BILKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CS319

Iteration 2

Project Analysis Report

Group 1F-Monopoly

Göktuğ Gürbüztürk 21702383

Ege Şahin 21702300

Aybars Altınışık 21601054

Ahmet Feyzi Halaç 21703026

Bulut Gözübüyük 21702771

1. Introduction

Monopoly is a multiplayer game where the aim of each player is to drive another player into bankruptcy. Players roll dice to move on the map and buy regions from the world map. In addition to that, they can trade with other players and develop their regions by constructing new buildings which cause more rent from the player who comes to the zone. In this way, the player collects money from other players' regions and causes other players to go bankrupt. Also, the theme of the game is on the pandemic, which adds features to the game regarding virus infection and spreading. This report describes the game in terms of functional requirements and non-functional requirements which explain components of the game. In addition to that, there are different types of diagrams which constitute the system models of the game. In the use case diagrams, how players can interact with the game is shown. Also, the class diagram shows the classes of the game and the associations between them. Lastly, there is a dynamic modeling section which shows the states of the game and interactions between different objects. At the end of the report, there is a user interface section that shows the display of the game screen in different states.

2. Rules

- Since this game is digital, there is no limitation of the number of houses or hotels.
- Each player has a name, pawn, color and id.
- Since this game is digital there is no bank. The money is limitless and all deed cards are stored by the computer. Computer executes the financial operations such as selling regions or paying salaries so the computer is similar to the banker of the board game.
- Players roll dice before starting the game. The player who rolled the highest total starts the game first and others continue after him/her in descending order of their rolled value. If some players rolled the same total, they will roll again to decide the order between them.
- At the beginning of the game, all the players are in the start region with the same amount of money. The amount of money is fixed in every game.
- Buying buildings is not related to where the buyer's pawn is located. In other words a player can buy a building without going to that location. The reason for this is that in our game infection may affect the players' path more than the normal game's path changing events, so going to a particular region could be harder.
- If you throw doubles, you move your pawn as usual, the sum of the two dice, and are subject to any privileges or penalties pertaining to the space on which he/she lands. Retaining the dice, throw again and move your pawn as before. If you throw doubles three times in succession, you do not go to the quarantine region (jail). Because we add another precaution (infection check region) for preventing the player from moving too much in the same turn. Infection check region may also send a player to quarantine (jail).
- Each time players pass the start region they get a salary from the computer.
- Since our game theme is about infection, we add a new type of region to the game which is infected regions.At the start of the game, a random region is selected by the game and becomes infected (any region but start). This infected region is changed in every two turns. If players come to this region, they become infected. After two turns, they become free of infection but the regions they stopped while they are infected become infected too for two turns.

- Whenever a player lands on an unowned city, he/she may buy that city from the computer. After that the deed card will be given to the player who buys that city. If the player does not wish to buy the city there will be no auction because in auction all players must communicate at the same time but our game is turn based.
- When a player lands on a city of another player, he/she must pay rent. If the city is mortgaged, no rent will be paid.
- There are no color groups of cities in our game. Initially all the regions are colorless. When a player buys a city, the city possesses the color of the player. When a player buys three or more consecutive cities, all regions possess the same color and form a group. The reason for this rule is that in our game there is an agreement feature which lets players form agreements between themselves. So we aimed to increase the usage of the agreements by changing the original concept of the game. (If the two cities are seperated with another type of region, they are not consecutive so even if a player has three cities which are seperated with some other region types, do not form a group.)
- If a player has three or more consecutive regions (a group), the rent of those regions (without buildings on region) duplicate. Players can also buy buildings to their consecutive regions. Buildings increase the rent of the city. The maximum number of houses allowed on a single city is four. The maximum number of hotels allowed on a single city is one. There is no rule for order of buying buildings (players can buy a hotel before having houses on his/her other consecutive regions). This rule is different from the original game because if we did not let players do it, for some groups of consecutive regions, buying a hotel would be so difficult (if a player forms a group of five consecutive regions).
- In our game there is no community chest. Instead there are only chance regions. We implemented the community chest tasks as chance tasks.
- There is no exit quarantine chance card, because a person should not exit quarantine before he/she gets better.
- There cannot be a region mortgaged with buildings on it. In order to mortgage a city, first buildings must be selled.
- In the original game, the owner of a city may forget to collect its rent. In our game there is no such rule because the computer automatically handles it.
- If a player lands on a pirate invasion region, he/she must pay money to the computer(Similar to income tax region).
- In our game players may become infected. There are three ways to become infected:

    - In our map, there is a quarantine region, similar to the jail region in the original game. If players land on that region they get infection because the virus is transmitted.
    - If a player stops on an infected region they get an infection. After that, for two turns players carry the infection with themselves and infect the regions they stopped in these two turns. These infected regions only stay as infected for two turns.
    - Players can also get infection from chance cards, by landing chance regions.
- In our game players may go to quarantine and stay there for three turns. There are three ways to go to quarantine:
    - If a player passes the infection check region while being infected or stops at that region they go to quarantine (similar to go to jail region).
    - If all the players become infected at the same time, they go to the quarantine region directly. They do not wait there for three turns but pay some amount of money. If there are already one or more players in the quarantine region before everyone gets infection and everyone gets infection while they are there, they will also leave the quarantine region with others and pay the fee.
    - If a player gets infection from a chance card, he/she directly goes to the quarantine region.
- Since we already have two different ways to go to quarantine, we do not implement the original game's rules for quarantine (jail).
- There is no free parking region in our game. We used that region for infection check region
- Players can make an agreement with other players in their turns. The player states his/her offer and demand. These can be "sell a region", "take percentage of the rent of another player's region", "give money" and "pay a rent or not".
- Mortgage value of a city is different for each city. If a player mortgages a city, he/she receives the mortgage value of the city from the computer. Players cannot collect rent from mortgaged cities. In order to lift the mortgage, the player must pay %110 of the value he/she receives when mortgaging the city.

- A mortgaged city can be sold to other players by agreements. the buyer of the mortgaged city must pay the %110 of the mortgage value of the city to lift the mortgage.
- If a player starts his/her turn with negative money, and finishes it with negative money then he/she goes bankrupt and leaves the game. When there is only one player in the game he/she is the winner.

3. Functional Requirements

3.1. Starting a Game

Player can start a new game from the main menu or he/she can continue a load game.

When the player starts a new game he/she can adjust the number of players, each player's name, pawn and color. He/she can also remove players.

3.2. One Turn in a Game

In each turn, players play according to starting order. They will decide which game state option to perform and end their turn.

If a player rolls doubles (the same number on both dice) they earn one more chance to roll.

After every player plays their part, one turn ends and a new turn begins.

3.3. Player Options

There are several options a player can do or must do in his/her turn.

3.3.1. Roll Dice

Each player rolls dice and moves as many regions they rolled in every turn.

3.3.2. Buy a Region

This option is available only if the player is on a purchasable region (city). After this operation the card of the region will be added to their hand.

3.3.3. Buy a Building

When a player forms a group of consecutive regions, he/she can buy buildings to those regions.

3.3.4. Sell a Building

Players can sell buildings to the same amount of money they construct them. Players can select which buildings to sell.

3.3.5. Offer an Agreement

Players can offer agreements to other players in their turn and request something in return. There are four options to offer and request. These offers can be accepted by other players.

3.3.5.1. Sell a Region

Players can offer selling regions to other players and they can also request from other players to sell their regions.

3.3.5.2. Give a Percentage of the Rent of any Region

Players can form an agreement and split up the rent of a region between them. Whenever a player comes to that region and pays rent, the payment is split between those two players who have an agreement on that region.

3.3.5.3. Give Money

Players can give money to each other. The amount of money can be adjusted on the offer screen.

3.3.5.4. Pay a Rent or Not

Players can form an agreement that adjusts the payments of the rents.

3.3.6. Go to the Mortgage

When a region is mortgaged, the owner of that region receives the mortgage value from the bank.

