BILKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CS319

Iteration 1

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. Functional Requirements

2.1. Starting a Game

When the player first opens the application, the main menu screen will be displayed. At this menu, there are two options. One of them is starting a new game and the other one is quitting the game. There is also an adjustment setting for volume before the player starts the game.

If the player chooses a new game option the screen changes to a different one in which the player can adjust the game settings such as

- Adding new players.
- Determining each players’ name, pawn, and color.
- Removing a player.

After these adjustments, the game starts and previous adjustments cannot be changed during the game.

2.2. Starting State of a Game

After all the adjustments, the game starts. In the beginning, all the players are in the start region with the same amount of money. The amount of money is fixed in every game. In order to determine which player will start first, each player rolls dice and players start the game in descending order of their number (if two players rolled the same, they roll against each other until the tie breaks). After the order is determined, the game starts.

The shape of the map and the route on it is fixed however, the infected region on the map is randomly chosen in each game so the starting state of the game can be different from one another (infected region will be explained).

2.3. One Turn in a Game

In each turn, players (according to starting order) roll dice and pass some number of regions equal to the number they rolled. After they move, they will follow the game state options and act according to them 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.

2.4. Map and Regions

The world map is used for this games’ map as a diagram. The world map is split into regions and players follow a determined path on this map. There are several types of regions. Some of them can be purchased by players, some of them made players do some duties and some of them made players play mini-games.

2.4.1. Purchasable Regions (Cities)

When players land on this type of region and if that region was not bought by some other player, they can buy it. The price of each region is written on the map under the region's name.

If the region belongs to some other player, when other players land in that region they have to pay rent. How much they must pay is written on the region’s card. The cost of the rent can be upgraded by additional buildings that were built in that region.

Purchasable regions are divided into groups. Each group has a different color. Groups can have three or two regions.

2.4.2. Chance Regions

There are several chance regions on the map that cannot be bought by players. When players land in that region, they draw a chance card. This card can either command the player to do something immediately or it can be a card which player can hold and use later.

2.4.3. 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.

Players can also get infection from chance cards, by landing chance regions. If a player gets infection from a chance card, he/she directly goes to the quarantine region and stays there for three turns.

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 were there, they will also leave the quarantine region with others and pay the fee.

2.4.4. Start Region

This is the region where the players start the game. Each time a player passes this region they will get some money from the bank.

2.4.5. Quarantine Region

This is a normal region which is included the path of the game. If players come here and stop it has no effect. If players come here because of infection or chance card, they must obey the explained rules. Players who rolled double three times in a row also become infected and come to this region and wait for three turns.

2.4.6. Mini-Game Regions

There are three regions on the map where players play mini-games against the computer. Each of those regions has a different game. Players must play that game if they land in that region. At the end of this mini-game, they can earn or lose money.

2.4.6.1. Las Vegas

If players come to that region, they roll dice against the computer. If the players’ number is higher than computer’s, then they earn 10 times of their number from the bank. If the players’ number is less, then they pay to the bank 10 times of their number.

2.4.6.2. Italy

When players land on that region a mini-game screen will pop up. On that screen, there is pizza dough and ingredients of the pizza are falling on that pizza (such as cheese). However, those ingredients are mixed with some harmful materials. Player must click on those harmful materials while they are falling to separate them. If the player is successful, he/she gets money from the bank, otherwise, the player will pay money to the bank.

2.4.6.3. Spain

When players land on that region a mini-game screen will pop up. On that screen, a human image is standing on the road and there are bulls running on the road towards the human image. Player will try to dodge those bulls by controlling the human image. If the player is successful, he/she gets money from the bank, otherwise, the player will pay money to the bank.

2.5. Game state

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

2.5.1. Roll Dice

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

2.5.2. Buy a Region

This option is available only if the player is on a purchasable region (city). If they have enough money, after this operation the card of the region will be added to their hand. If the player's money is not enough this operation cannot be done.

2.5.3. Buy a Building

When a player has the whole group of regions, he/she can buy buildings to those regions. Buildings increase the rent of the region. When other players come to that region, they pay according to buildings on it. Each region can have one to four houses or one special building. Cost of each building is written on the region's card.

2.5.4. Sell a Building

Players can sell buildings to the same amount of money they construct them. When this option is chosen players can select which buildings to sell.

2.5.5. 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 on their turns.

2.5.5.1. Sell a Region

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

2.5.5.2. 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.

2.5.5.3. Give Money

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

2.5.5.4. Pay a Rent or Not

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

2.5.6. Go to the Mortgage

Bank can hold a mortgage on a region if the player cannot pay his/her debt. Every region has a mortgage value. When a region is mortgaged, the owner of that region receives the mortgage value from the bank. In order to mortgage a region, the owner must sell the buildings on that region first (if any exist). After that he/she can go to mortgage. If a region is mortgaged, the owner cannot collect any rent from that region. Mortgaged regions can also be sold to other players.

2.5.7. Lift the Mortgage

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

2.5.8. End Turn

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

2.6. Bankruptcy and End of the Game

If a player has no way to pay his/her debts he/she goes bankrupt. When a player goes bankrupt, they lose and leave the game. The last player that stays on the game wins the game.

3. Non-functional Requirements

- User Experience
  - In-game buttons should be big enough so that users can click buttons with the mouse easily.

