cards, which normally have the effect of ending a game quickly by vastly increasing rent prices. To counteract this, we instead have a victory condition of whoever has the most money after 20 turns instead of waiting for all but one player to go bankrupt.

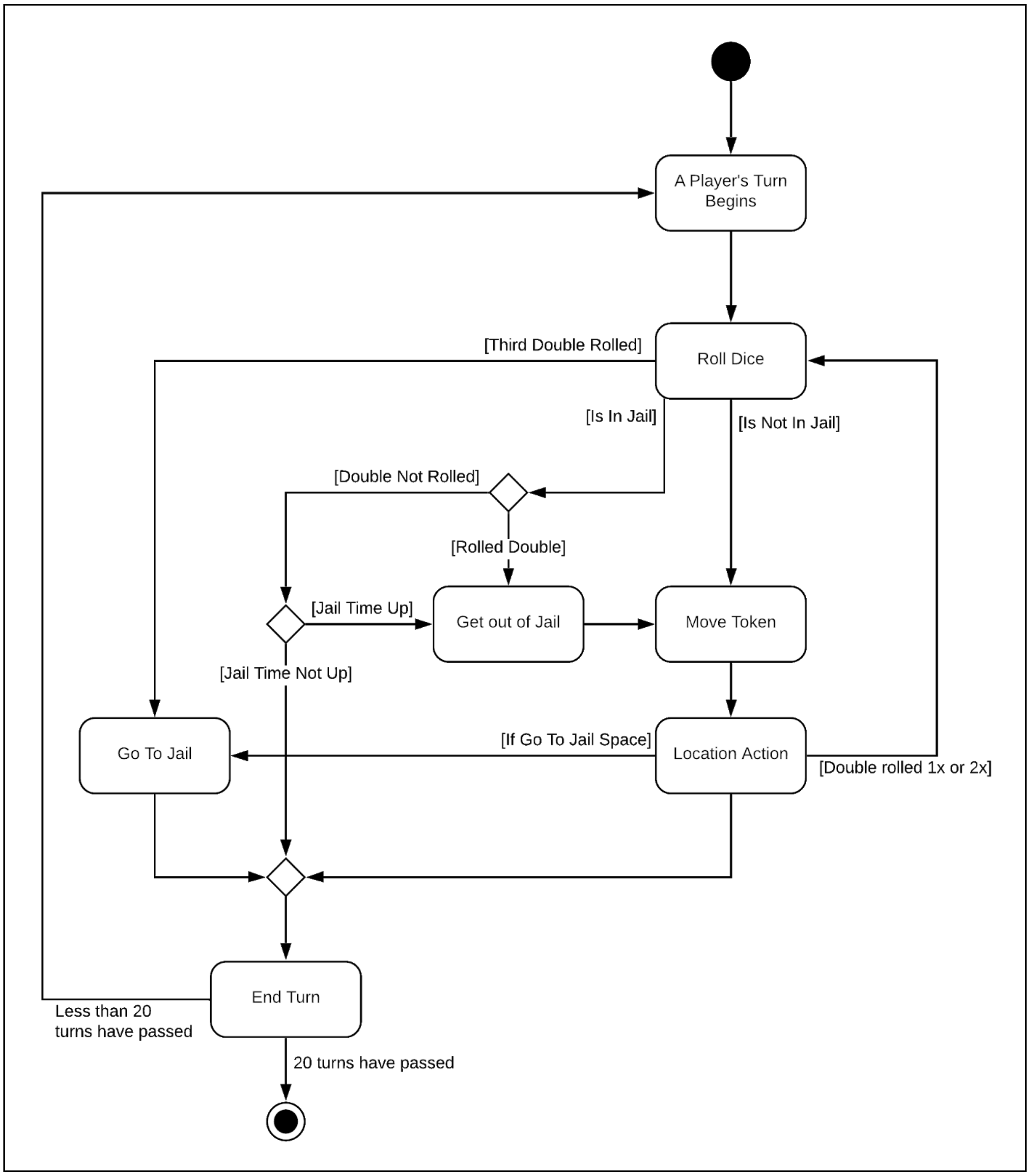


Figure 3: Activity diagram for taking a turn.

In Figure 4, a player will roll dice and move their piece to a community chest or chance tile. A card will be randomly picked and some action will be performed to implement the card. The card may instruct the player to collect money, pay money, or advance to a new location. For collect and pay money, the player may be instructed to make a one time payment to or from the bank or the player can collect from or pay every player a certain amount of money. Since the houses and hotels feature was not implemented, these community and chance cards were discarded.

The other option is to advance to a property, the go to jail tile, the income tax tile, or another community chest or chance space. Property was included because certain community chest and chance cards directed the player to a basic, utility, or railroad property. A chance card specifies for the player to move back 3 spaces, so property, the go to jail tile, the income tax tile, or another community chest or chance space were included. If the player lands on a property, the player either pays rent to the owner or can purchase the property. The player is sent to jail when landing on the go to jail tile. The player pays 10% of their money or $200 (whichever is greater) when landing on the Income Tax Space. If a player moves back 3 spaces to another community chest or chance space, another card is randomly drawn and the process repeats.

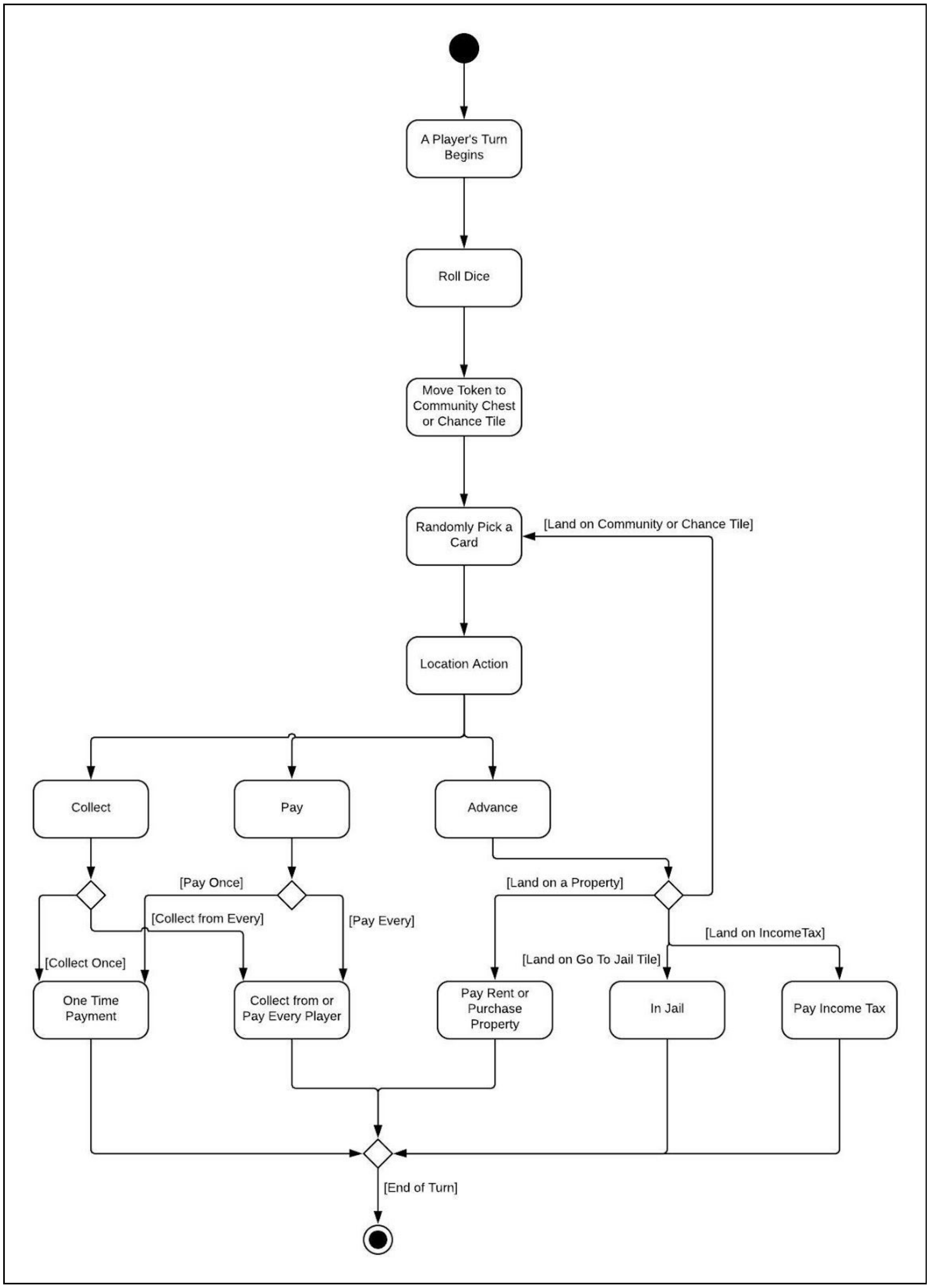
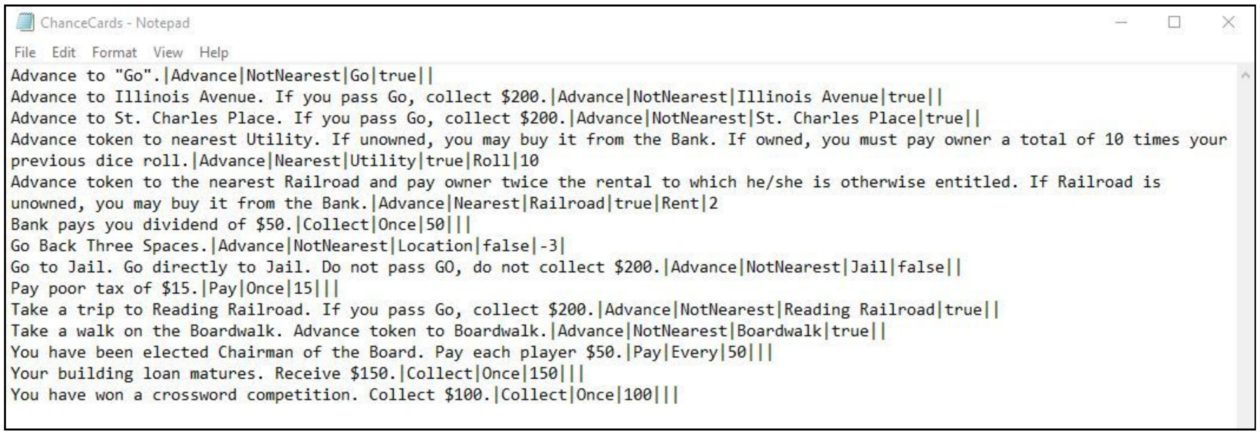


Figure 4: Activity diagram for landing on a community chest or chance tile when taking a turn.

A major design decision was to create a text file for each group to process all the community chest and chance cards. The text file for community cards is shown in Figure 5. Each card is separated by pipes for processing. The first entry is the phrase. After the pipe, the next step is determine if the player will “Collect”, “Pay”, or “Advance”. If “Collect” or “Pay”, the next step is to determine if the player or bank makes a one time payment named “Once” or if the player collects or pays every player in the game an allotted amount of money named “Every”. The amount to pay or collect follows.

If “Advance”, the next step is to determined if the new location is the “Nearest” or a specific location named “NotNearest”. The subsequent step in the text file is the new location. The next processing step is whether the player can collect $200 when passing go or not (true or false). If the player is directed to the nearest utility, he or she must pay 10 times their dice roll shown by “Roll” followed by “10”. If the player is directed to the nearest railroad, he or she must pay double the rent shown by “Rent” followed by “2”. One of the card specifies for the player to move back 3 spaces. So, the new location step says “Location” followed by false for not collecting $200 when passing go, and then a “-3” to symbolize going back three spaces.



```
Advance to "Go".|Advance|NotNearest|Go|true||
Advance to Illinois Avenue. If you pass Go, collect $200.|Advance|NotNearest|Illinois Avenue|true||
Advance to St. Charles Place. If you pass Go, collect $200.|Advance|NotNearest|St. Charles Place|true||
Advance token to nearest Utility. If unowned, you may buy it from the Bank. If owned, you must pay owner a total of 10 times your
previous dice roll.|Advance|Nearest|Utility|true|Roll|10
Advance token to the nearest Railroad and pay owner twice the rental to which he/she is otherwise entitled. If Railroad is
unowned, you may buy it from the Bank.|Advance|Nearest|Railroad|true|Rent|2
Bank pays you dividend of $50.|Collect|Once|50|||
Go Back Three Spaces.|Advance|NotNearest|Location|false|-3|
Go to Jail. Go directly to Jail. Do not pass GO, do not collect $200.|Advance|NotNearest|Jail|false||
Pay poor tax of $15.|Pay|Once|15|||
Take a trip to Reading Railroad. If you pass Go, collect $200.|Advance|NotNearest|Reading Railroad|true||
Take a walk on the Boardwalk. Advance token to Boardwalk.|Advance|NotNearest|Boardwalk|true||
You have been elected Chairman of the Board. Pay each player $50.|Pay|Every|50|||
Your building loan matures. Receive $150.|Collect|Once|150|||
You have won a crossword competition. Collect $100.|Collect|Once|100|||
```

Figure 5: Chance card text file.