3.3.7. Lift the Mortgage

A mortgage on a region can be removed if the owner pays the %110 of the mortgage value of that region to the bank. After that, the owner can collect rent from that region again.

3.3.8. End Turn

When a player has done playing, his/her turn ends and the next player will play.

4. Non-functional and Pseudo Requirements

4.1. Non-functional requirements :

- User Experience
  - In-game buttons and in-game texts' font size should be big enough so that users can click buttons with the mouse easily.
  - Font size should be minimum 18px.
  - The minimum button size should be 150x150 pixels.

- Performance and Graphics
  - The window resolution of the game will be 1920x1080.
  - The latency of the game after each input must be less than 0.1s.
  - The minimum framerate of the game must be always 30.

- Compatibility
  - The game will be compatible with Windows 10.

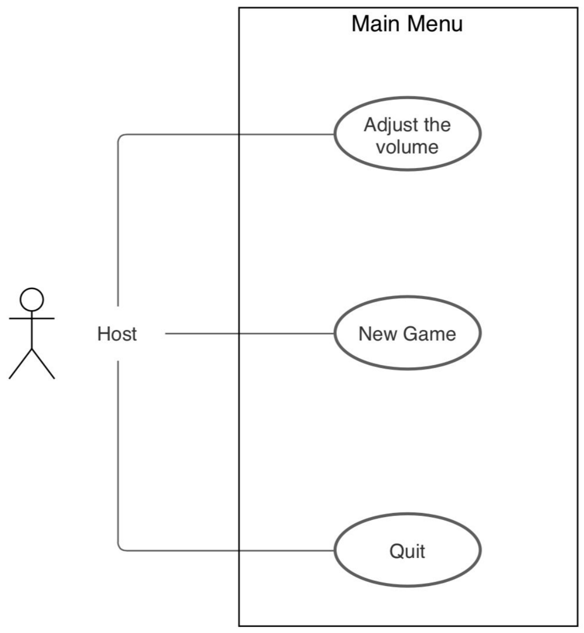
4.2. Pseudo requirements (Constraints) :

- The game will be written using the Java programming language.
- The game will be a desktop application in full screen mode.
- The button count on each window must not pass beyond 7 except the create game stage.

5. System Models

In this section, the game is described with the help of different UML models.

5.1. Use Case Diagrams



Use Case: New Game
Participating Actor: Host
Flow of Events:

1. Game is started with specified players.

Entry Conditions: Game has not started yet.
Exit Conditions: Screen switches to starting game screen.
Quality Requirements: None

Use Case: Adjust volume
Participating Actor: Host
Flow of Events:

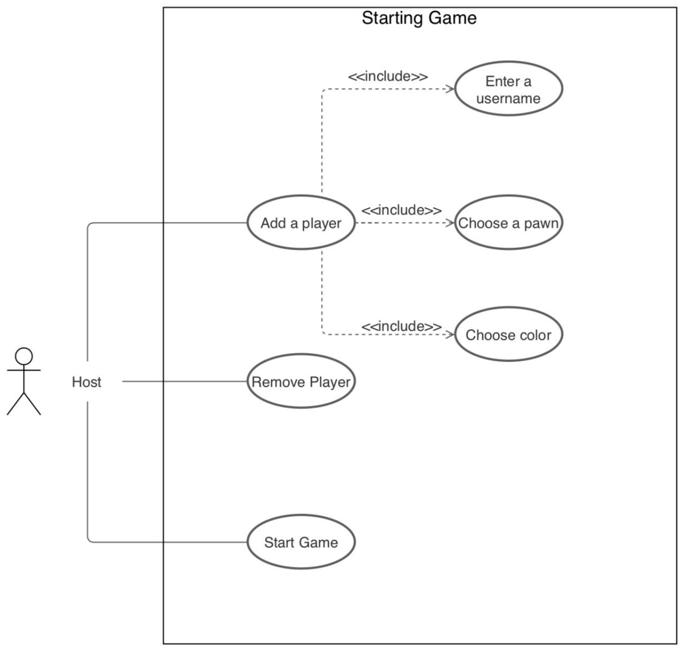
1. Host adjusts the volume.

Entry Conditions: Game has not started yet.
Exit Conditions: Volume is adjusted by the host.
Quality Requirements: None

Use Case: Quit
Participating Actor: Host
Flow of Events:

1. Host clicks the quit button.
2. Game is closed.

Entry Conditions: Game has not started yet.
Exit Conditions: Game is closed.
Quality Requirements: None



Use Case: Add a player
Participating Actor: Host
Flow of Events:

1. Player with specified username, pawn and color are added to the player list.

Entry Conditions: Username, pawn and color should not be used by other players and the game has not started yet.
Exit Conditions: Player with specified username, pawn and color is added to player list.
Quality Requirements: This use case should **include** 'Enter a username', 'Choose a pawn' and 'Choose color'. These use cases are initiated one by one when the host invokes these functions so that a player with specified features is added.

Use Case: Enter a username
Participating Actor: Host
Flow of Events:

1. Host enters the nickname of the player who will be added.

Entry Conditions: Username should not be used by other players and the game has not started yet.
Exit Conditions: Player with specified nickname is added.
Quality Requirements: None

Use Case: Choose a pawn

Participating Actor: Host
Flow of Events:

1. Host chooses a pawn for the player who will be added.

Entry Conditions: Pawn should not be used by other players and the game has not started yet.
Exit Conditions: Pawn of the specified player is selected by the host.
Quality Requirements: None

Use Case: Choose color
Participating Actor: Host
Flow of Events:

1. Host chooses color for the player who will be added.

Entry Conditions: Color should not be used by other players and the game has not started yet.
Exit Conditions: Color of the specified player is selected by the host.
Quality Requirements: None

Use Case: Remove player
Participating Actor: Host
Flow of Events:

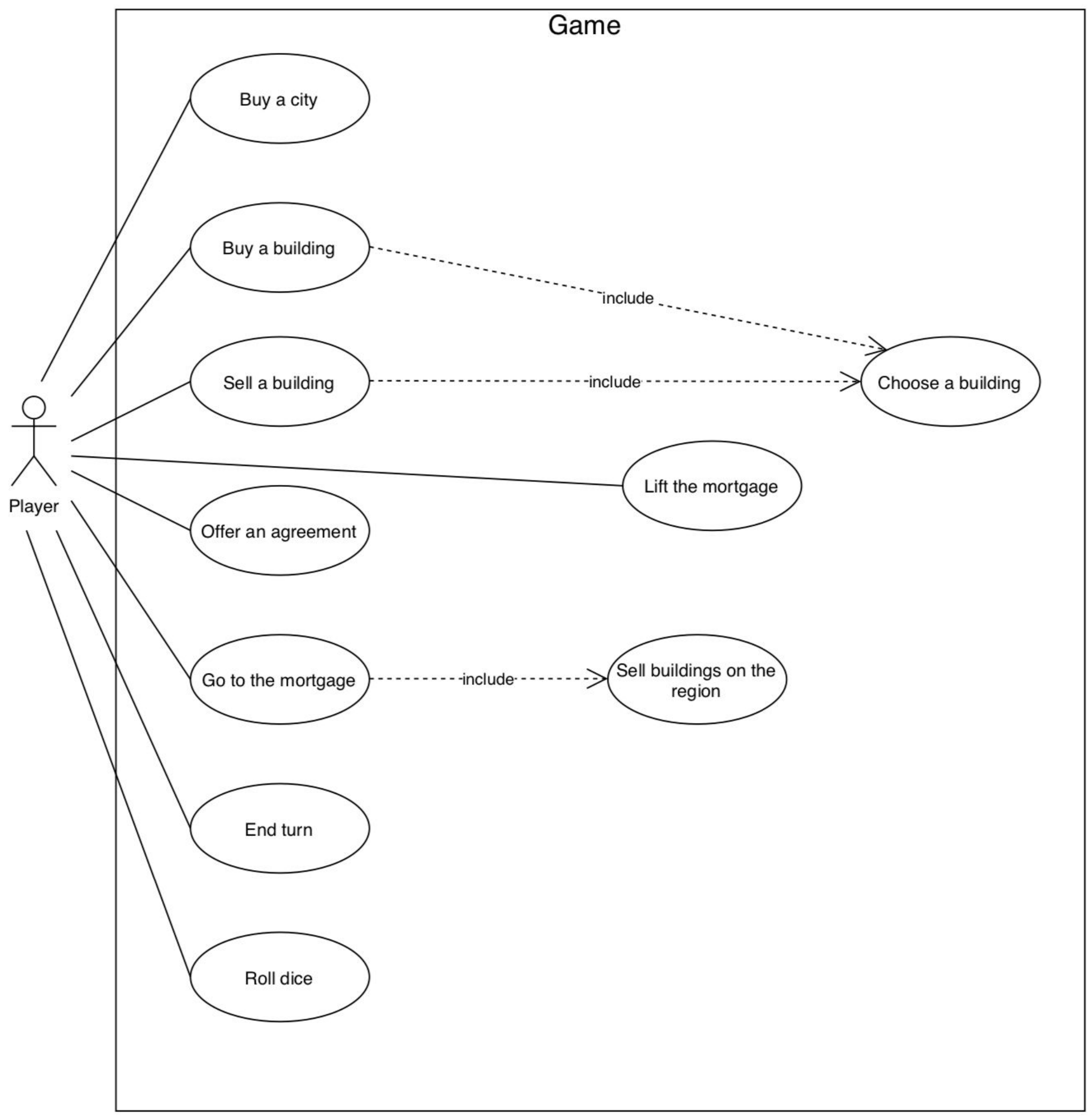
1. Host removes the selected player.