- Performance and Graphics
  - The window resolution of the game will be 1280x800.
  - The latency of the game after each input must be less than 0.1s.
  - The framerate of the game must be always 30.

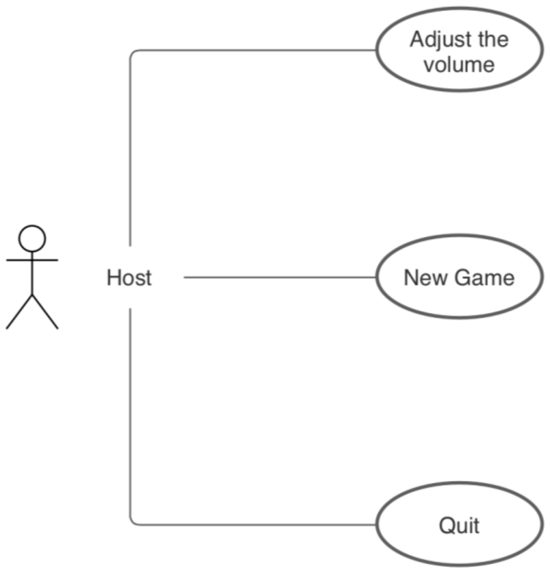
- Compatibility
  - The game will be compatible with Windows 10.

- Pseudo requirements (Constraints) :
  - The game will be written using the Java programming language.
  - The game will be a desktop application in windowed mode.
  - The button count on each window must not pass beyond 7.

4. System Models

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

4.1. Use Case Diagrams



Main Menu State

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.

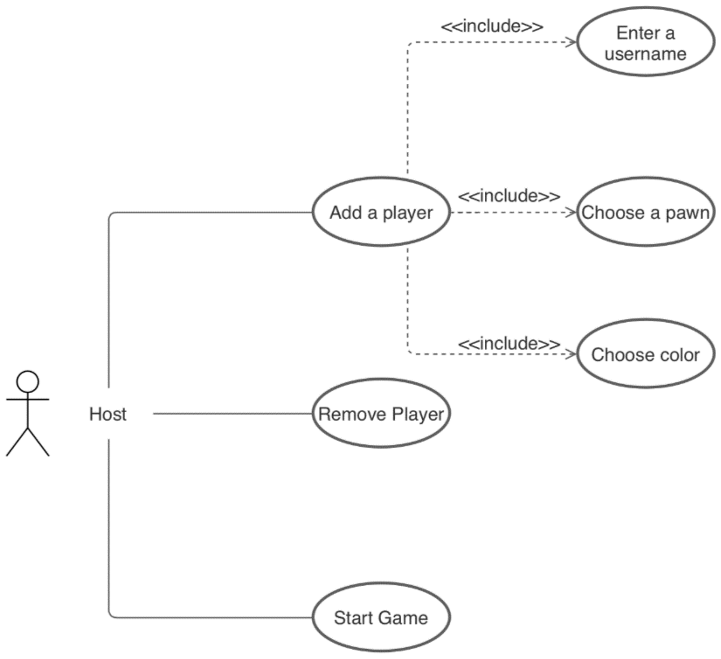
Entry Conditions: Game has not started yet.
Exit Conditions: Volume is adjusted by the host.
Quality Requirements: None

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



Starting Game State

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:

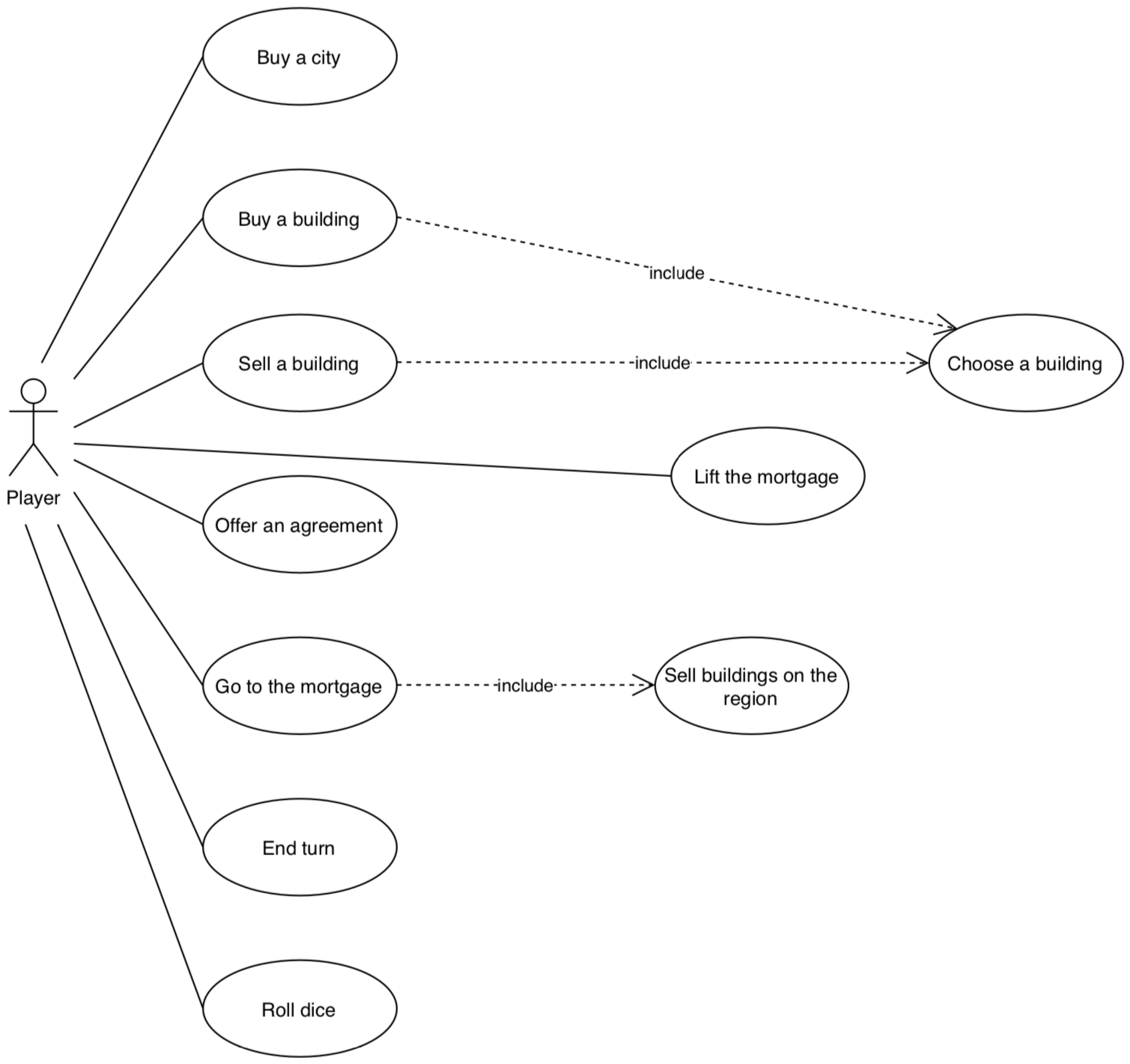
1. Host removes the selected player.