Figure 6 shows some of the actions a player can make during his or her turn. A player can roll dice and move to a tile. A player has the opportunity to purchase a property, or must pay rent to the owner or fees such as income tax, luxury tax, or community chest / chance card fees. Additionally, the player will win the game after 20 turns if he or she has the most money at that time.

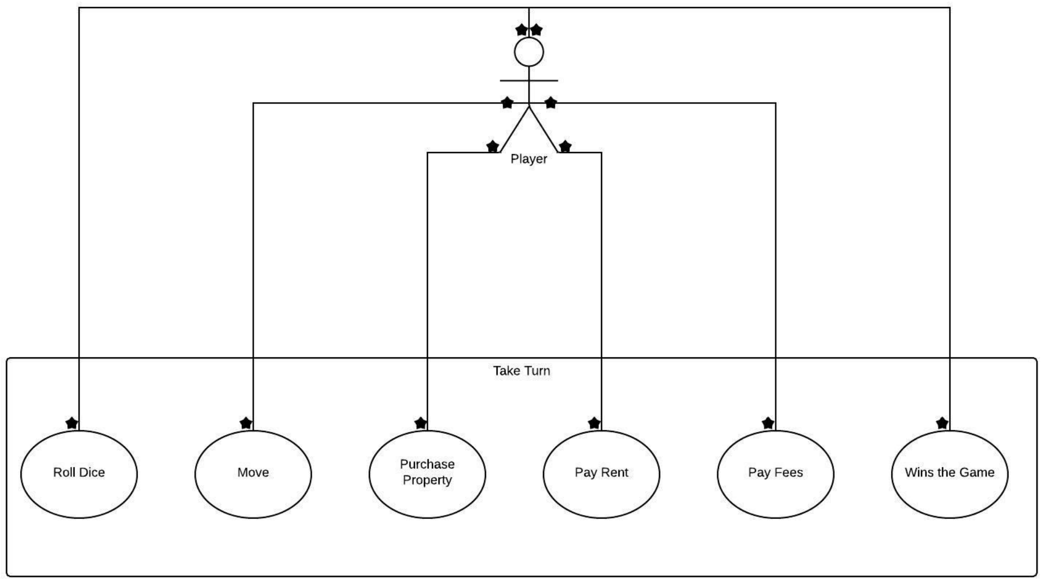
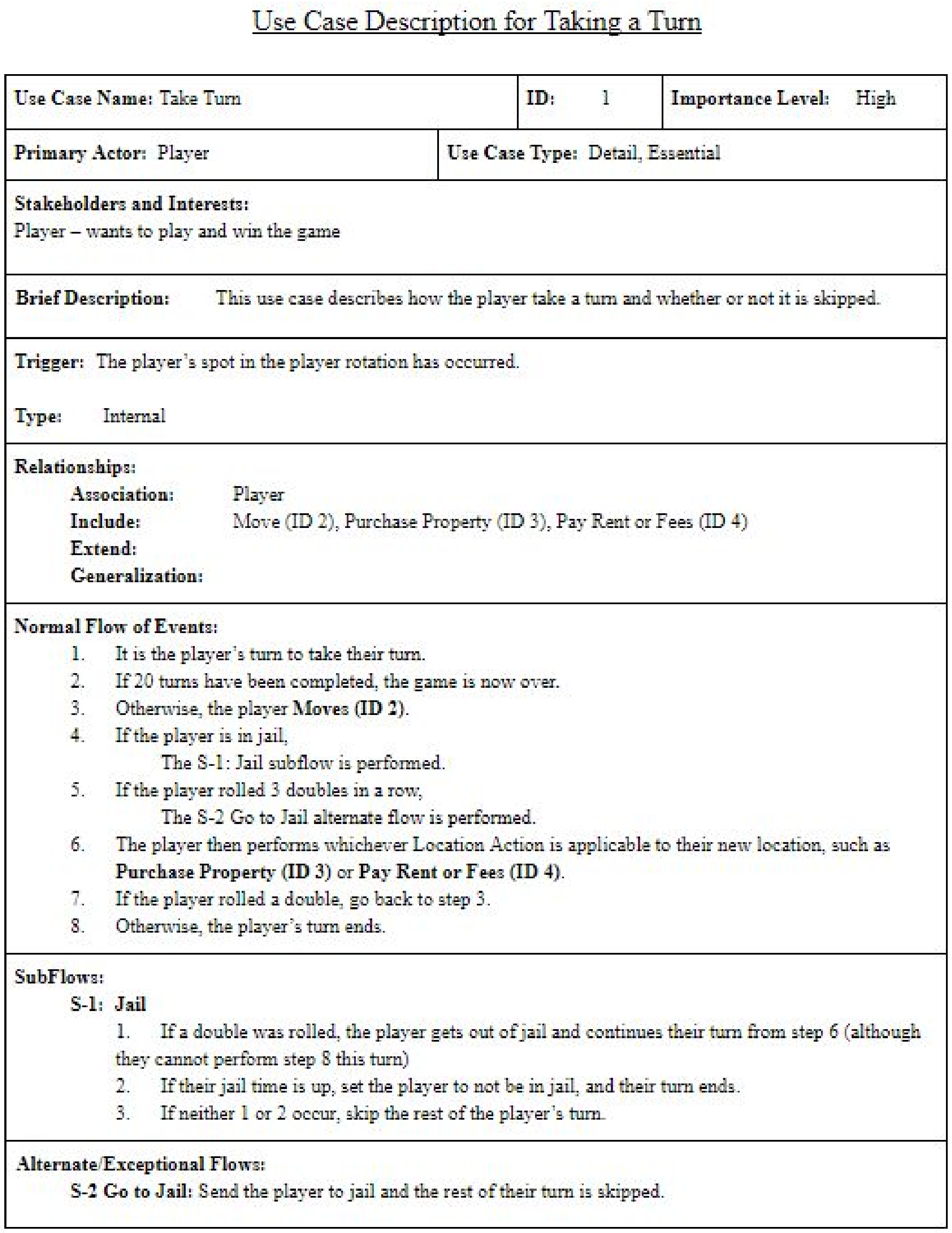


Figure 6: Use Case for Taking a Turn.

In Figure 7, the use case describes the sequence of events for taking a turn. We rated this use case as a high level of importance because it controls the main component of the game. The Take a Turn use case is associated with player since the player interacts with this use case. We decided that Move (ID 2), Purchase Property (ID 3), and Pay Rent or Fees (ID 4) have an include relationship because Take a Turn includes the functionality of these use cases. For the flow and sub-flow of events of Take a Turn, see the subsections in Figure 7.



Use Case Description for Taking a Turn

Use Case Name: Take Turn	**ID:** 1	**Importance Level:** High
Primary Actor: Player	**Use Case Type:** Detail, Essential	

Stakeholders and Interests:
Player – wants to play and win the game

Brief Description: This use case describes how the player take a turn and whether or not it is skipped.

Trigger: The player's spot in the player rotation has occurred.

Type: Internal

Relationships:

Association:	Player
Include:	Move (ID 2), Purchase Property (ID 3), Pay Rent or Fees (ID 4)
Extend:	
Generalization:	

Normal Flow of Events:

1. It is the player's turn to take their turn.
2. If 20 turns have been completed, the game is now over.
3. Otherwise, the player **Moves (ID 2)**.
4. If the player is in jail,
   The S-1: Jail subflow is performed.
5. If the player rolled 3 doubles in a row,
   The S-2 Go to Jail alternate flow is performed.
6. The player then performs whichever Location Action is applicable to their new location, such as **Purchase Property (ID 3)** or **Pay Rent or Fees (ID 4)**.
7. If the player rolled a double, go back to step 3.
8. Otherwise, the player's turn ends.

SubFlows:

S-1: Jail

1. If a double was rolled, the player gets out of jail and continues their turn from step 6 (although they cannot perform step 8 this turn)
2. If their jail time is up, set the player to not be in jail, and their turn ends.
3. If neither 1 or 2 occur, skip the rest of the player's turn.

Alternate/Exceptional Flows:

S-2 Go to Jail: Send the player to jail and the rest of their turn is skipped.

Figure 7: Use case description for taking a turn.

As shown in Figure 8, a player can move to various tiles. He or she can land on pass go, free parking, property, jail (either in jail or just visiting), Go to Jail tile, community chest space, chance space. The group decided that the move method includes all the described tiles because move includes the functionality of the tile use cases. We incorporated income tax and luxury tax into the free parking class because in our MVP, these spaces were free parking. An income tax and luxury tax extends free parking because we made them options of the free parking.