Entry Conditions: Game has not started yet.
Exit Conditions: Selected player is removed from the player list.
Quality Requirements: None

Use Case: Start the game
Participating Actor: Host
Flow of Events:

1. Host starts the game by clicking the start button.

Entry Conditions: There should be at least one player in the player list and the game has not started yet.
Exit Conditions: Game is started by the host.
Quality Requirements: None



Use Case: Buy a city
Participating Actor: Player
Flow of Events:

1. Player click the accept button to buy it.
2. Player gives money to purchase it.
3. City is assigned to the player.

Entry Conditions: Player should have enough money and the city should be available.
Exit Conditions: Player buys the city or rejects to buy it.
Quality Requirements: None

Use Case: Buy a building
Participating Actor: Player
Flow of Events:

1. Player chooses the type of the building.

Entry Conditions: Player should have enough money.
Exit Conditions: Buildings are added to the region.

Quality Requirements: This use case should **include** choose a building use case. Choose building use case is initiated when the user invokes the choose building function so that user can specify which building will be added.

Use Case: Sell a building
Participating Actor: Player
Flow of Events:
1. Money from the selected buildings is added to the player's budget.

Entry Conditions: None
Exit Conditions: Building is demolished.
Quality Requirements: This use case should **include** choose a building use case. Choose building use case is initiated when the user invokes the choose building function so that the user can specify which building will be demolished.

Use Case: Choose a building
Participating Actor: Player
Flow of Events:
1. Player specify the building that will be demolished

Entry Conditions: None
Exit Conditions: Player selects the building.
Quality Requirements: None

Use Case: Lift the mortgage
Participating Actor: Player
Flow of Events:
1. Player pays the money required to lift the mortgage on deed.
2. Deed becomes available for the player.

Entry Conditions: None
Exit Conditions: Mortgage is lifted and deed becomes available.
Quality Requirements: None

Use Case: Offer an agreement with others
Participating Actor: Player
Flow of Events:
1. Player chooses the type of an agreement for requested and offered things.
2. Agreement is made if the other player accepts the trade.

Entry Conditions: Player should have the offers.
Exit Conditions: The offer for the agreement is made.
Quality Requirements: This use case should include ‘Choosing to be offered’ and ‘Choosing to be requested’ so that the player can specify the conditions of the agreement that will be offered.

Use Case: Go to the mortgage
Participating Actor: Player

Flow of Events:

1. Player mortgage the region to the bank.
2. Corresponding amount of money is added to the player's budget.
3. Region becomes unavailable.

Entry Conditions: There should be no buildings on the region.
Exit Conditions: Region becomes unavailable.
Quality Requirements: This use case should **include** sell buildings on the region use case. Sell buildings on the region use case is initiated when the user invokes the function so that the user can mortgage the region.

Use Case: Sell buildings on the region
Participating Actor: Player
Flow of Events:

1. Player sell all buildings in the region.
2. Corresponding amount of money is added to the player's budget.
3. Buildings in the region are demolished.

Entry Conditions: None
Exit Conditions: Buildings in the region are demolished.
Quality Requirements: None

Use Case: End turn
Participating Actor: Player
Flow of Events:

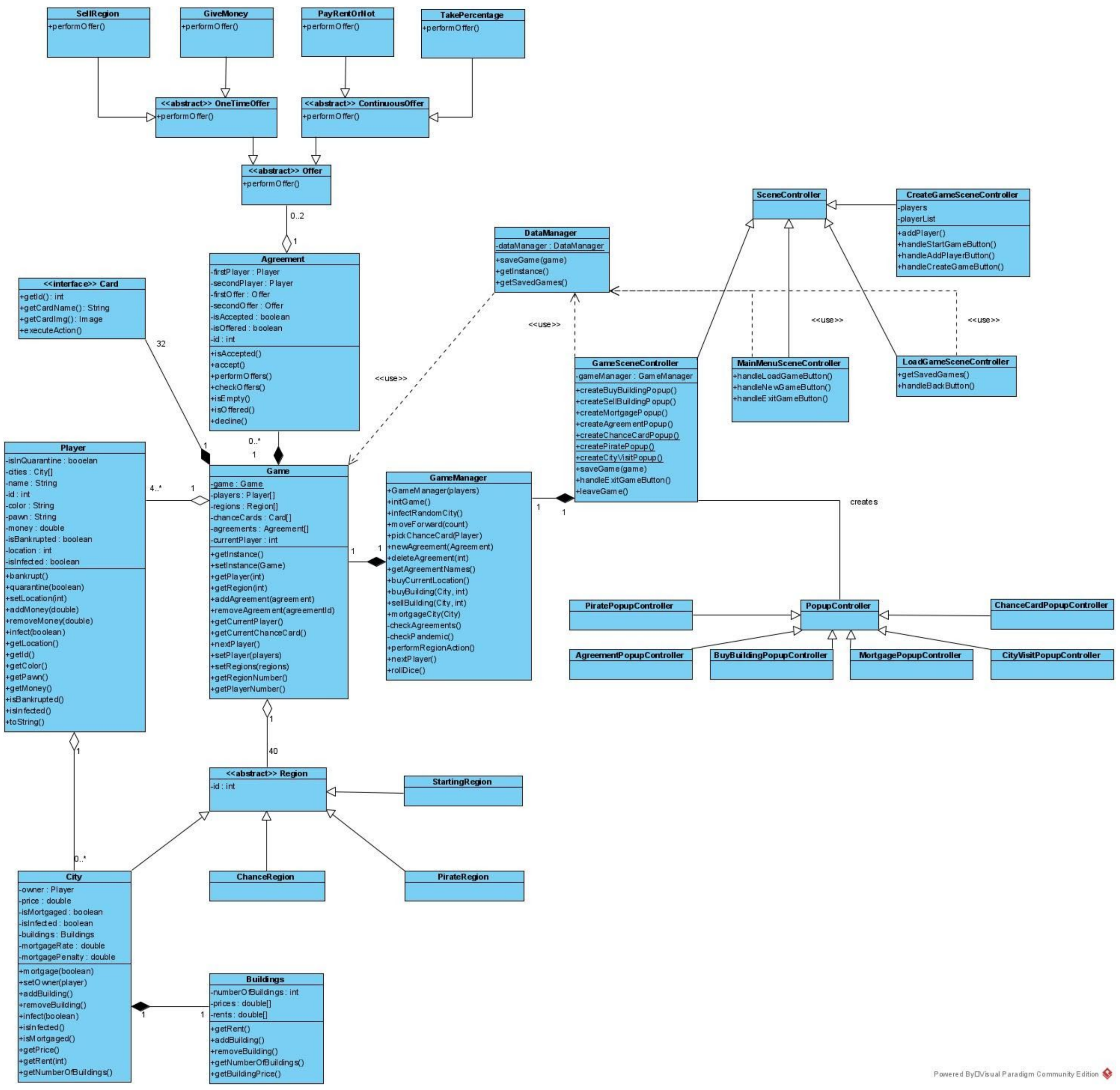
1. Player clicks the end turn button.
2. Turn passes to other player.