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



Game State

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

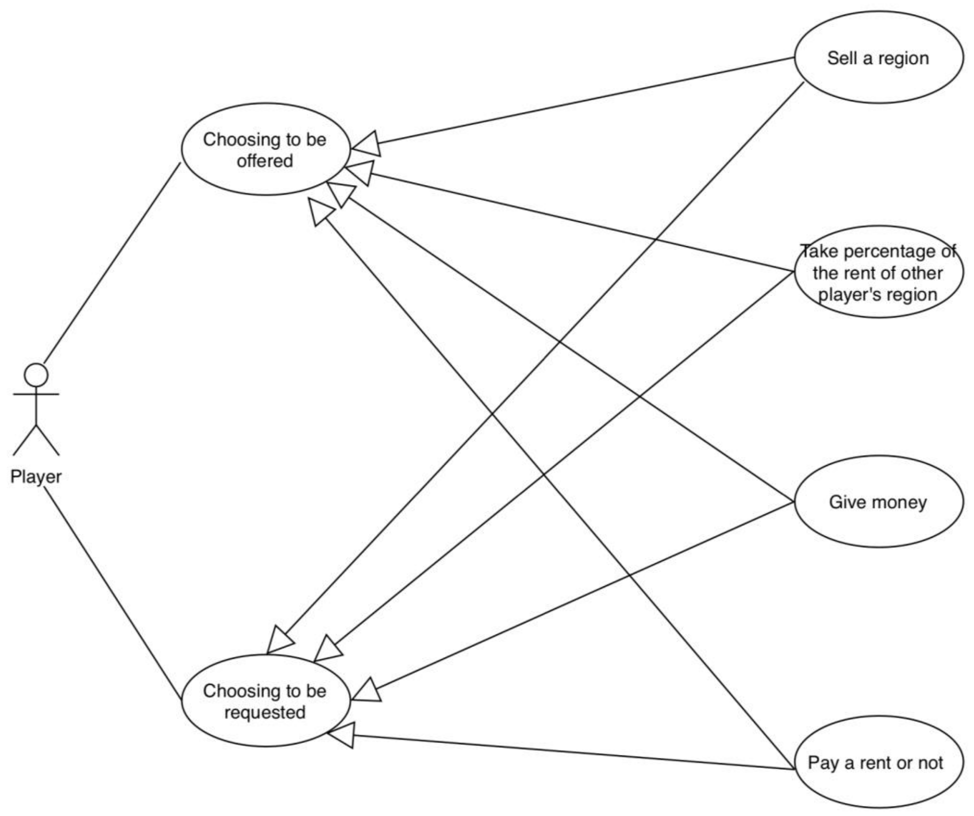
Use Case: End turn
Participating Actor: Player
Flow of Events:

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



Offer an Agreement State

Use Case: Choosing to be offered
Participating Actor: Player
Flow of Events:

1. Player chooses the type of an agreement that will be offered.

Entry Conditions: None.
Exit Conditions: The offer is made.
Quality Requirements: None

Use Case: Choosing to be requested
Participating Actor: Player
Flow of Events:

1. Player chooses the type of an agreement that will be requested.

Entry Conditions: None.
Exit Conditions: Request is made.
Quality Requirements: None

Use Case: Sell a region
Participating Actor: Inherited from Choosing to be offered and requested use cases.
Flow of Events:

1. Player specifies the regions that will be sold.
2. Corresponding offer for the selected region(s) will be implemented to the player's account.