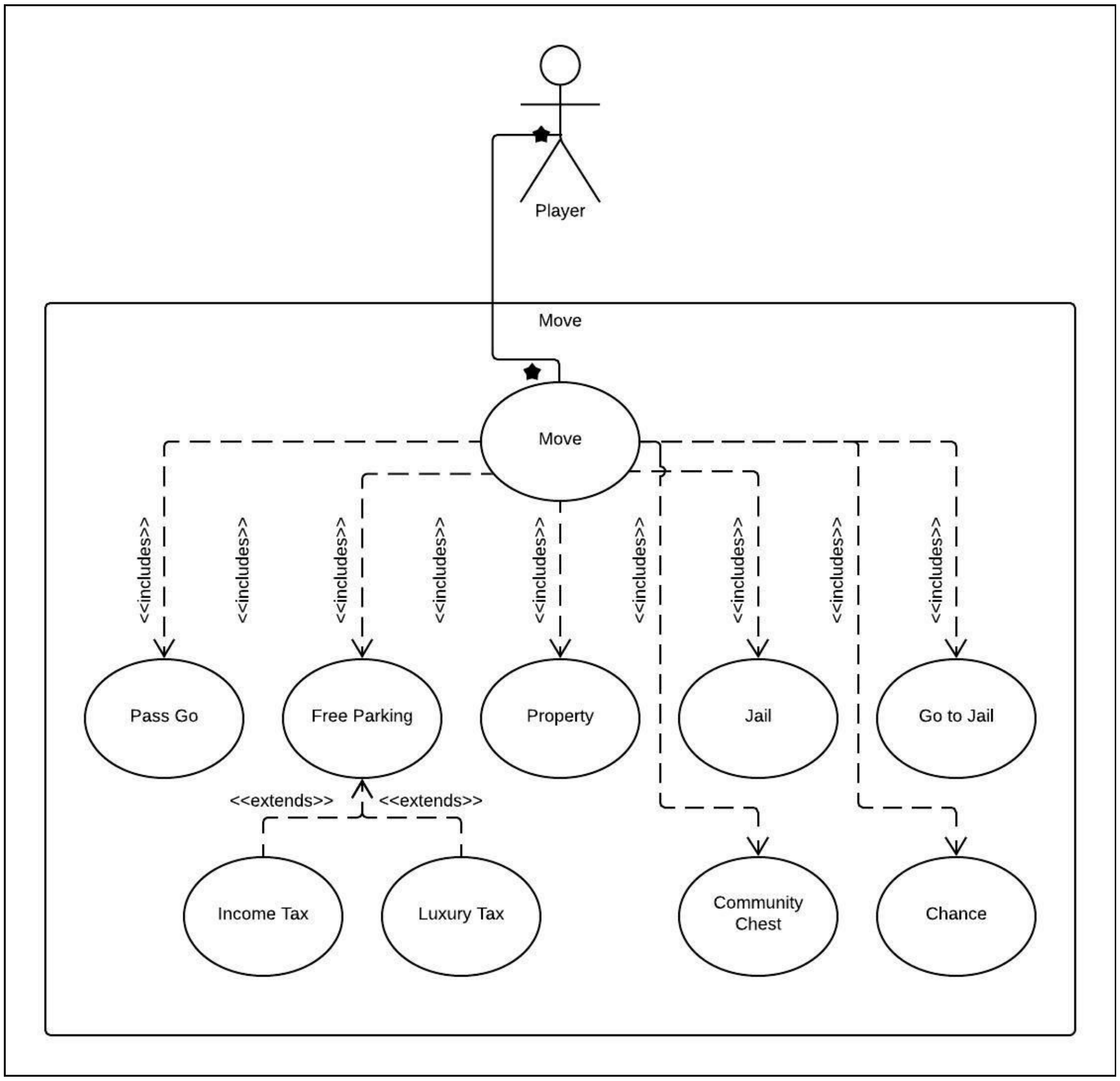
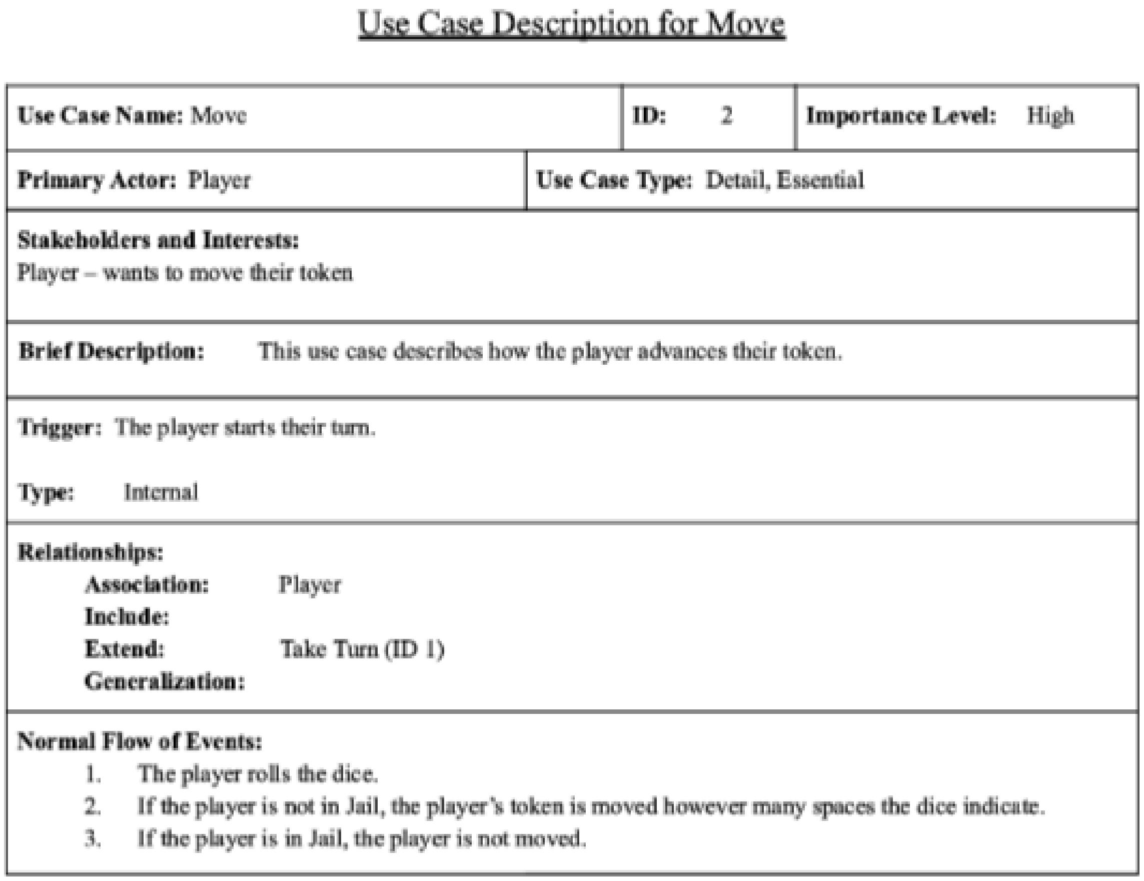


Figure 8: Use case for moving.

Figure 9 describes whether or not the player can move depending on their *In Jail* state. The Move use case is associated with player since the use case relocates the player to a new location. Move extends Take Turn (ID 1) because Move is an optional behavior that extends functionality to Take a Turn.For the flow of events of Move, see the subsection in Figure 9.



Use Case Description for Move

Use Case Name: Move	**ID:** 2	**Importance Level:** High
Primary Actor: Player	**Use Case Type:** Detail, Essential	

Stakeholders and Interests:
Player – wants to move their token

Brief Description: This use case describes how the player advances their token.

Trigger: The player starts their turn.

Type: Internal

Relationships:

Association:	Player
Include:	
Extend:	Take Turn (ID 1)
Generalization:	

Normal Flow of Events:
1. The player rolls the dice.
2. If the player is not in Jail, the player's token is moved however many spaces the dice indicate.
3. If the player is in Jail, the player is not moved.

Figure 9: Use case description for moving.

Figure 10 describes a player purchasing a property. A player can purchase a basic, railroad, or utility property. If a player cannot afford to purchase a property, the game will notify the player and make the Buy button disabled.

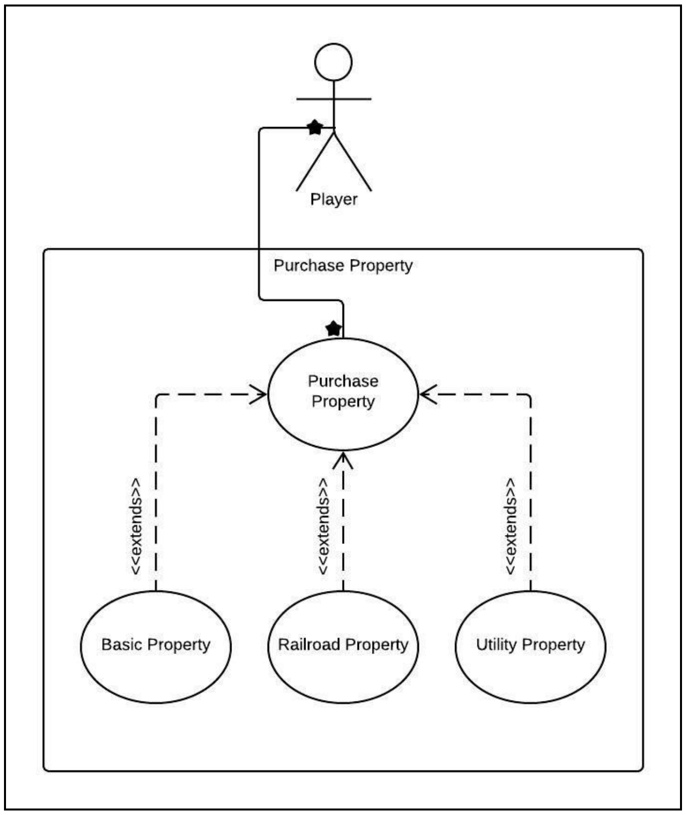
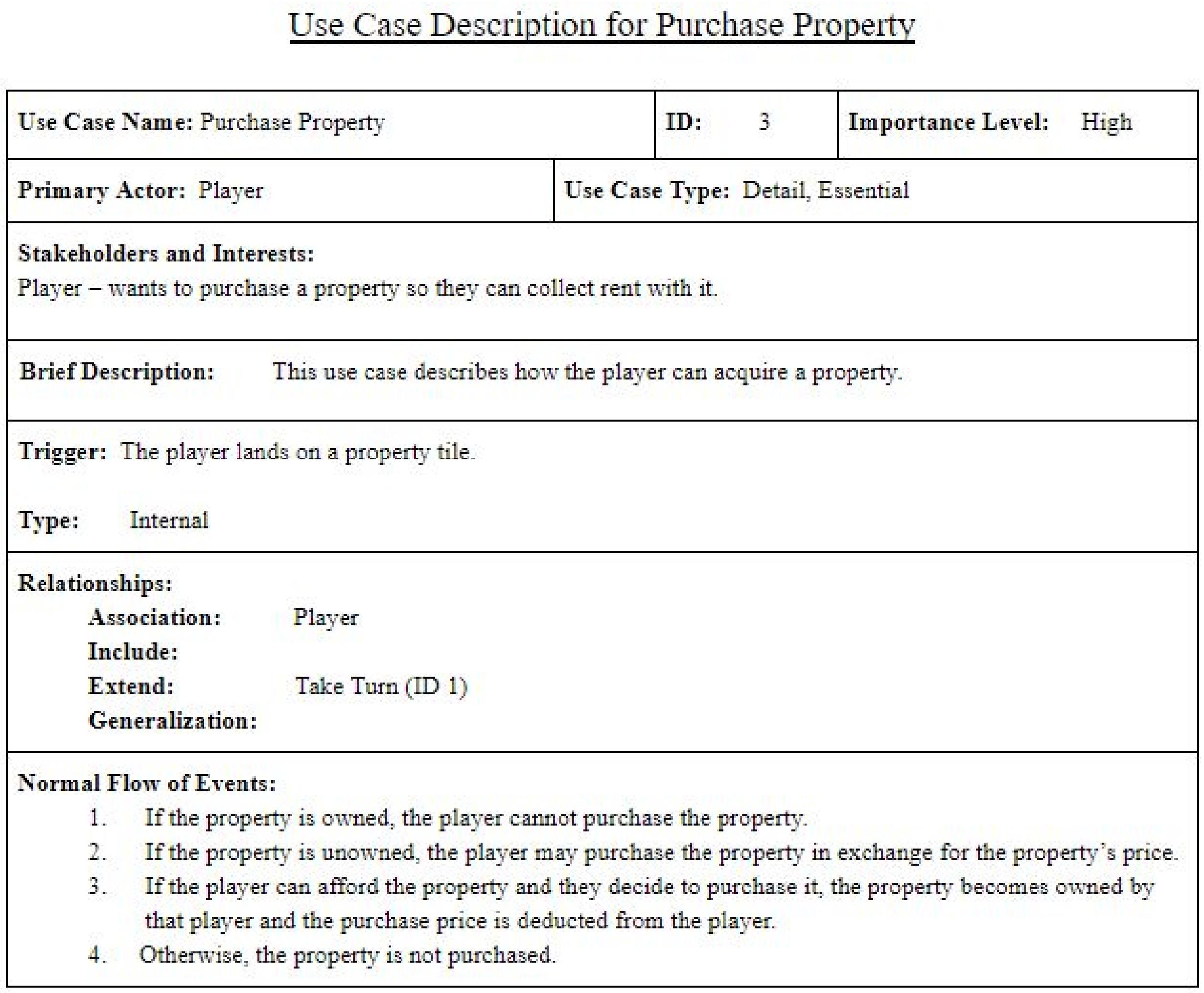


Figure 10: Use case for purchasing a property.

In Figure 11, the Purchase Property use case shows whether a player can purchase a property. The Purchase Property use case is associated with player since the use case determines if the player can buy a property or not . Purchase Property extends Take Turn (ID 1) because purchasing a property is an optional behavior depending on if the player moves to a new location that can be purchased. So, Purchase Property extends the functionality of Take a Turn. We decided that Purchase Property should not extend Move (ID 2) because once move finishes, the ability to buy a property begins while the player is still taking his or her turn. For the flow of events of Purchase Property, see the subsection in Figure 11.



Use Case Description for Purchase Property

Use Case Name: Purchase Property	**ID:** 3	**Importance Level:** High
Primary Actor: Player	**Use Case Type:** Detail, Essential	

Stakeholders and Interests:
Player – wants to purchase a property so they can collect rent with it.

Brief Description: This use case describes how the player can acquire a property.

Trigger: The player lands on a property tile.

Type: Internal

Relationships:

Association:	Player
Include:	
Extend:	Take Turn (ID 1)
Generalization:	

Normal Flow of Events:

1. If the property is owned, the player cannot purchase the property.
2. If the property is unowned, the player may purchase the property in exchange for the property's price.
3. If the player can afford the property and they decide to purchase it, the property becomes owned by that player and the purchase price is deducted from the player.
4. Otherwise, the property is not purchased.

Figure 11: Use case description for purchasing a property.

The player can pay rent or fees as shown in Figure 12. A player must pay rent to the owner when landing on a basic, railroad, or utility property. When the first railroad is purchased, the amount of rent is $25. For every subsequent railroad purchased, the rent doubles (i.e. $50, $100, $200). We wrote a recursive method to calculate the rent owed. If one utility is owned, rent is 4 times the dice roll. If both utilities are owned, rent is 10 times the dice roll. When the player has to pay fees, he or she lands on income tax, luxury tax, or a community chest or chance space. The income tax is when the player pays 10% of their money or $200 (whichever is greater). The luxury tax elicits the player to pay $100. This is also displayed in document form in Figure 13.