Entry Conditions: Player must have rolled dice.
Exit Conditions: Turn passes to other player**.**
Quality Requirements: None
Use Case: Roll a dice
Participating Actor: Player
Flow of Events:

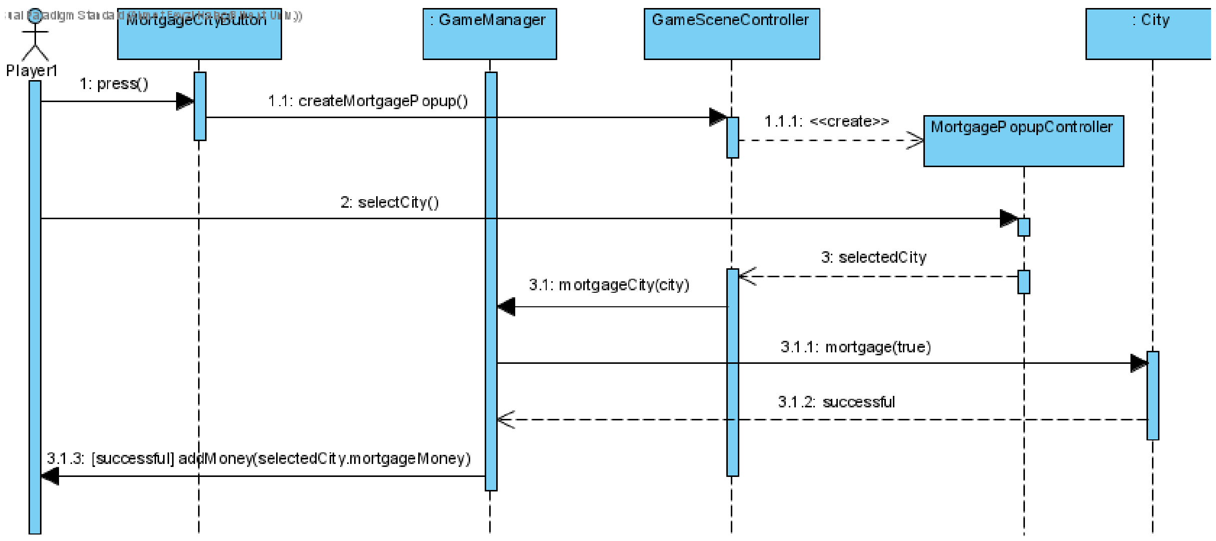
1. Player rolls dice.

Entry Conditions: Player should not be in quarantine and should never have rolled the dice.
Exit Conditions: Dices become unavailable.
Quality Requirements: None