3. Region is assigned to other player.

Entry Conditions: Inherited from Choosing to be offered and requested use cases.
Exit Conditions: Inherited from Choosing to be offered and requested use cases.
Quality Requirements: Both sides should accept the agreement.

Use Case: Give percentage of the rent of any player's region
Participating Actor: Inherited from Choosing to be offered and requested use cases.
Flow of Events:

1. Player specifies the percentage of the rent of the other player's region.
2. Corresponding offer will be implemented to the player's account.
3. Specified percentage of the region is assigned to the player.
4. Unsold percentage of the region remains in the traded player.

Entry Conditions: Inherited from Choosing to be offered and requested use cases.
Exit Conditions: Inherited from Choosing to be offered and requested use cases.
Quality Requirements: None

Use Case: Give money
Participating Actor: Inherited from Choosing to be offered and requested use cases.
Flow of Events:

1. Player specifies the amount of money that will be given.
2. Corresponding offer for the selected region(s) will be implemented to the player's account.
3. Specified amounts of money are added to other player.
4. Specified amount of money is deducted from the player's budget.

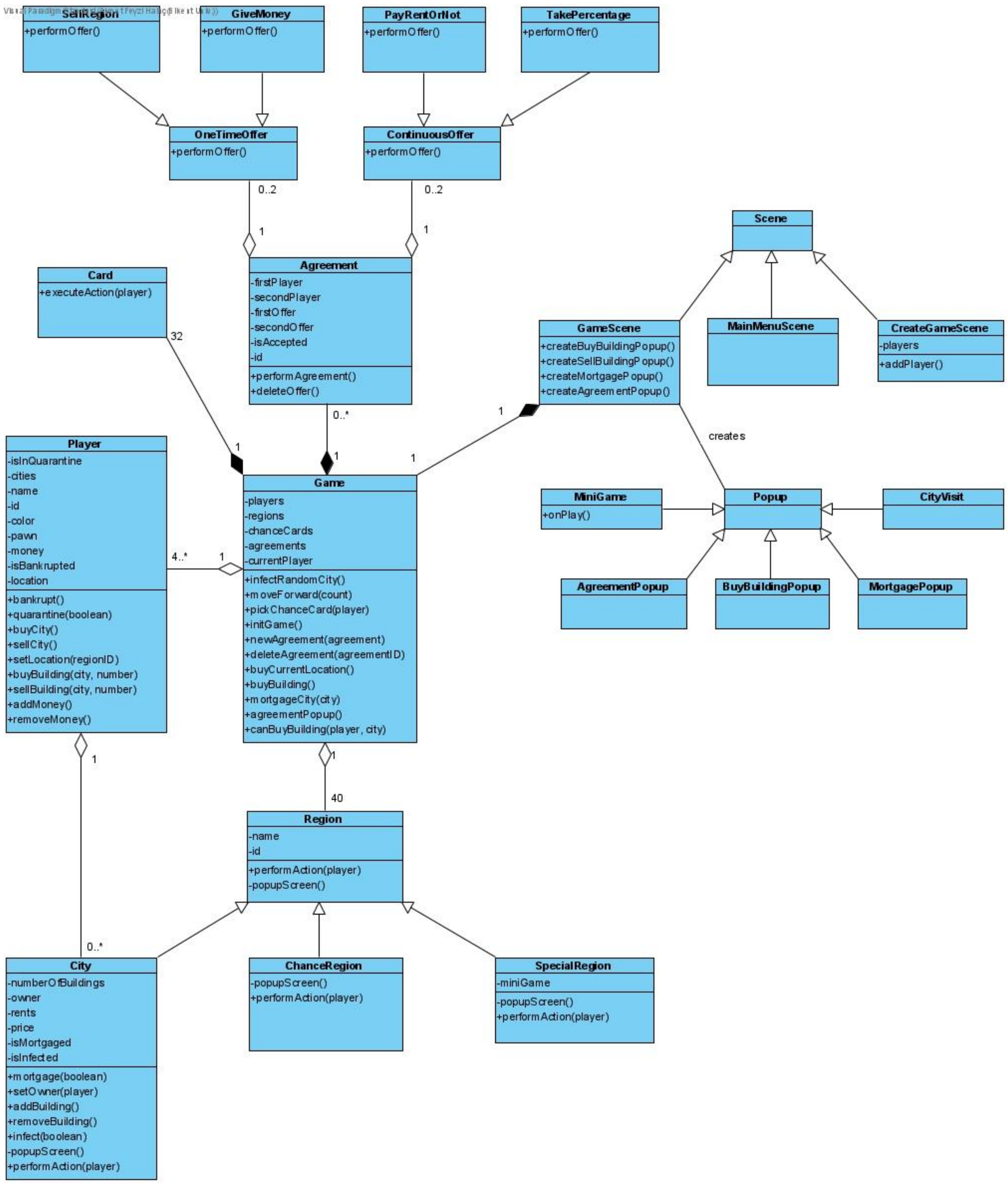
Entry Conditions: Inherited from Choosing to be offered and requested use cases.
Exit Conditions: Inherited from Choosing to be offered and requested use cases.
Quality Requirements: None

Use Case: Pay a rent or not
Participating Actor: Inherited from Choosing to be offered and requested use cases.
Flow of Events:

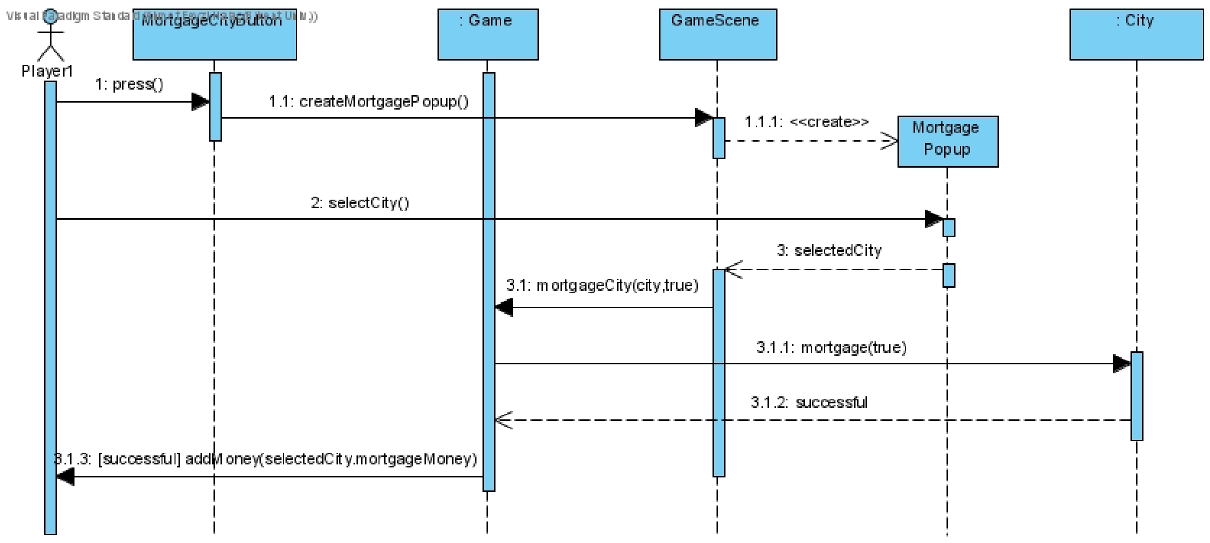
1. Player decides whether the player in the region will pay a rent or not to the owner.

Entry Conditions: Inherited from Choosing to be offered and requested use cases.
Exit Conditions: Inherited from Choosing to be offered and requested use cases.
Quality Requirements: None