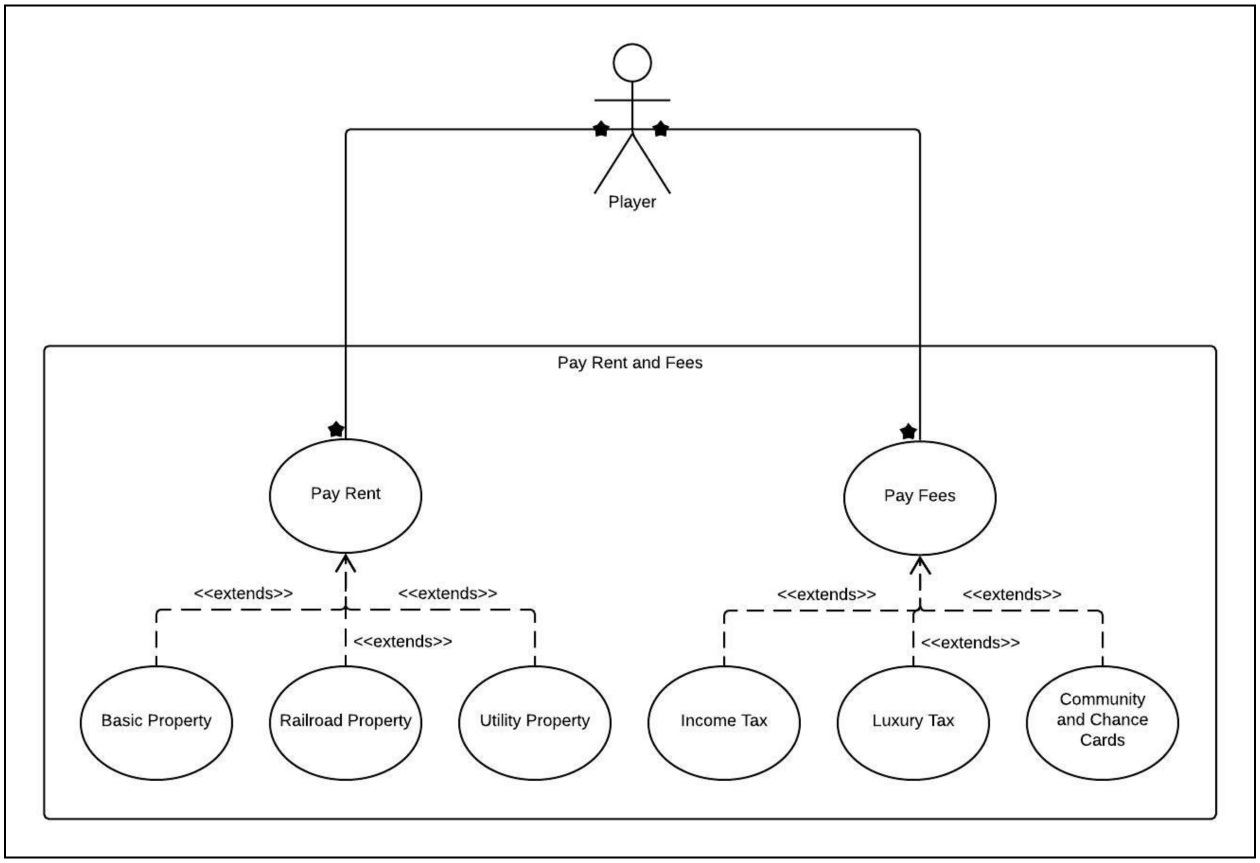
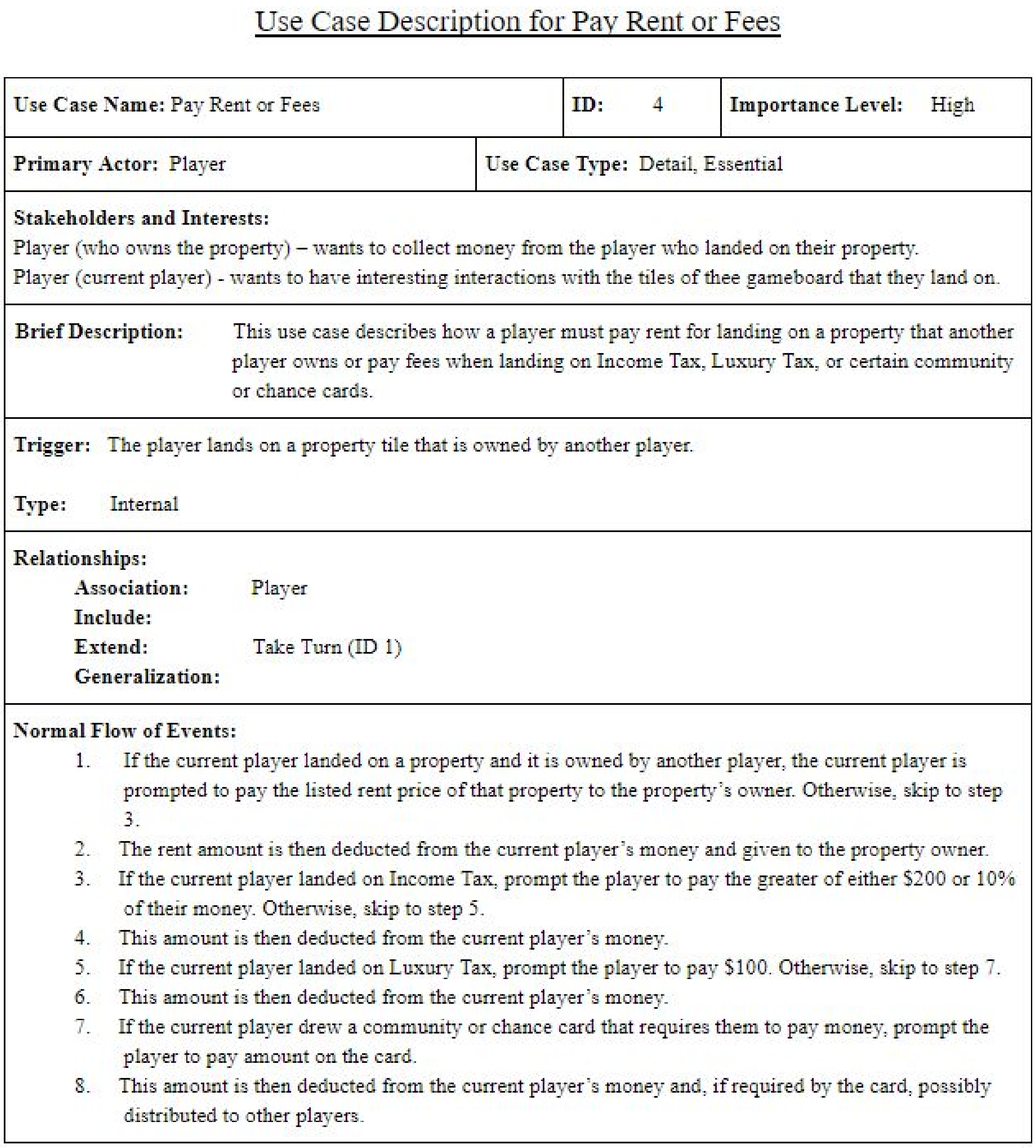


Figure 12: Use case for paying property rent or fees.

In Figure 13, the Pay Rent or Fees use case shows whether a player has to pay rent to a property owner or pay fees to the bank. The Pay Rent or Fees use case is associated with player since the use case determines if the player has to pay money to a property owner or the bank . Pay Rent or Fees extends Take Turn (ID 1) because both paying rent and pay fees is an optional behavior depending on if the player moves to a owned property, Income Tax, Luxury Tax, or a community chest or chance card that requires a payment to the bank. So, Pay Rent or Fees extends the functionality of Take a Turn. We decided that Pay Rent or Fees should not extend

Move (ID 2) because once move finishes, the requirement to pay rent or fees begins while the player is still taking his or her turn. For the flow of events of Pay Rent or Fees, see the subsection in Figure 13.



Use Case Description for Pay Rent or Fees

Use Case Name: Pay Rent or Fees	**ID:** 4	**Importance Level:** High

Primary Actor: Player	**Use Case Type:** Detail, Essential

Stakeholders and Interests:
Player (who owns the property) – wants to collect money from the player who landed on their property.
Player (current player) - wants to have interesting interactions with the tiles of thee gameboard that they land on.

Brief Description: This use case describes how a player must pay rent for landing on a property that another player owns or pay fees when landing on Income Tax, Luxury Tax, or certain community or chance cards.

Trigger: The player lands on a property tile that is owned by another player.

Type: Internal

Relationships:

Association:	Player
Include:	
Extend:	Take Turn (ID 1)
Generalization:	

Normal Flow of Events:

1. If the current player landed on a property and it is owned by another player, the current player is prompted to pay the listed rent price of that property to the property's owner. Otherwise, skip to step 3.
2. The rent amount is then deducted from the current player's money and given to the property owner.
3. If the current player landed on Income Tax, prompt the player to pay the greater of either $200 or 10% of their money. Otherwise, skip to step 5.
4. This amount is then deducted from the current player's money.
5. If the current player landed on Luxury Tax, prompt the player to pay $100. Otherwise, skip to step 7.
6. This amount is then deducted from the current player's money.
7. If the current player drew a community or chance card that requires them to pay money, prompt the player to pay amount on the card.
8. This amount is then deducted from the current player's money and, if required by the card, possibly distributed to other players.

Figure 13: Use case description for paying property rent or utility fees.

Structural Model

Internally, M0n0p0ly is structured around the singleton class, GameLoop, as shown below in Figure 14. This class contains the Gameboard, the list of selectable player icons, and the save/load functionality. The GameBoard class contains a list of players, an array of tiles, several lists of Cards (Community and Chance), a random number generator, the Roll and Move methods, the turn-state, which player's turn it is, and the and the initialization code for reading in the community and chance card data from .txt files. This is also the top class that will be saved/loaded when using save/load. (The GUI class, MainWindow.xaml.cs, which has the button controls, contains the initial calls to GameLoop and Gameboard, in addition to some of the game flow logic found in the Taking-a-Turn use case. Calls to update the GUI could not be placed in the GameLoop class, and, since we were not confident in our ability figure out how to implement the MVVM within the duration of the project while still adding enough features to flesh out the program, we elected to move some of the functionality to this GUI class.)

Game loop and Gameboard are connected by an association relationship. The player has an aggregation relationship because a player is a logical part of the Gameboard. The tiles are a physical part of the Gameboard, so it has a composition relationship. Property and Player have a 2-way association, where Players can have multiple Properties and each Property can have one Player (their owner).

Tile is an abstract class. Property, Jail, Free Parking, Go To Jail, Pass Go, and CardSpace are specifications of Tile, so the group gave these tiles a generalization relationship. Property is an abstract class. Basic, Railroad, and Utility are specifications of Property, so these tiles have a generalization relationship. Basic is a type of Property without any more specific attributes, unlike Railroad and Utility, but there does need to be a way to distinguish them from the other Property types, so making Basic properties their own class served that purpose.