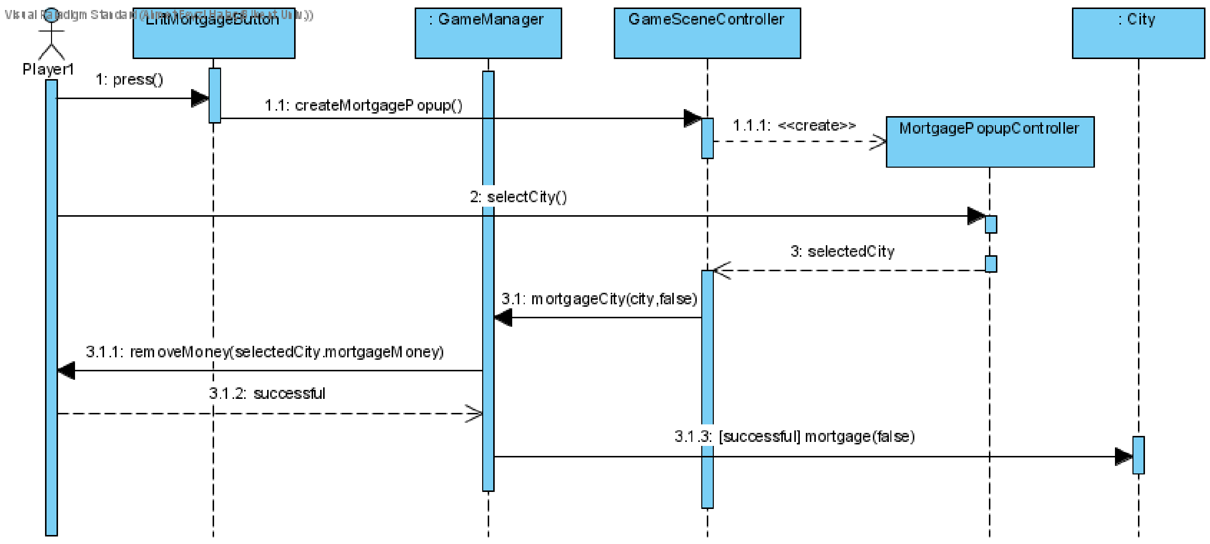
5.2. Class Diagram



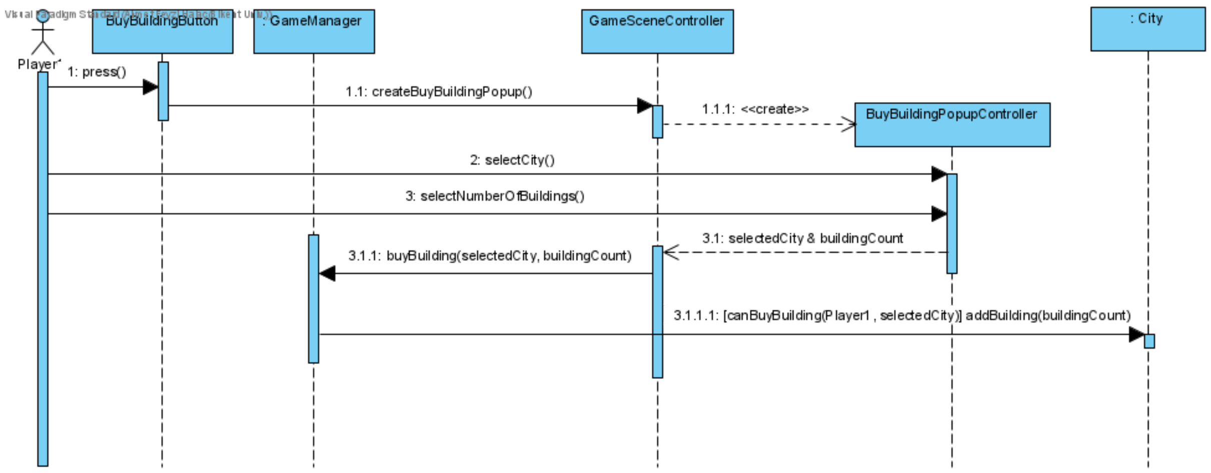
5.3. Sequence Diagrams



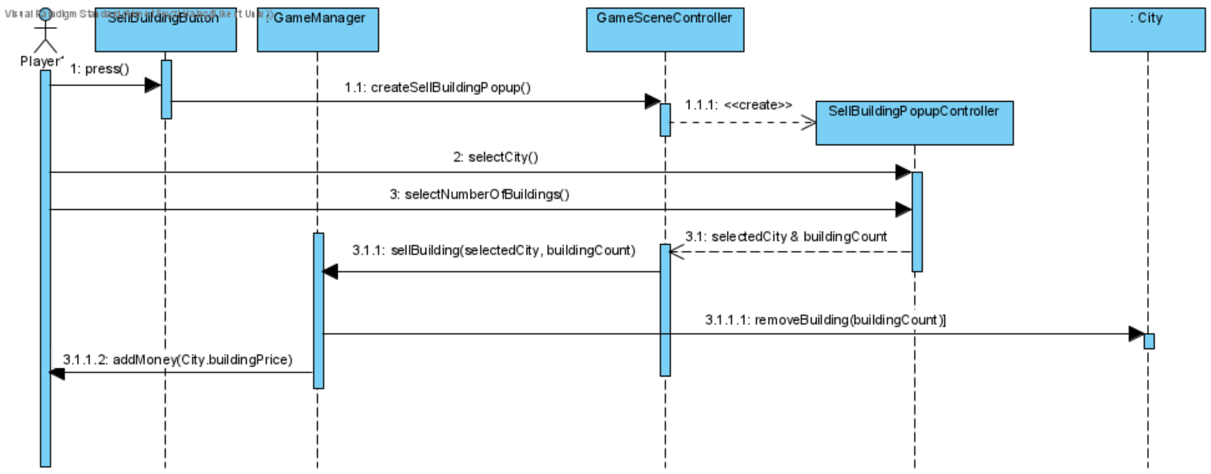
Mortgage City Sequence Diagram



Lift Mortgage Sequence Diagram

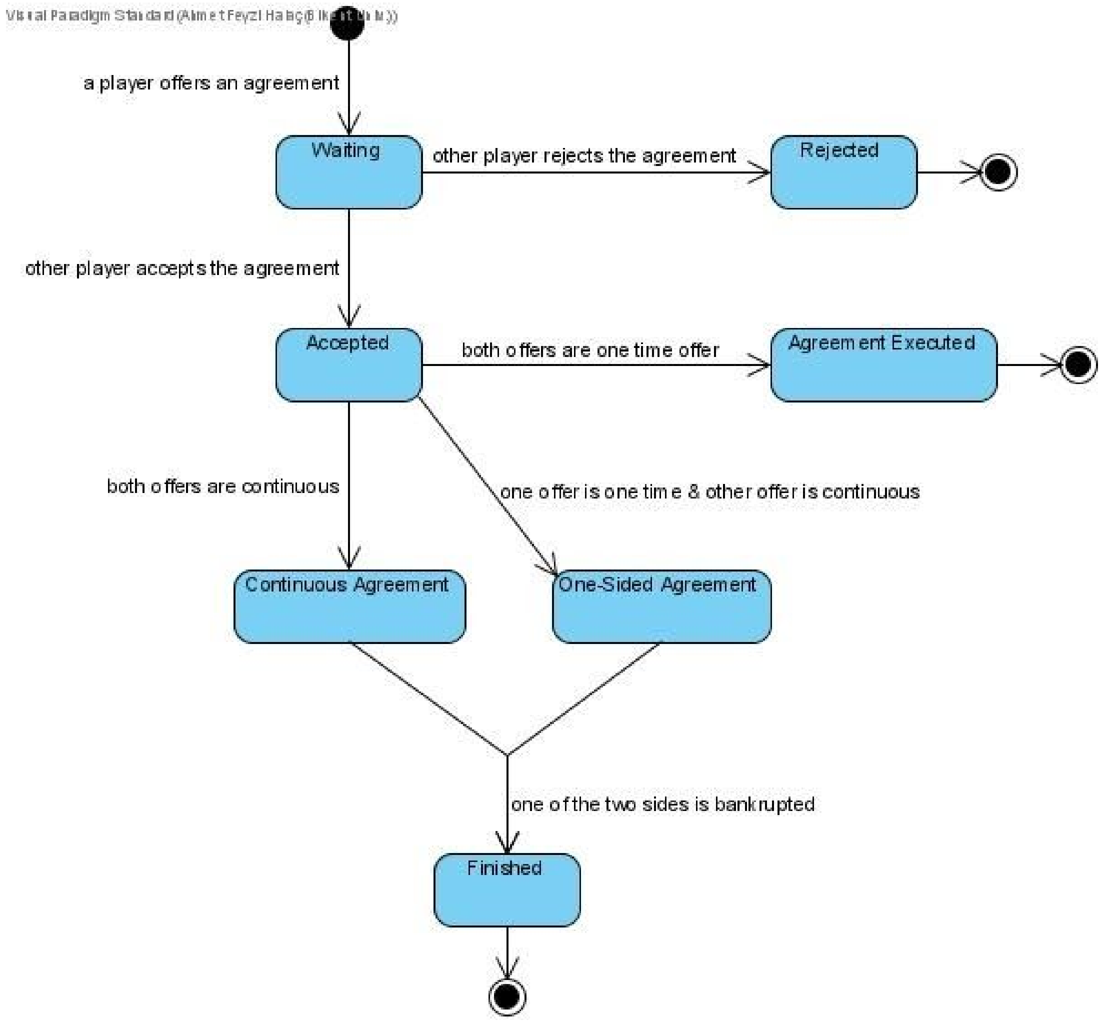

Player
BuyBuildingButton
: GameManager
GameSceneController
: City
1: press()
1.1: createBuyBuildingPopup()
1.1.1: <<create>>
BuyBuildingPopupController
2: selectCity()
3: selectNumberOfBuildings()
3.1: selectedCity & buildingCount
3.1.1: buyBuilding(selectedCity, buildingCount)
3.1.1.1: [canBuyBuilding(Player1, selectedCity)] addBuilding(buildingCount)

Buy Building Sequence Diagram


Player
SellBuildingButton
: GameManager
GameSceneController
: City
1: press()
1.1: createSellBuildingPopup()
1.1.1: <<create>>
SellBuildingPopupController
2: selectCity()
3: selectNumberOfBuildings()
3.1: selectedCity & buildingCount
3.1.1: sellBuilding(selectedCity, buildingCount)
3.1.1.1: removeBuilding(buildingCount)]
3.1.1.2: addMoney(City.buildingPrice)

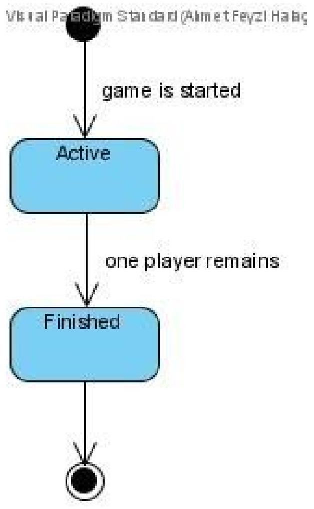
Sell Building Sequence Diagram

5.4. State Diagrams

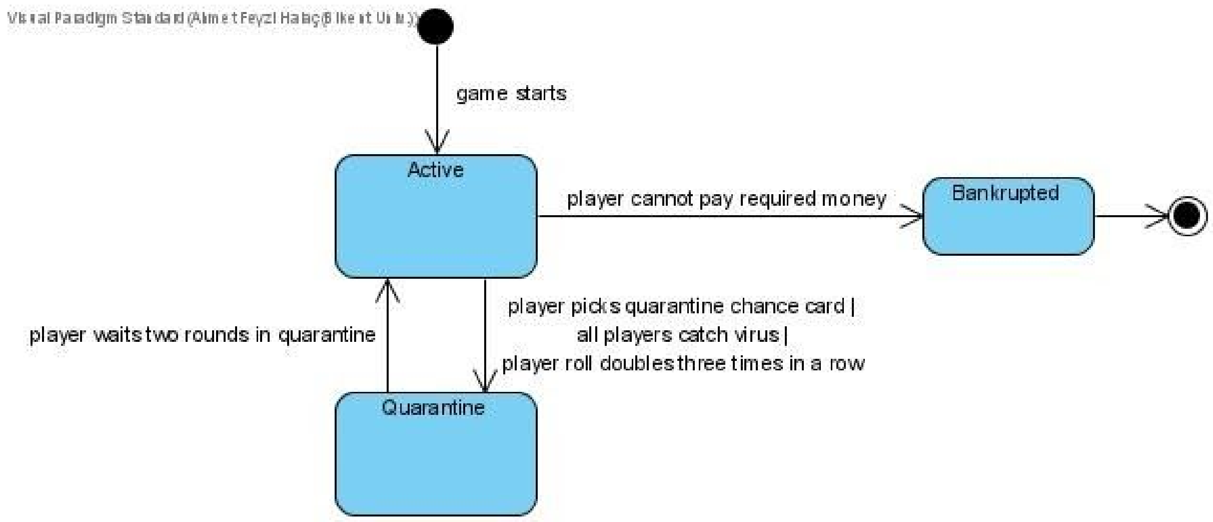


Visual Paradigm Standard (Ahmet Feyzi Halaç (Bilkent Univ.))
a player offers an agreement
Waiting
other player rejects the agreement
Rejected
other player accepts the agreement
Accepted
both offers are one time offer
Agreement Executed
both offers are continuous
one offer is one time & other offer is continuous
Continuous Agreement
One-Sided Agreement
one of the two sides is bankrupted
Finished