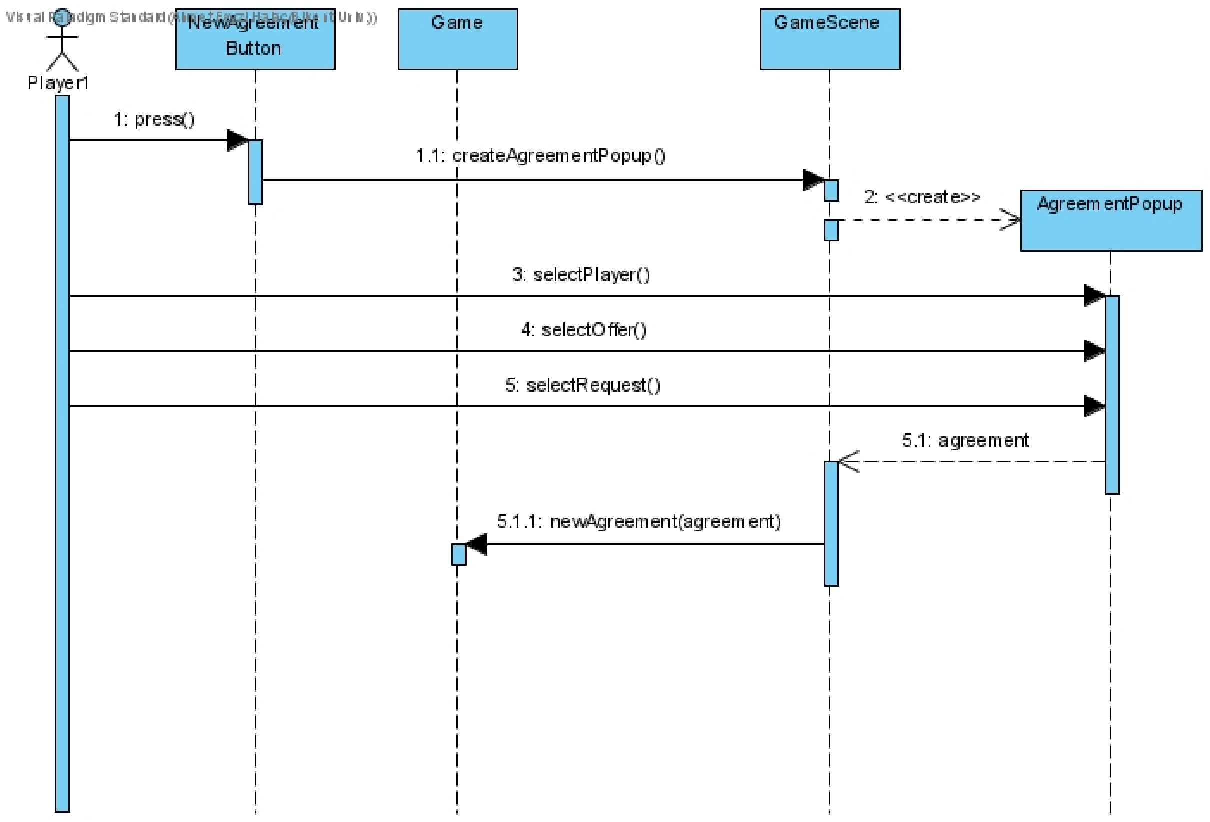
4.2. Class Diagram



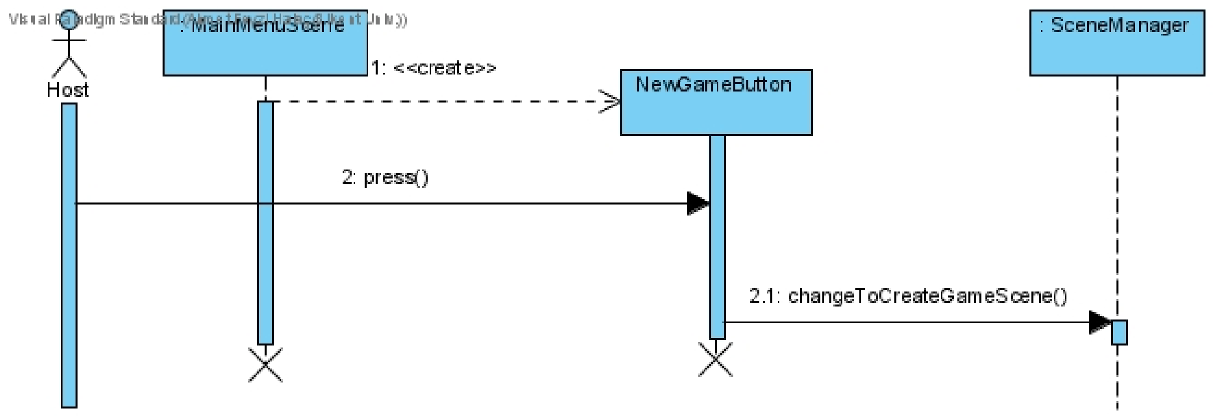
4.3. Sequence Diagrams



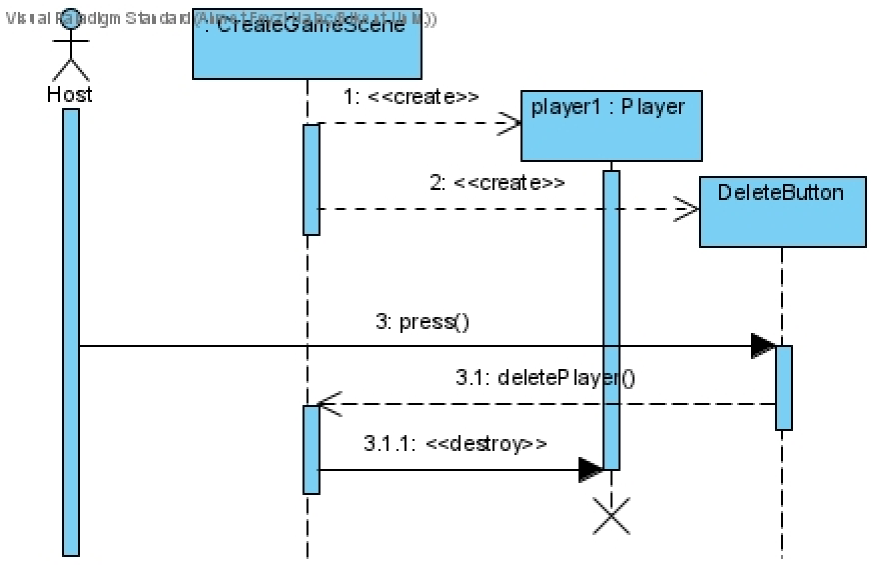
Mortgage City Sequence Diagram



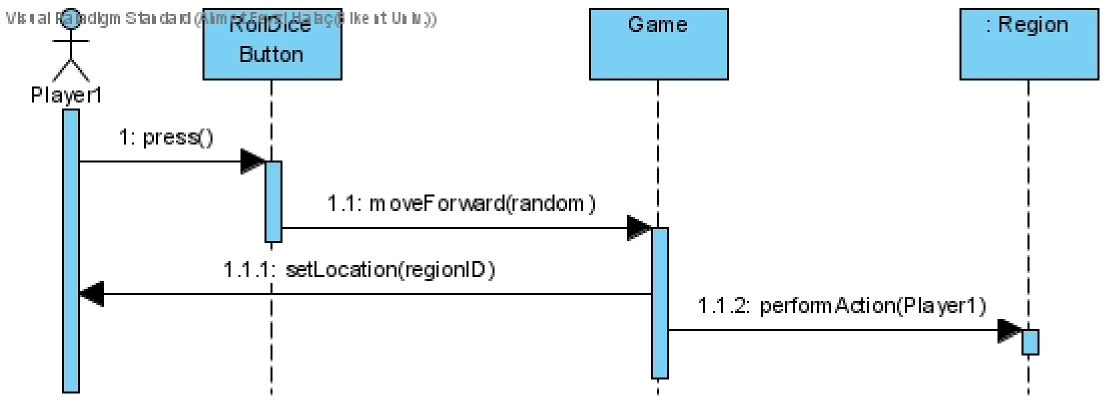
New Agreement Sequence Diagram