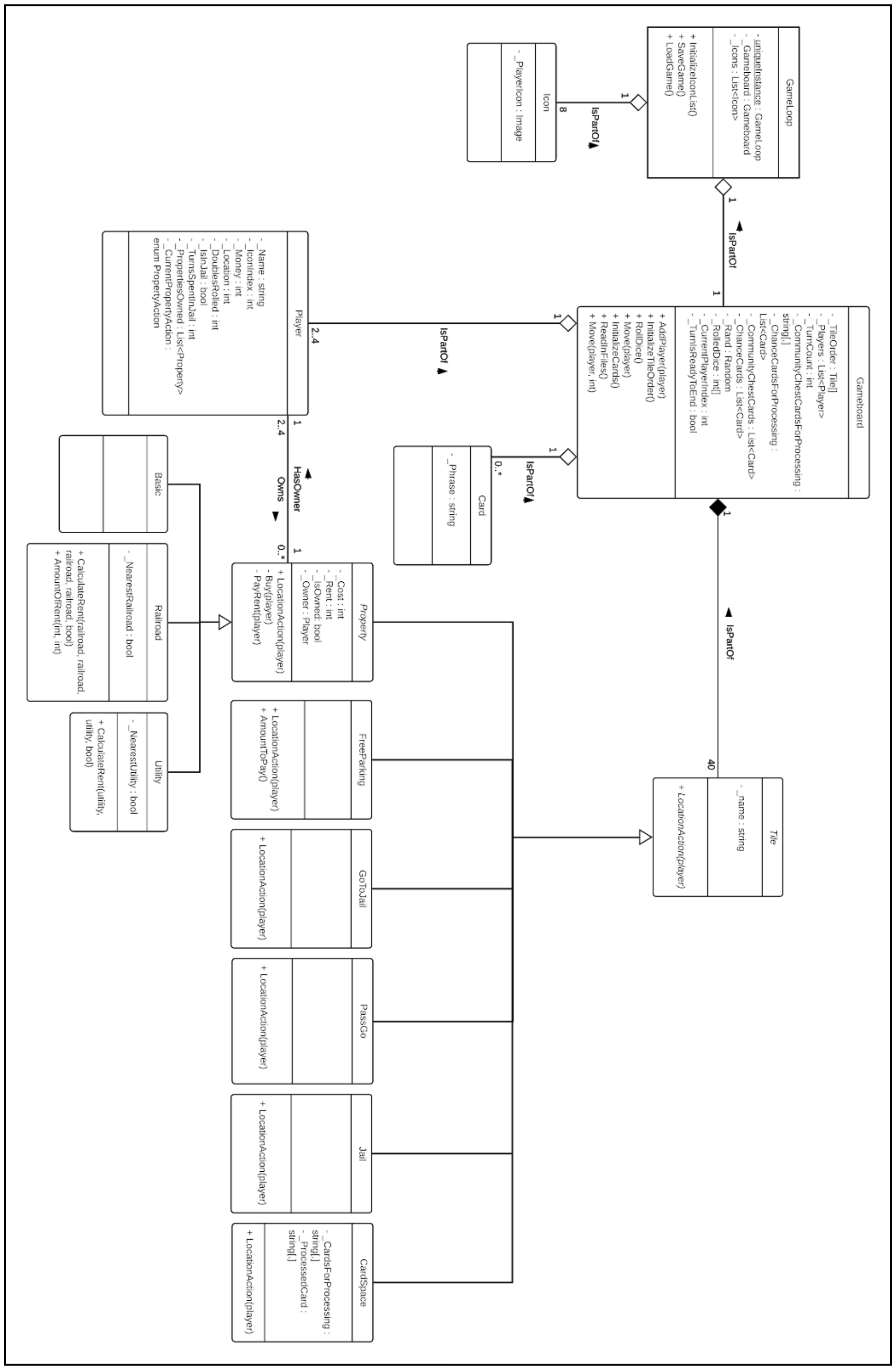


Figure 14: Structural model for final version of M0n0p0ly.

Behavioral Models

The team decided the major use cases are Move, Purchase Property, and Pay Rent, which are all part of the Take a Turn use case. The Move use case includes whether the player does not roll doubles or rolls doubles once, twice, or three times. Figure 15 describes the different possibilities of moving to a tile. A player rolls the dice to begin their turn. If the player does not roll doubles or rolls doubles once or twice, the player moves to either pass go, visit jail, or land on a property, free parking, Go to Jail tile, luxury tax, income tax, community chest tile, or chance tile. If a player rolls doubles for the third time, the player is sent to jail. The following turn, the player rolls the dice. If the player does not roll doubles, the player remains in jail and reattempts to roll doubles on the next turn. If the player rolls doubles or three turns have passed, the player gets out of jail.

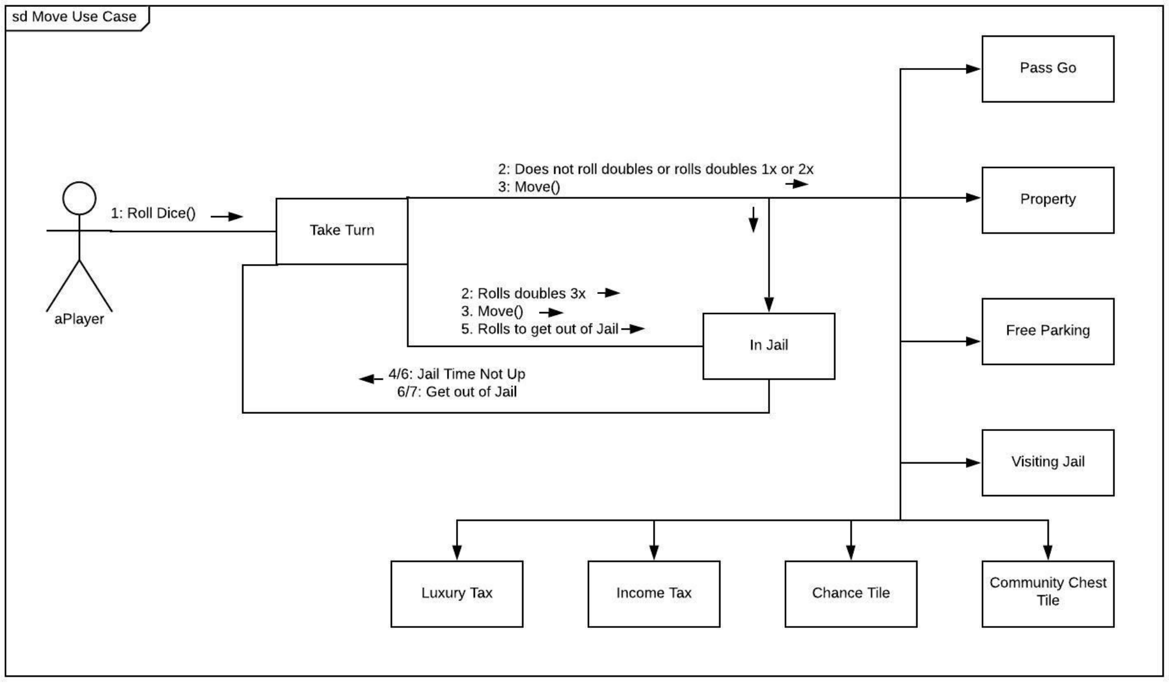


Figure 15: Communication diagram for the use case of moving.

Figure 16 shows the dynamic aspects of landing on a property. A player moves to a property. If a basic, railroad, or utility property is available, the player can purchase the property as part of location action. If the property is already owned, the player pays rent to the owner through location action. For more information on calculating the rent of the railroads and utilities, see Figure 12.

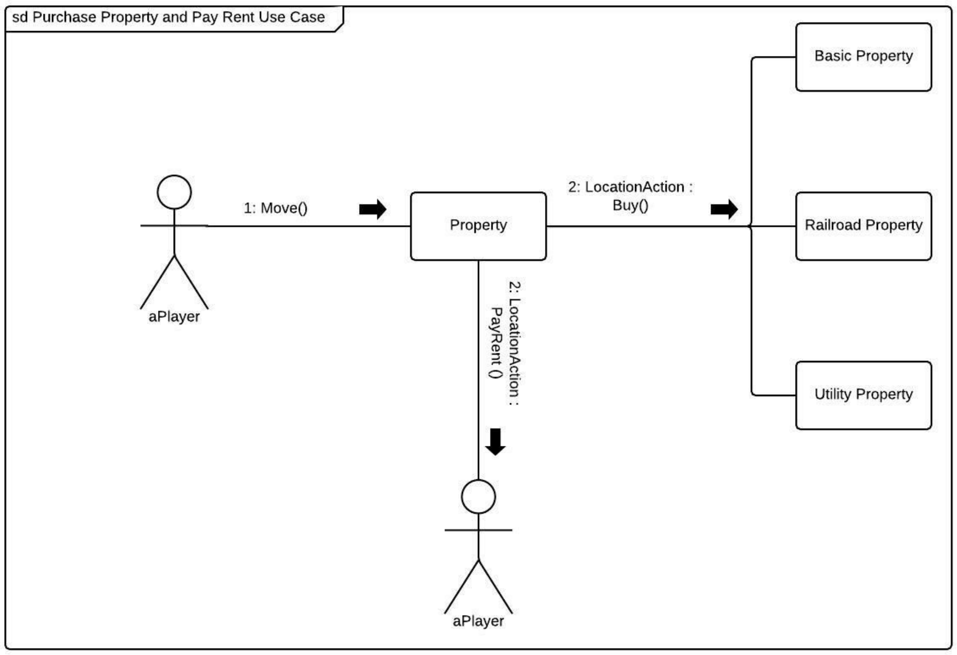

sd Purchase Property and Pay Rent Use Case
1: Move()
Property
2: LocationAction : Buy()
Basic Property
Railroad Property
Utility Property
2: LocationAction : PayRent ()
aPlayer
aPlayer

Figure 16: Communication diagram for the use cases of purchasing property and paying rent.

Figure 17 shows the various aspects of landing on or being sent to jail. When the player is in jail, the player has three turns to roll doubles to get out of jail. After the third turn, the player gets out of jail regardless. When a player enters a the Jail tile, it is checked whether or not their state is *in Jail*. If they are not in jail, they are set in *Just Visiting*; the following next turn, the player is free to leave and take their turn as normal.

Alternatively, if the player *is in Jail*, they are set to be *in Jail* on the player class. During their next turn (turn 1), if the player rolled a double, they are released from jail. Otherwise they must wait in jail until next turn (turn 2). This process is repeated until they roll doubles or three turns have passed causing the player to be released. Due to time constraints, the project did not implement the use of "Get out of Jail Free" cards from community chest or chance or the paying of a fee to be released early.

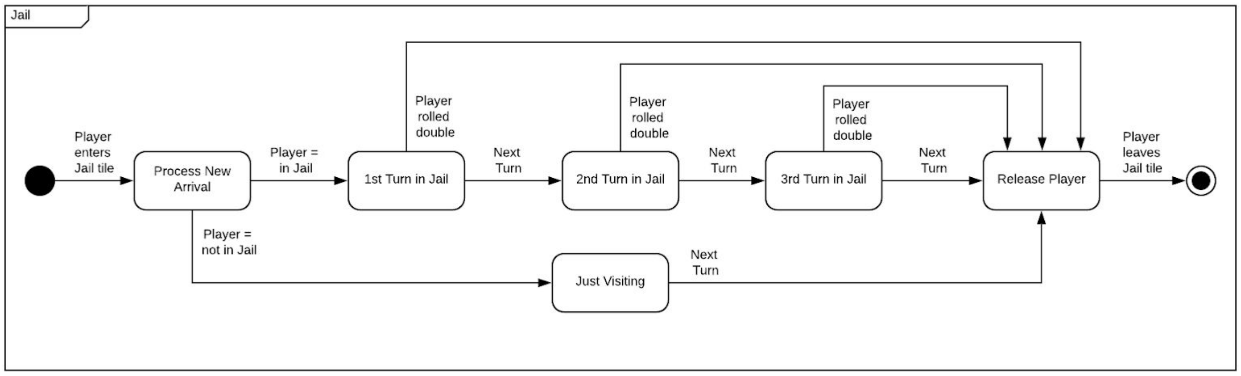


Figure 17: Behavioral state machine for jail.

The various states of a property ownership are shown in Figure 18. When the game is started, all properties are unowned. As the game progresses, a player may purchase a property, which will, from the property's perspective, set the property's state to *Active*. The lifetime of properties is equivalent to that of the game. With the time constraints of the project, we were not able to implement whether a player can mortgage the property. So, *Mortgaged* is not a current state of property in our game. Another feature we were not able to implement was allowing players to put houses and hotels on monopolies (or all the properties of one color).

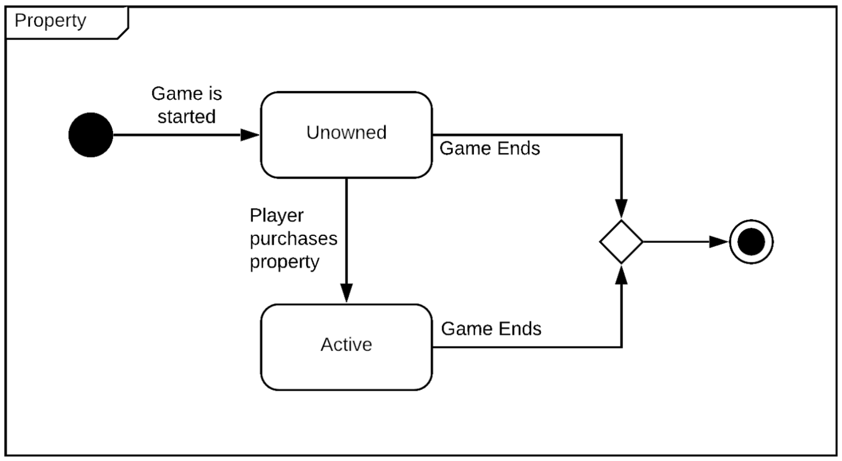


Figure 18: Behavioral state machine for property.