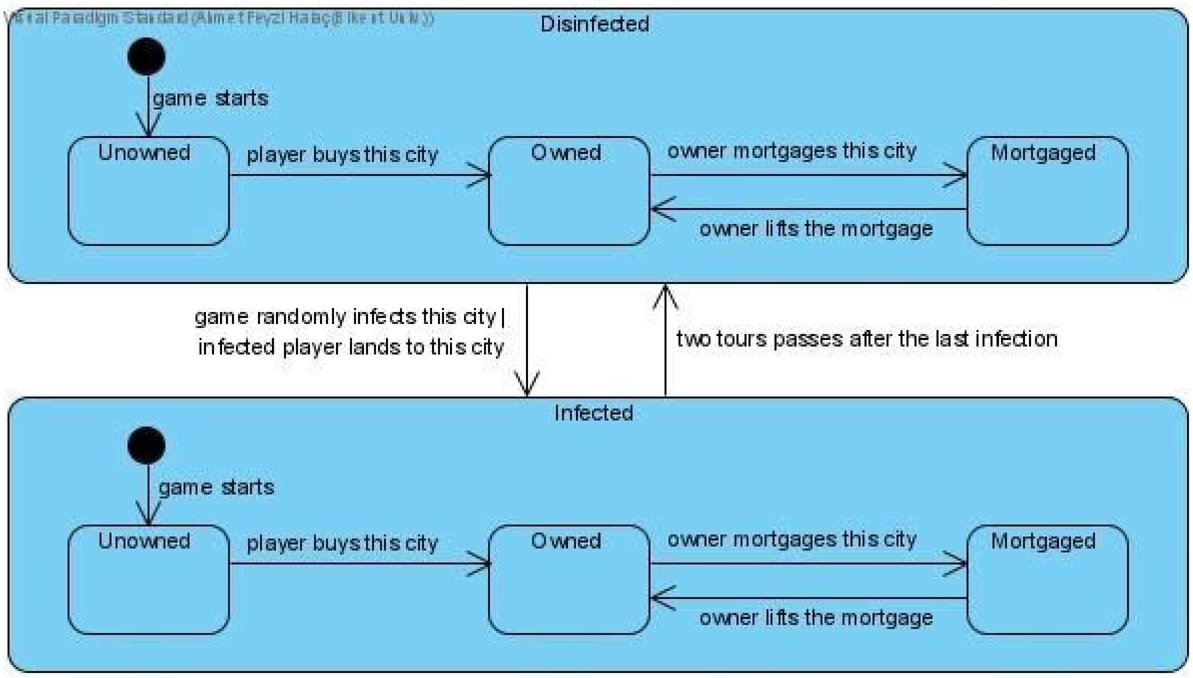
Agreement Class State Diagram



Game Class State Diagram

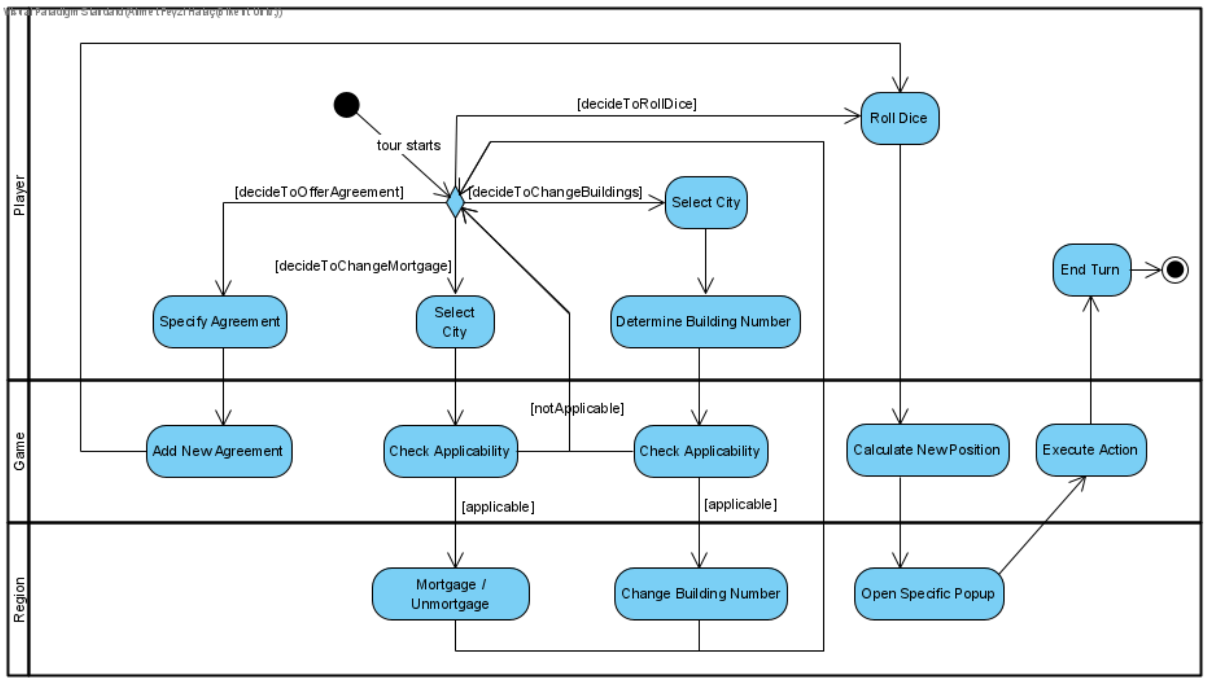

Visual Paradigm Standard (Ahmet Feyzi Halaç(Bilkent Univ.))
game starts
Active
player cannot pay required money
Bankrupted
player picks quarantine chance card |
all players catch virus |
player roll doubles three times in a row
player waits two rounds in quarantine
Quarantine