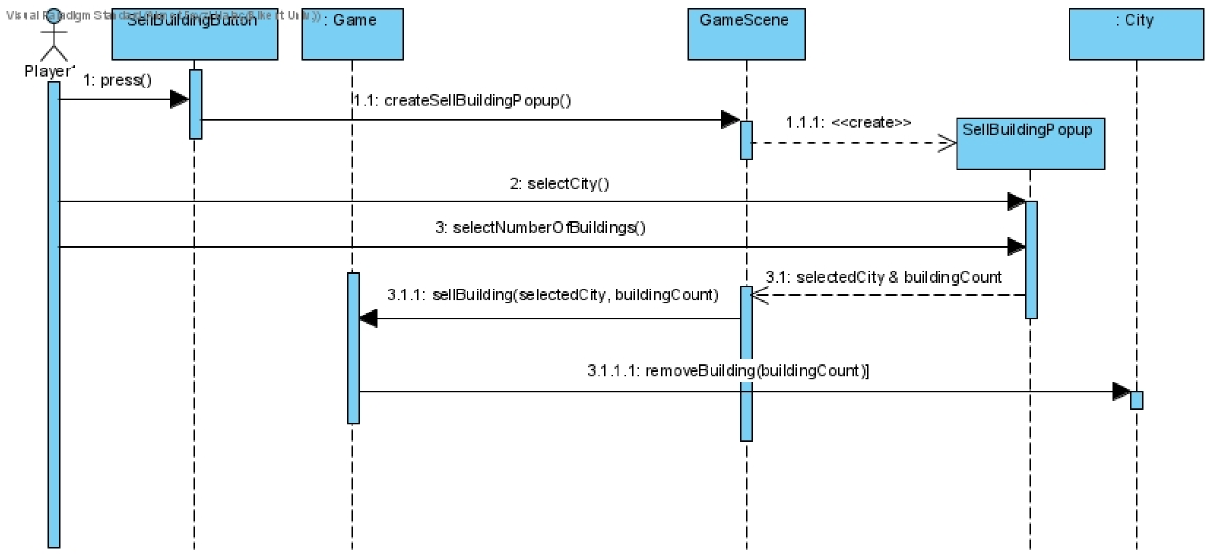
New Game Sequence Diagram



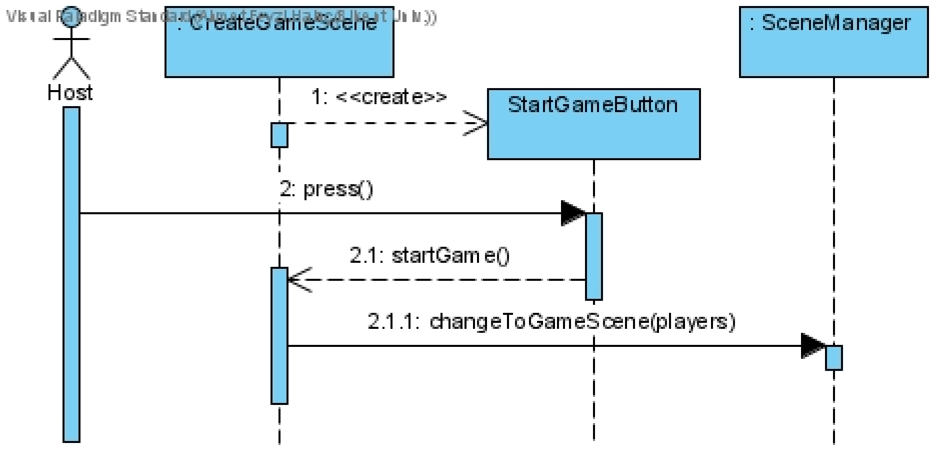
Remove Player Sequence Diagram



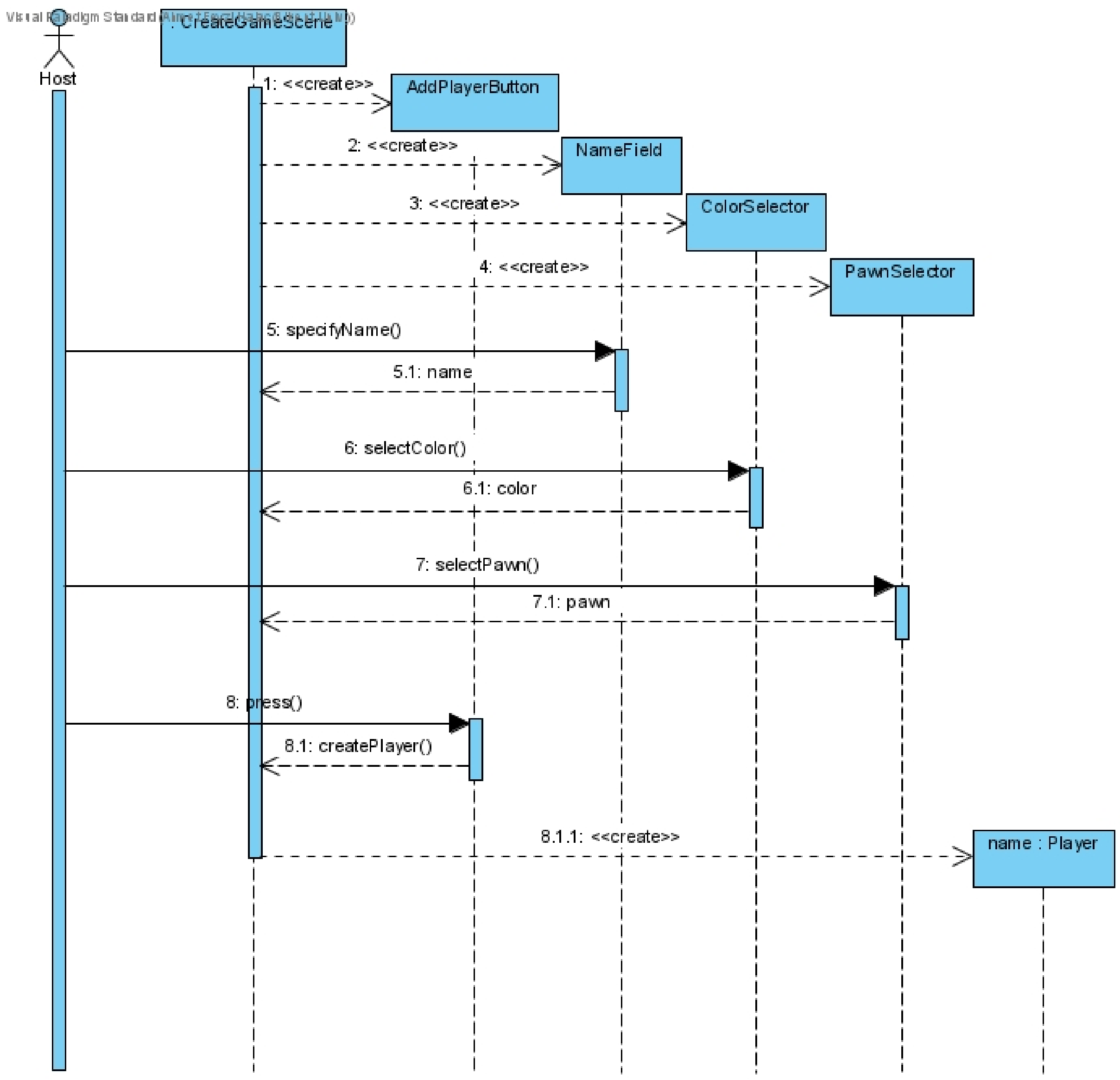
Roll Dice Sequence Diagram


Player
SellBuildingButton
: Game
GameScene
: City
1: press()
1.1: createSellBuildingPopup()