Figure 19 describes the sequence of events throughout a player's turn. To start his or her turn, a player rolls the dice. If the player rolls doubles three times, he or she is sent to jail and set to the *In Jail* state. The program checks whether the player is *In Jail*. If the player is *In Jail*, the program checks if doubles were rolled or if the player completed his or her third turn. If either of these occurred, the player gets out of jail.When the player is not in jail, he or she can move to either pass go, visit jail, or land on a property, free parking, Go to Jail tile, luxury tax, income tax, community chest tile, or chance tile. The icon position is updated. Next, Location Response is called which loads various forms for purchasing a property, paying rent, drawing a community chest card, drawing a chance card, landing on income tax, or landing on luxury tax. These forms initiates the tile's location action upon interaction and the results are subsequently displayed. The project allows 2 to 4 players in the game, which falls short of the desired 2 to 8 players. The player with the most amount of money wins the game after 20 turns.

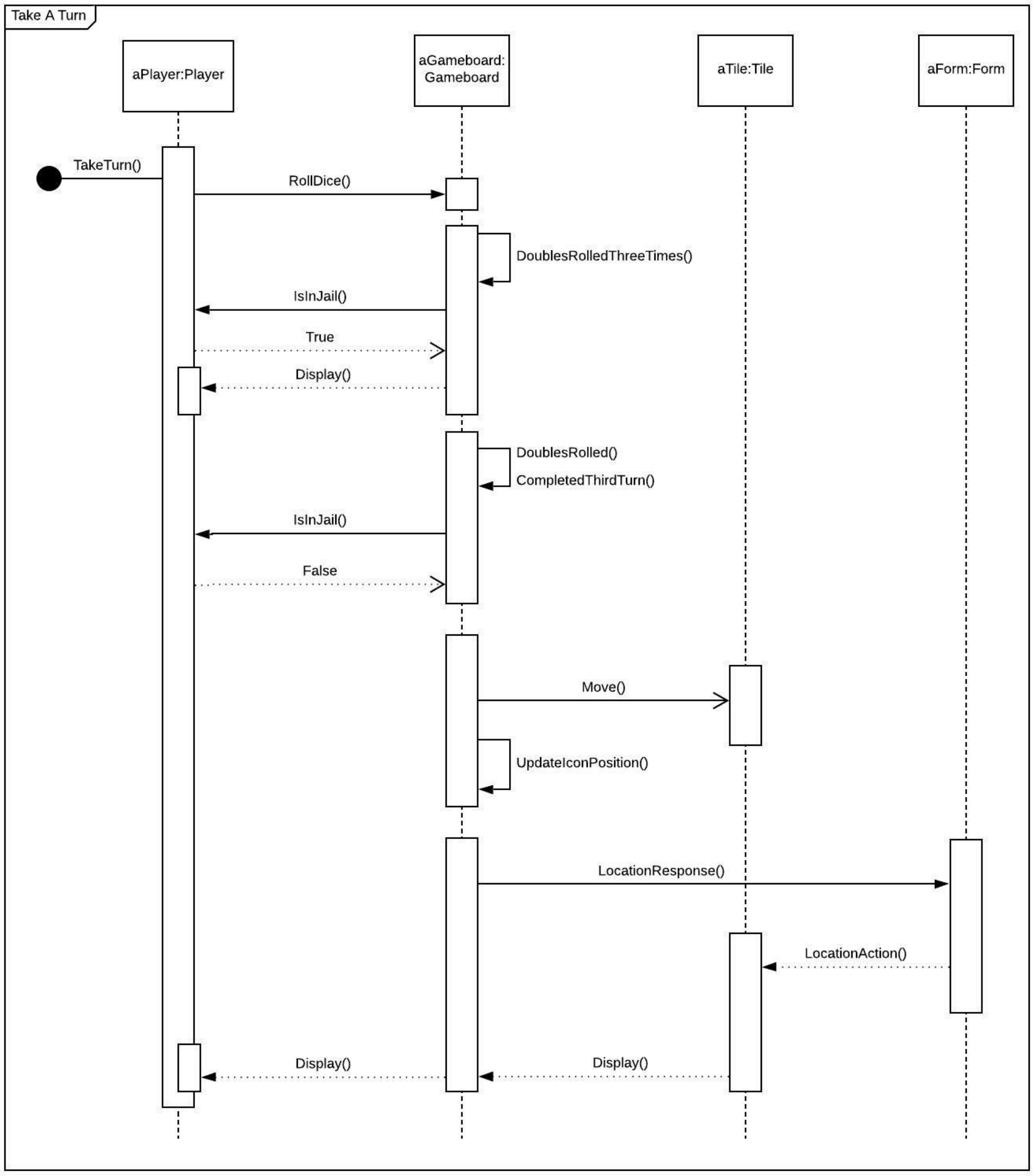


Figure 19: Sequence diagram for taking a turn.

Appendix

This section is devoted to walking through and explaining the graphical user interface (GUI) of M0n0p0ly. Note that if launching the game from the M0n0p0ly executable file (.exe), the files CommunityCard.txt and ChanceCard.txt need to be in the same directory as the M0n0p0ly executable file.

When the game is started, the first thing to appear is the title screen, as in Figure 20. There are three buttons: “New Game”, “Load Game”, and “Exit”. “New Game” opens up the player customization screen (Figure 21). “Load Game” opens the load file dialog (Figure 24), and if a save file is successfully loaded, opens the main game window (Figure 22) with the data from the save file correctly loaded and displayed to the player(s). “Exit”, meanwhile, simply closes the game.

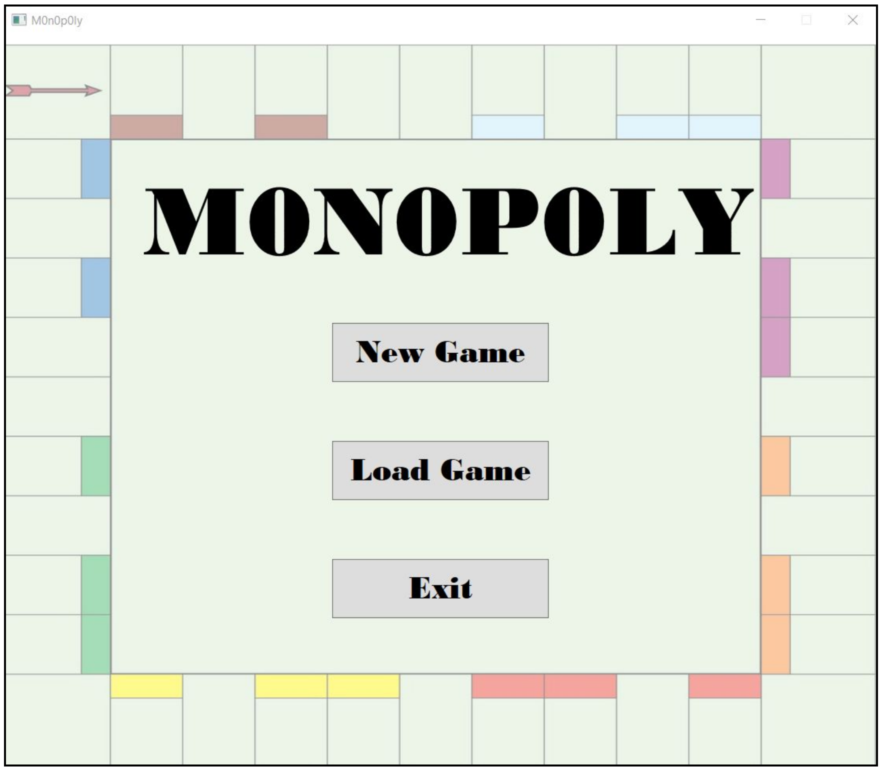
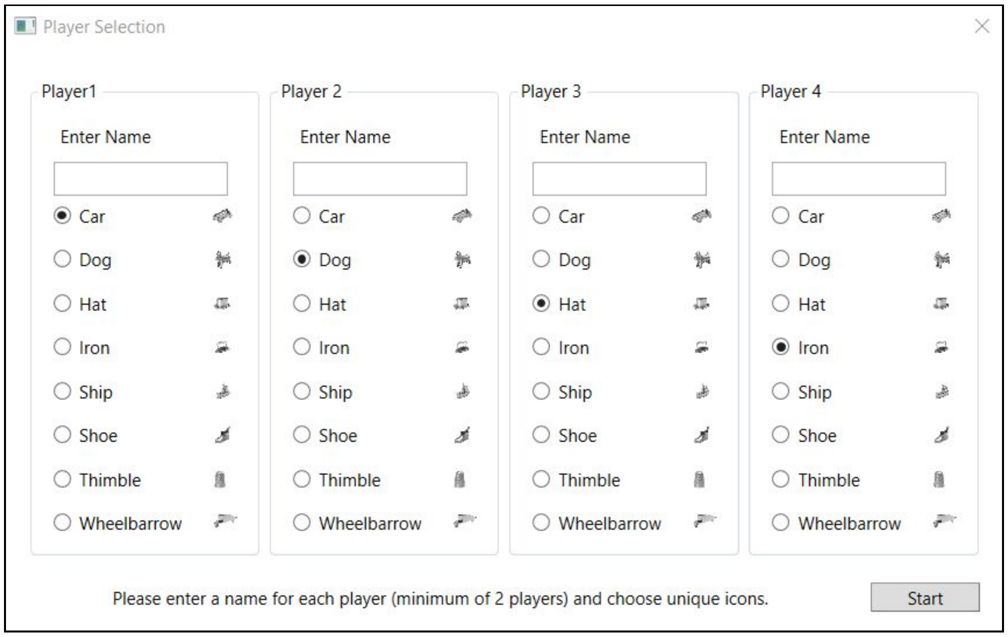


Figure 20: Title screen for M0n0p0ly.

The player customization screen (Figure 21) allows for 2 to 4 players to play the game. Each player may enter their name and choose from 1 of 8 icons to represent them on the gameboard. Note that there must be *at least* 2 players and all players must choose different icons in order to start the game. During the game, turns will proceed in order from Player 1 to Player 2 to (if they exist) Player 3 to Player 4. Clicking the “Start” button will then launch the main game window (Figure 22) and start the game with the players that were entered.



Player Selection

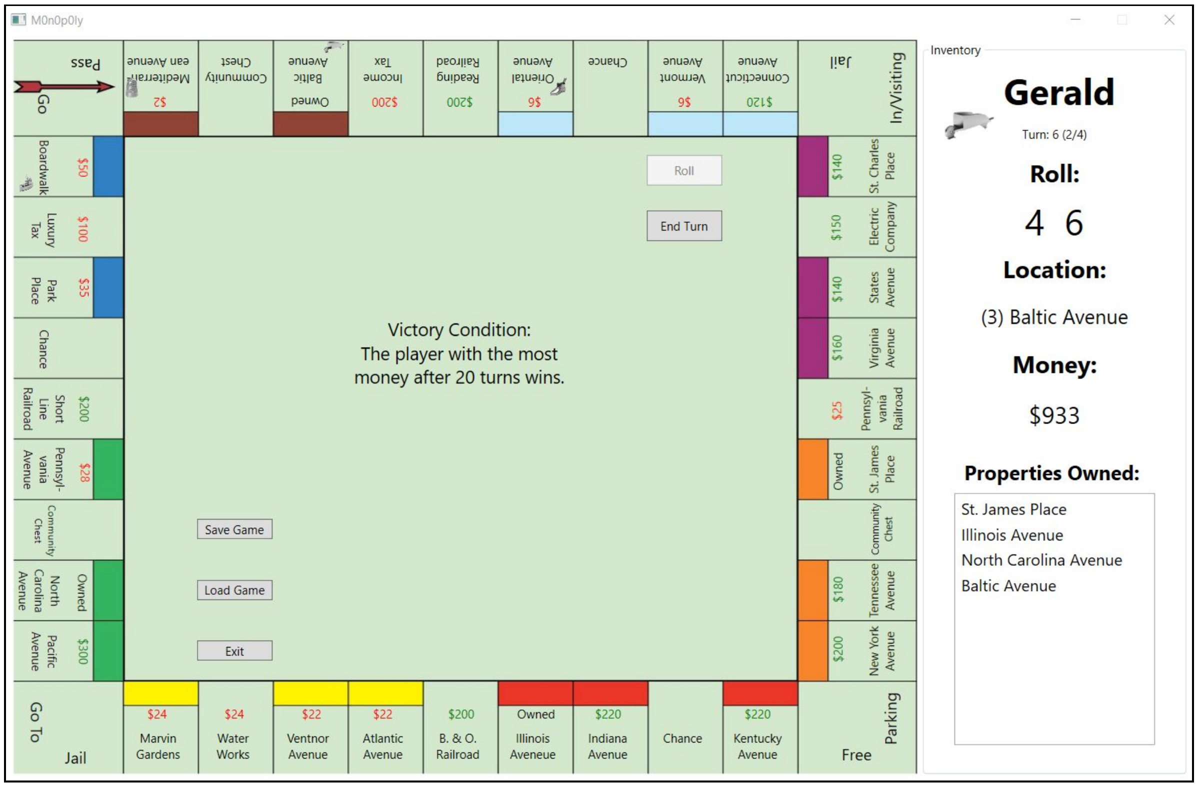
Player1	Player 2	Player 3	Player 4
Enter Name	Enter Name	Enter Name	Enter Name
◉ Car	○ Car	○ Car	○ Car
○ Dog	◉ Dog	○ Dog	○ Dog
○ Hat	○ Hat	◉ Hat	○ Hat
○ Iron	○ Iron	○ Iron	◉ Iron
○ Ship	○ Ship	○ Ship	○ Ship
○ Shoe	○ Shoe	○ Shoe	○ Shoe
○ Thimble	○ Thimble	○ Thimble	○ Thimble
○ Wheelbarrow	○ Wheelbarrow	○ Wheelbarrow	○ Wheelbarrow

Please enter a name for each player (minimum of 2 players) and choose unique icons.