Player Class State Diagram

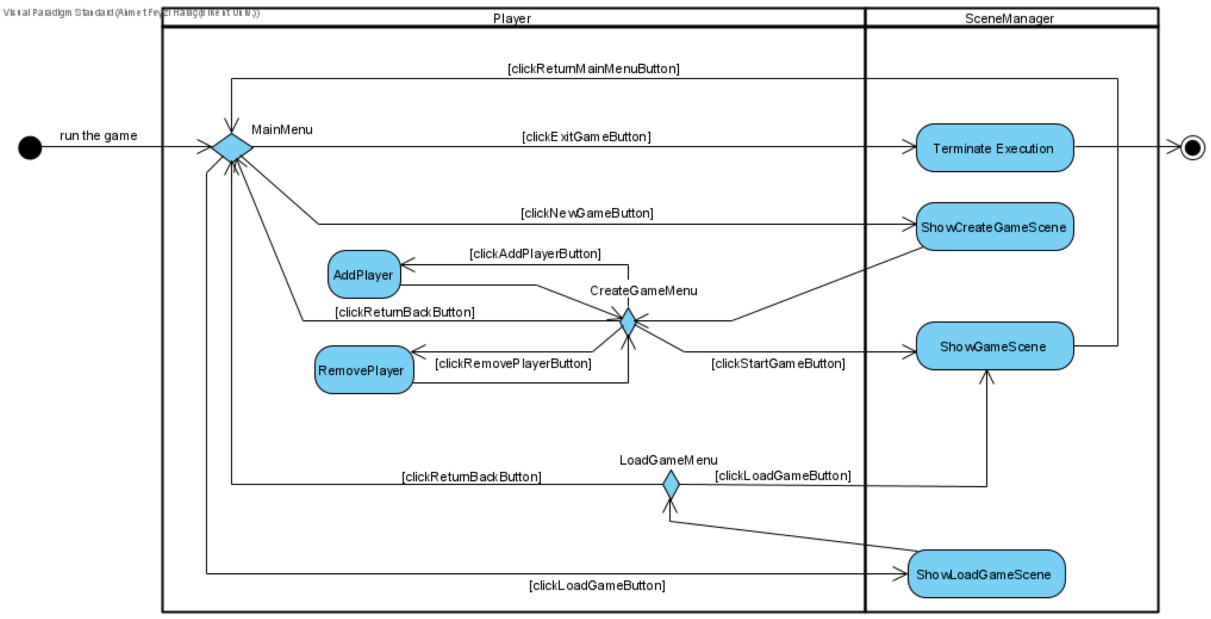

Visual Paradigm Standard (Ahmet Feyzi Halaç(Bilkent Univ.))
Disinfected
game starts
Unowned
player buys this city
Owned
owner mortgages this city
Mortgaged
owner lifts the mortgage
game randomly infects this city |
infected player lands to this city
two tours passes after the last infection
Infected
game starts
Unowned
player buys this city
Owned
owner mortgages this city
Mortgaged
owner lifts the mortgage

City Class State Diagram

5.5. Activity Diagrams



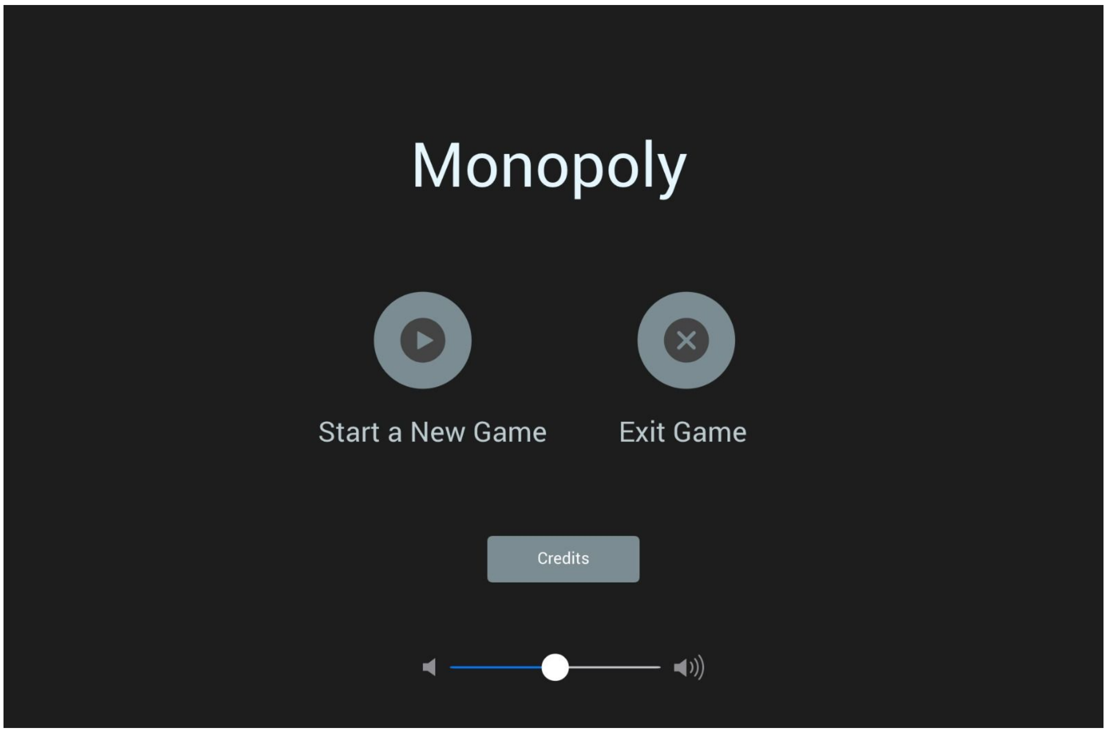
Activity Diagram of One Turn of a Player



Activity Diagram of Navigation Between Scenes

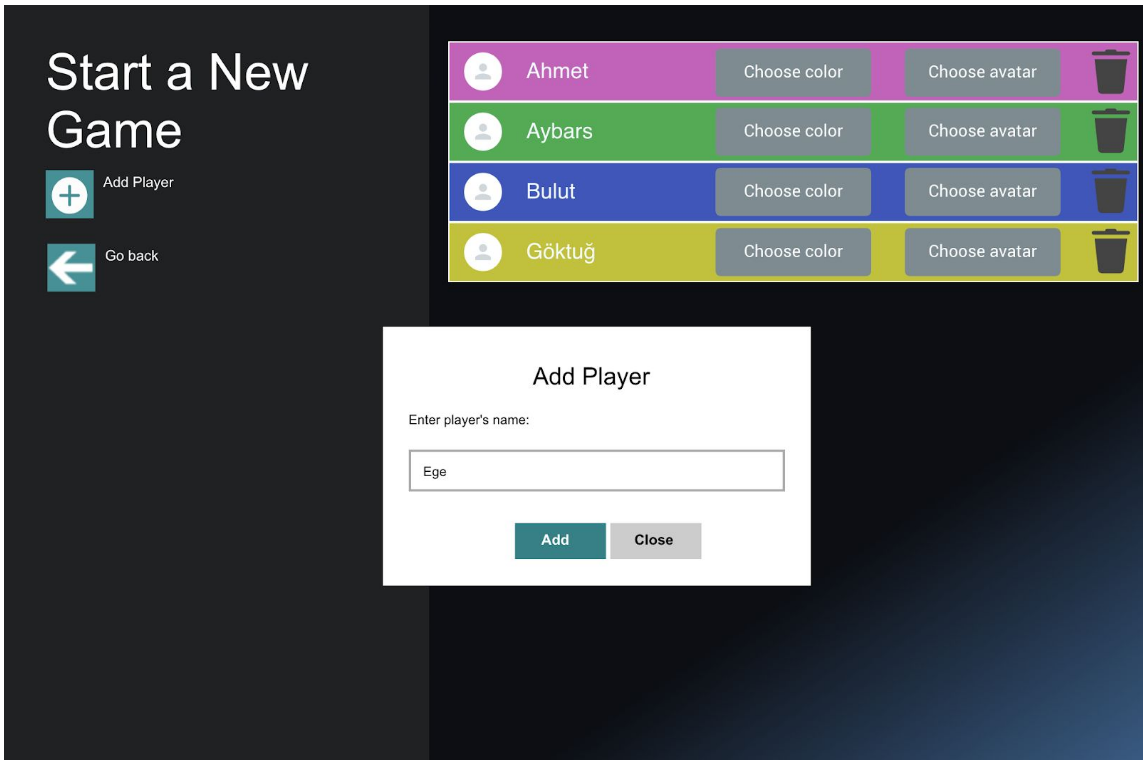
6. User Interface

Main menu:



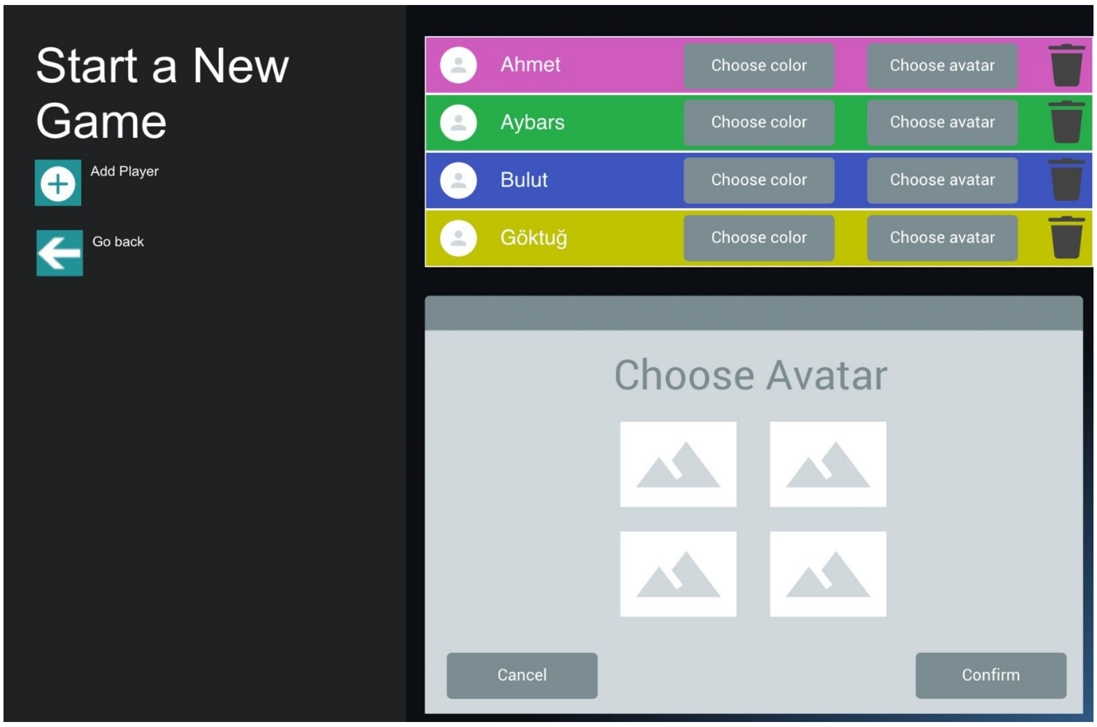
Creating a game player addition:

When the "Add Player" button is clicked, a pop-up window shows up and waits for input.



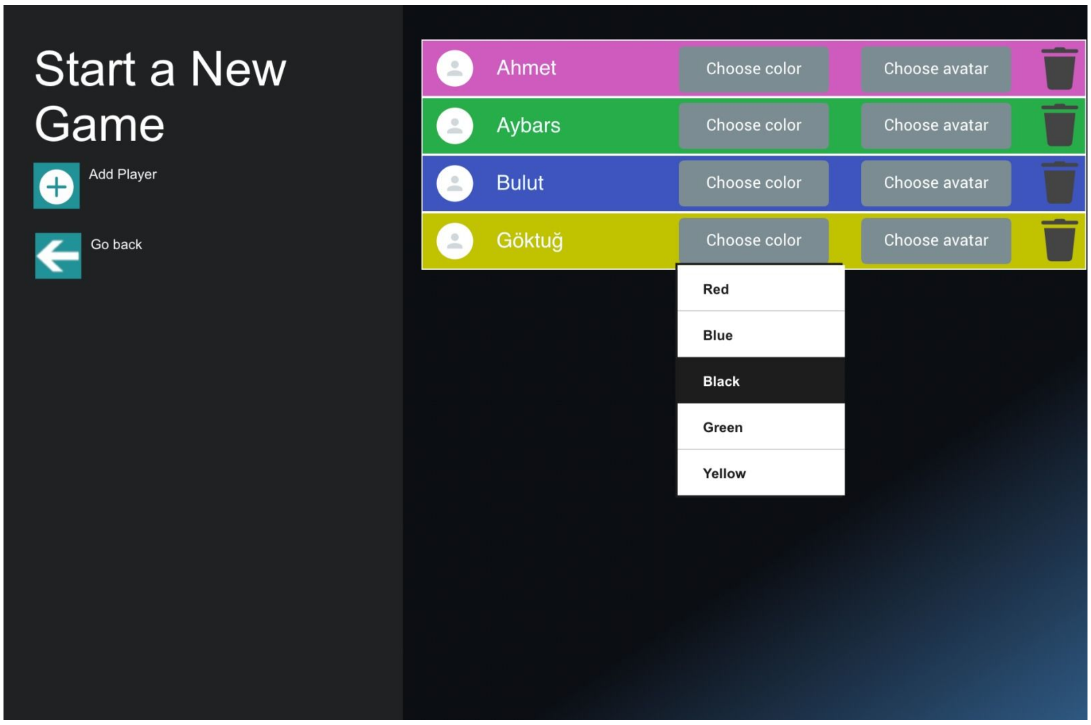
Creating a game avatar selection:

When the choose avatar button is selected, a player can choose its icon from available characters in the game.



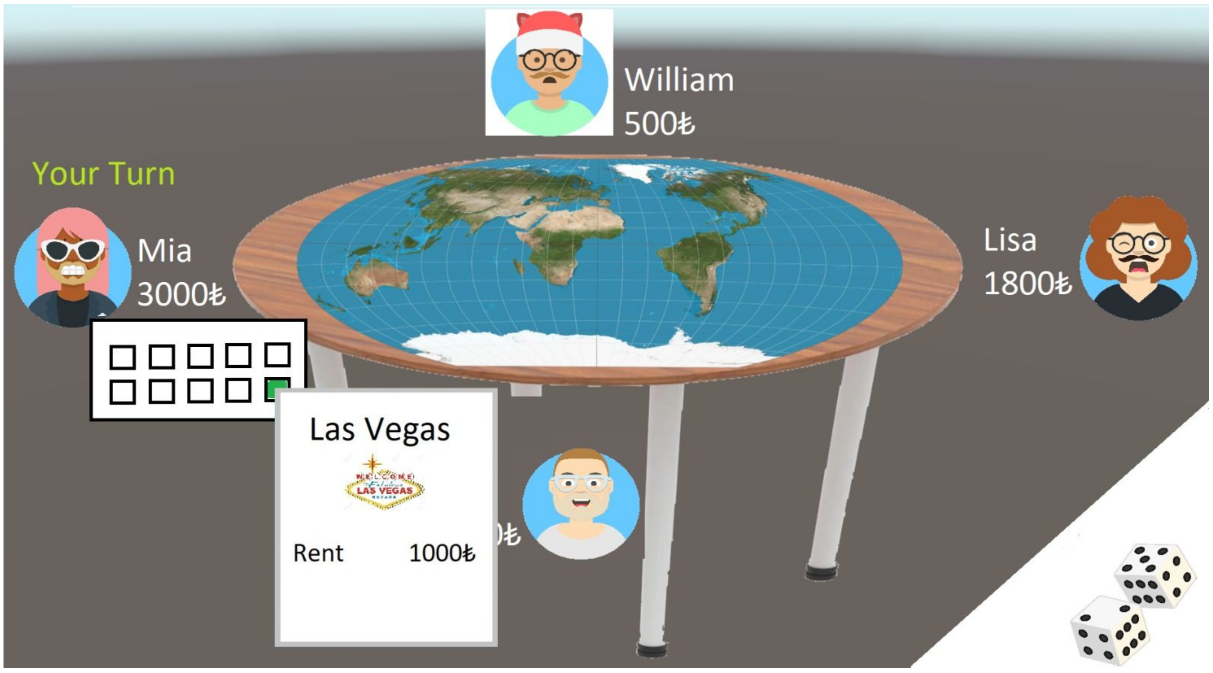
Creating a game color selection:

When choose color button is pressed, a dropdown menu shows up and color can be selected from that menu.



In-game interface:

During the game, a player's properties can be seen by placing the mouse on top of the player. In each turn, there will be a "Your Turn" message at one player's upper section. In the bottom right corner, there will be a dice icon that rolls dice at each turn.


William
500₺
Your Turn
Mia
3000₺
Lisa
1800₺
Las Vegas
LAS VEGAS
Rent
1000₺