Figure 21: Player customization screen.

Figure 22 shows the main game screen from a game that is several turns in. The gameboard shows all of the tiles that players can land on, along with which properties are owned by that player, which ones are available to be purchased (and the money required), and which ones the player will have to pay rent on (and the money required). In addition, the player icons are on each tile where that respective player is. In the very center of the board is the victory condition, to remind players what their goal is. The victory condition is that the player with the most money after 20 turns of play will be crowned the victor.

Also in the middle of the board are several buttons. "Roll" is how a player takes their turn. It will do so by rolling two dice, moving the player to their new location, and triggering the action of that new space. If doubles were rolled, the player can click this button again. Otherwise, it will become greyed-out and the player must end their turn by clicking the "End Turn" button. The next player's turn will then start.



Victory Condition:
The player with the most money after 20 turns wins.

Gerald

Turn: 6 (2/4)

Roll:

4 6

Location:

(3) Baltic Avenue

Money:

$933

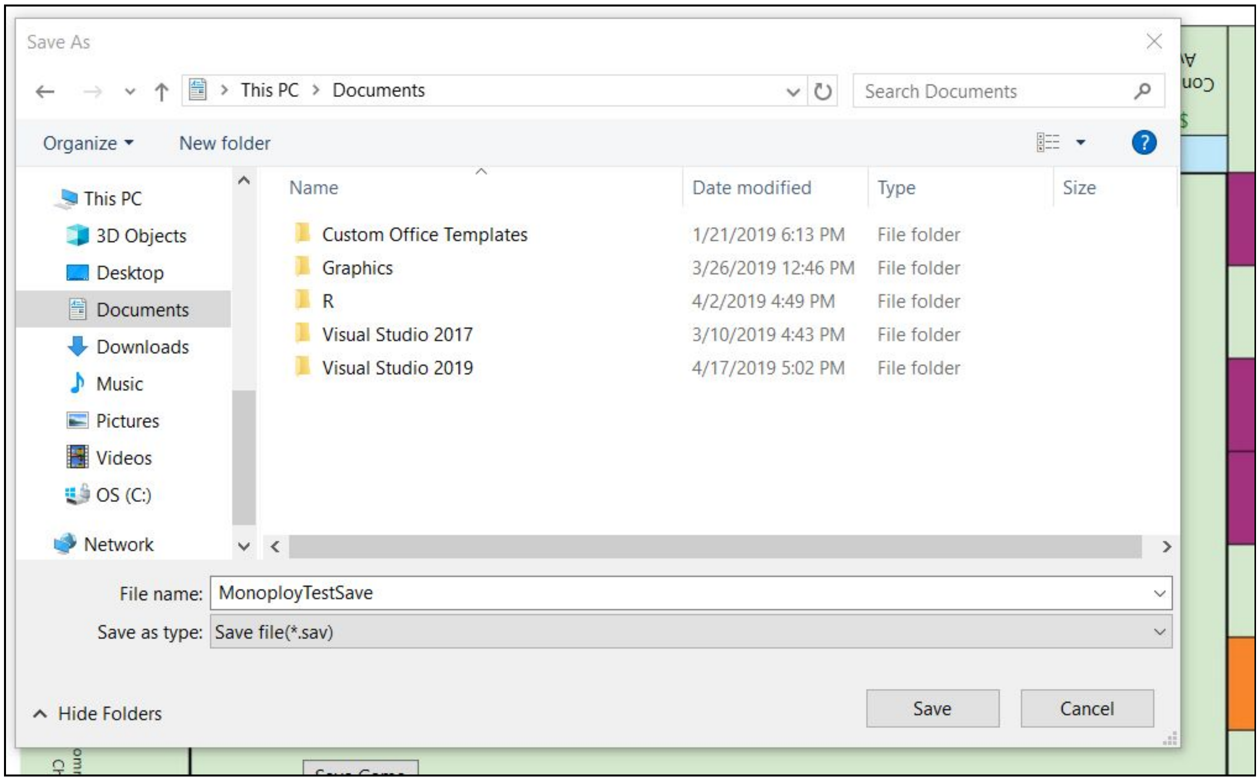
Properties Owned:

St. James Place
Illinois Avenue
North Carolina Avenue
Baltic Avenue

Figure 22: Main game screen.

Next, there are the "Save Game" and "Load Game" buttons. The "Save Game" button will open the save game dialog screen (Figure 23). Feedback will be given to the user if the save was successful. The "Load Game" button will open the load game dialog screen (Figure 24) and, if the load is successful, will inform the user, update all the game data to match that of the save file, and update all the player information displays. Finally, the “Exit” button closes the game.

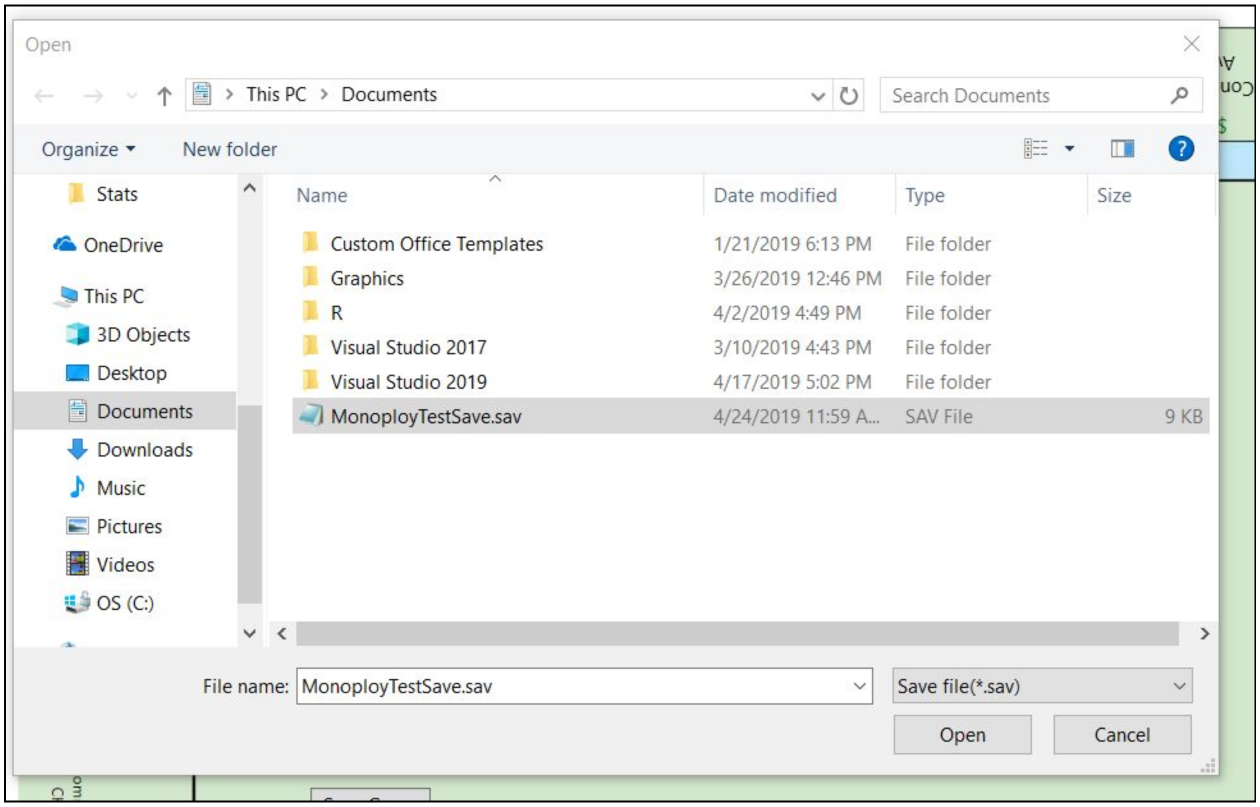
On the right-hand side of the screen is the player information display, which shows the current player’s name, icon, dice roll (after they have clicked “Roll”), location (tile location and name), money, and owned properties. The current turn and progress through the turn are also displayed here.



Name	Date modified	Type	Size
Custom Office Templates	1/21/2019 6:13 PM	File folder	
Graphics	3/26/2019 12:46 PM	File folder	
R	4/2/2019 4:49 PM	File folder	
Visual Studio 2017	3/10/2019 4:43 PM	File folder	
Visual Studio 2019	4/17/2019 5:02 PM	File folder	

File name: MonoployTestSave
Save as type: Save file(*.sav)

Figure 23: Save game screen.

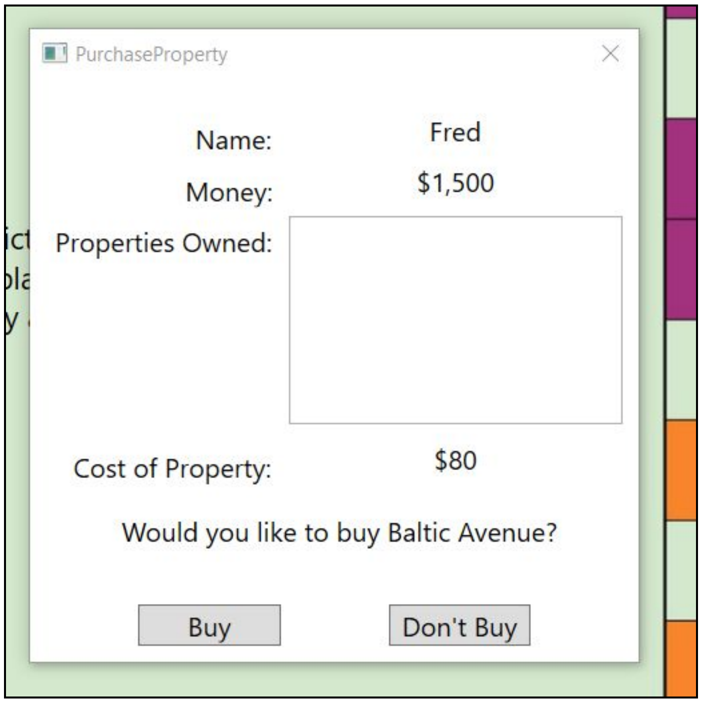


Name	Date modified	Type	Size
Custom Office Templates	1/21/2019 6:13 PM	File folder	
Graphics	3/26/2019 12:46 PM	File folder	
R	4/2/2019 4:49 PM	File folder	
Visual Studio 2017	3/10/2019 4:43 PM	File folder	
Visual Studio 2019	4/17/2019 5:02 PM	File folder	
MonoployTestSave.sav	4/24/2019 11:59 A...	SAV File	9 KB

File name: MonoployTestSave.sav
Save file(*.sav)

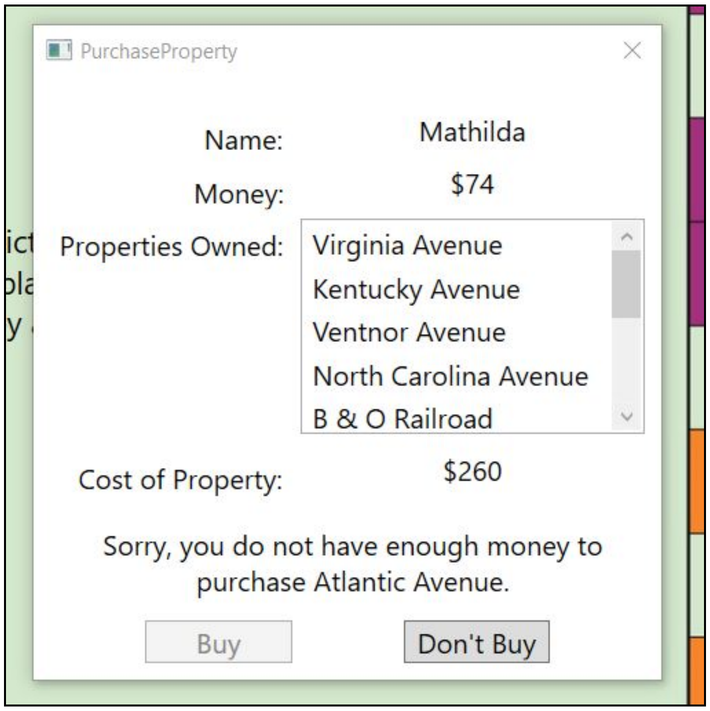
Figure 24: Load game screen.

In Figures 25 and 26, we have the two possible screen that can be displayed when landing on an unowned property. Figure 25 displays the name of the player, their money, their currently owned properties, the cost to purchase the new property, and has the normal behavior of allowing the player to purchase or to not purchase the property. Figure 26 also displays this data, but does not allow the player to make the purchase if they do not have enough money to do so. It should be noted that players are allowed to have negative amounts of money, the penalty for which is that no properties can be purchased. Having a negative amount of money can occur through rent payments or community/chance cards.



Name: Fred
Money: $1,500
Properties Owned:
Cost of Property: $80
Would you like to buy Baltic Avenue?

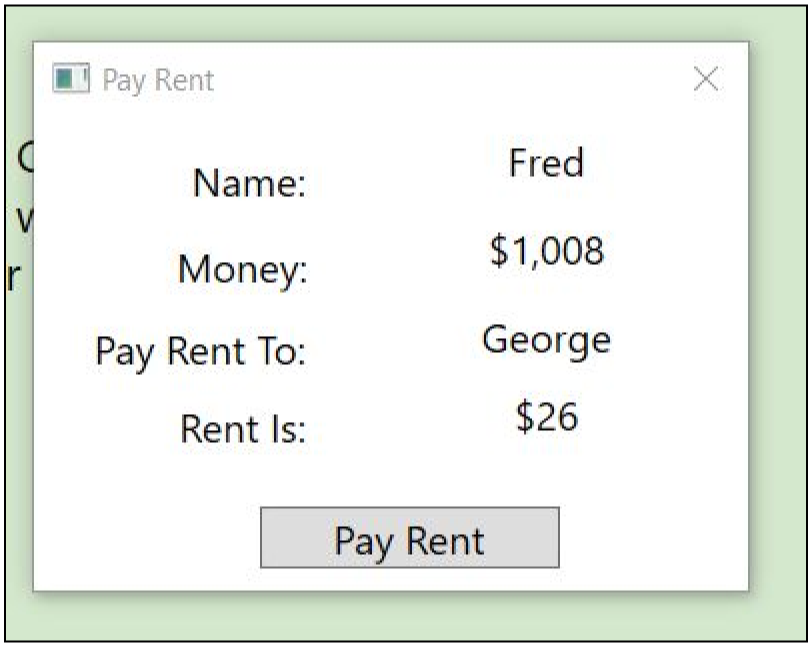
Figure 25: Purchasing a property.



Name: Mathilda
Money: $74
Properties Owned: Virginia Avenue
Kentucky Avenue
Ventnor Avenue
North Carolina Avenue
B & O Railroad
Cost of Property: $260
Sorry, you do not have enough money to purchase Atlantic Avenue.

Figure 26: Purchasing a property with too little money.

Figure 27 shows the rent payment screen. The name of the current player, their money, the name of the owner of the property, and the rent due are all displayed to the player, who has no choice but to pay this amount.



Name: Fred

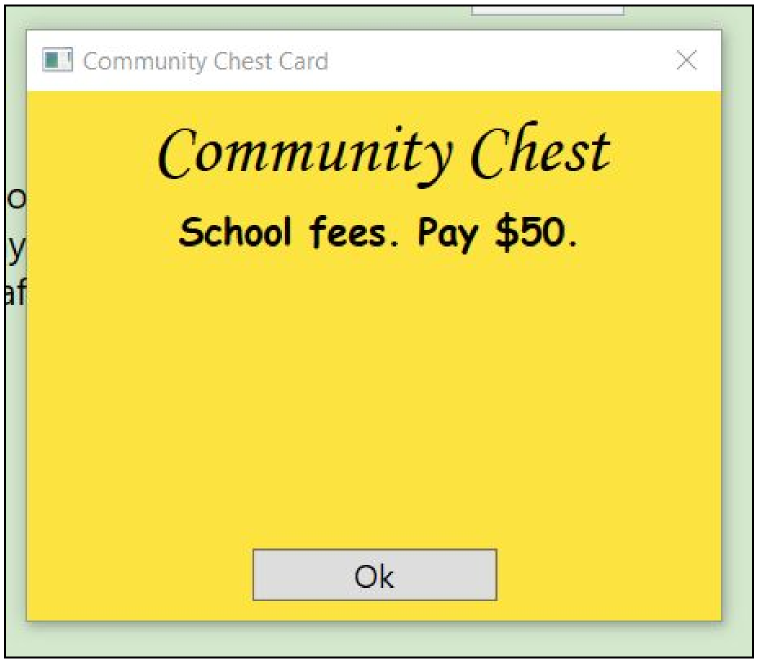
Money: $1,008

Pay Rent To: George

Rent Is: $26

Figure 27: Paying rent on a property.

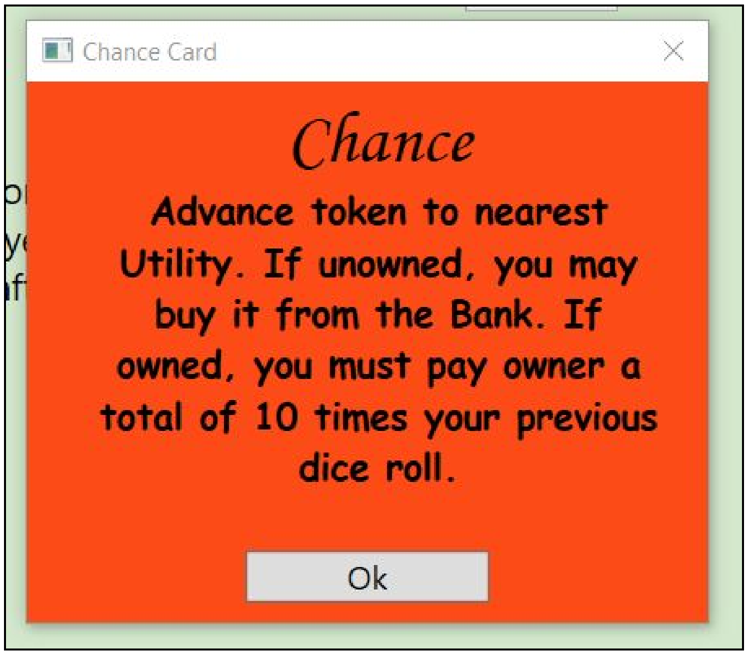
In Figure 28 is an example of a simple community chest card that will be selected at random from the "CommunityCard.txt" file when the player lands on a community chest tile. This one will subtract $50 from the current player's money. In Figure 29 is an example of a complicated chance card that will be selected at random from the "ChanceCard.txt" file when the player lands on a chance tile. This one will find the utility nearest the current player's location and move (by advancement) the player to that tile. If that utility is already owned by another player, the current player must pay rent equal to 10 times the dice roll that made them land on the chance card tile.



Community Chest

School fees. Pay $50.

Figure 28: Community chest card example.

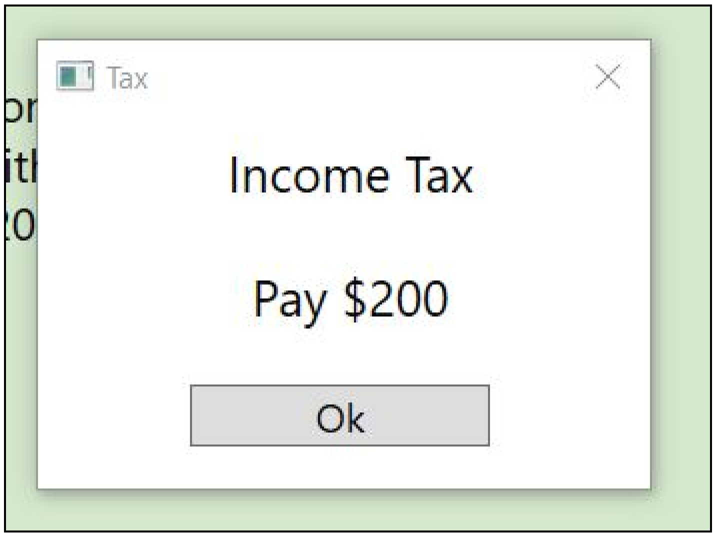


Chance

Advance token to nearest Utility. If unowned, you may buy it from the Bank. If owned, you must pay owner a total of 10 times your previous dice roll.

Figure 29: Chance card example.

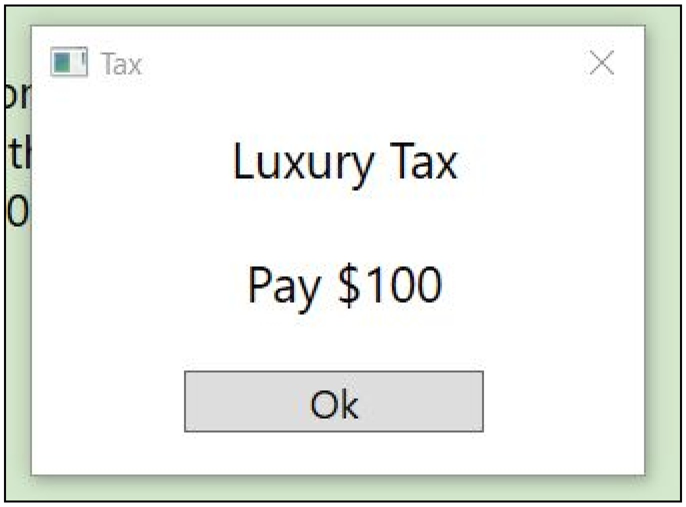
If a player lands on the Income Tax tile, they must pay $200 or 10% of their money, whichever is greater (Figure 30). Similarly, if a player lands on the Luxury Tax tile, they must pay $100, as is shown Figure 31.


Tax

Income Tax

Pay $200

Figure 30: Income tax payment.

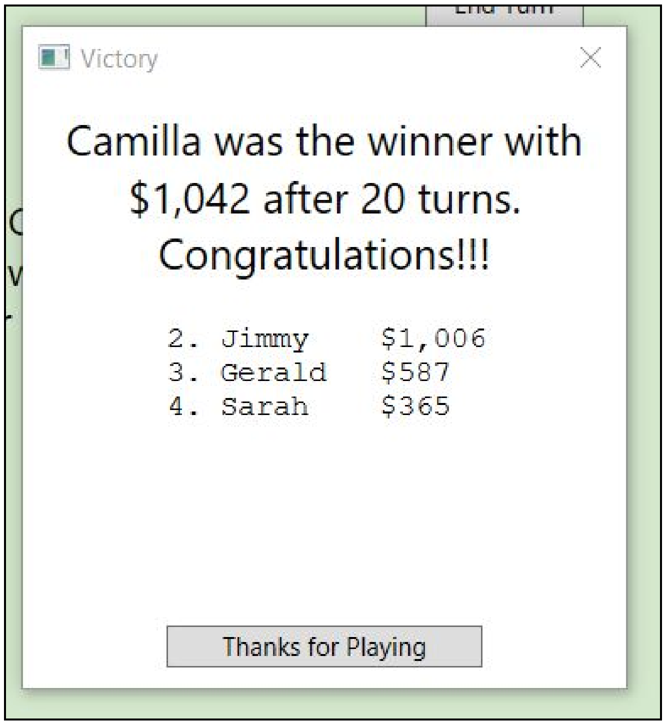

Tax

Luxury Tax

Pay $100

Figure 31: Luxury tax payment.

Finally, we have the victory screen (Figure 32). After each player has played through their 20th turn, the game ends and the player with the most money is declared the winner. The remaining players and their money are also displayed in decreasing order. Clicking on the "Thanks for Playing" button will close the game.



Camilla was the winner with $1,042 after 20 turns. Congratulations!!!

```
2. Jimmy    $1,006
3. Gerald   $587
4. Sarah    $365
```

Figure 32: Victory results screen.

References

Balduino, R. *Basic Unified Process: A Process for Small and Agile Projects*. Retrieved https://www.eclipse.org/proposals/beacon/Basic Unified Process.pdf

Dennis, A., Wixom, B., & Tegarden, D. (2015). *System Analysis & Design: An Object-oriented Approach with UML* (5th ed.). Davers, MA: John Wiley & Sons.