1.1.1: <<create>>
SellBuildingPopup
2: selectCity()
3: selectNumberOfBuildings()
3.1: selectedCity & buildingCount
3.1.1: sellBuilding(selectedCity, buildingCount)
3.1.1.1: removeBuilding(buildingCount)]

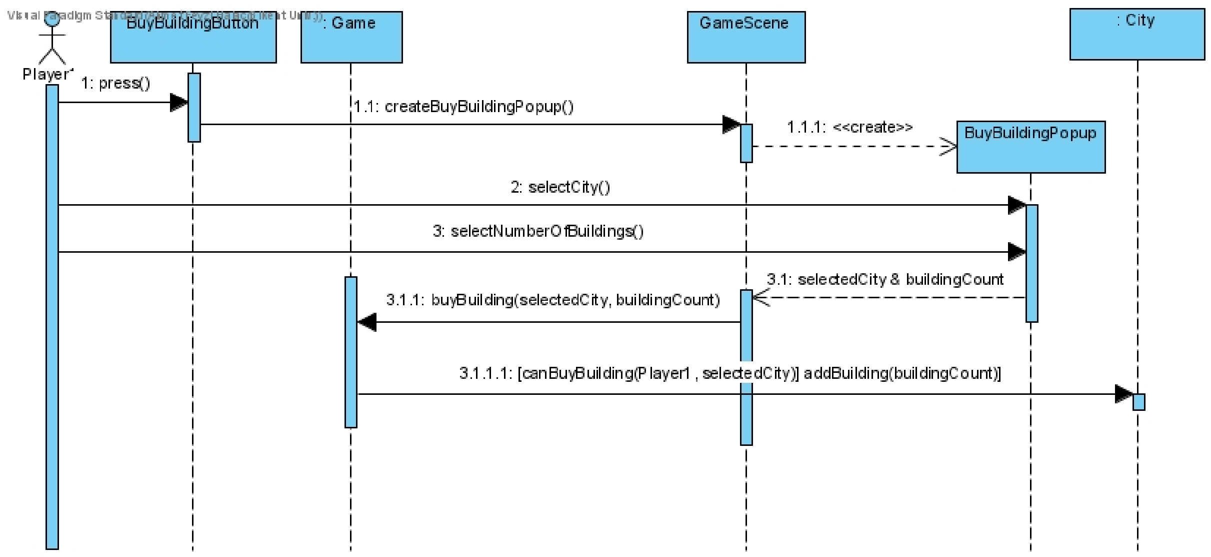
Sell Building Sequence Diagram



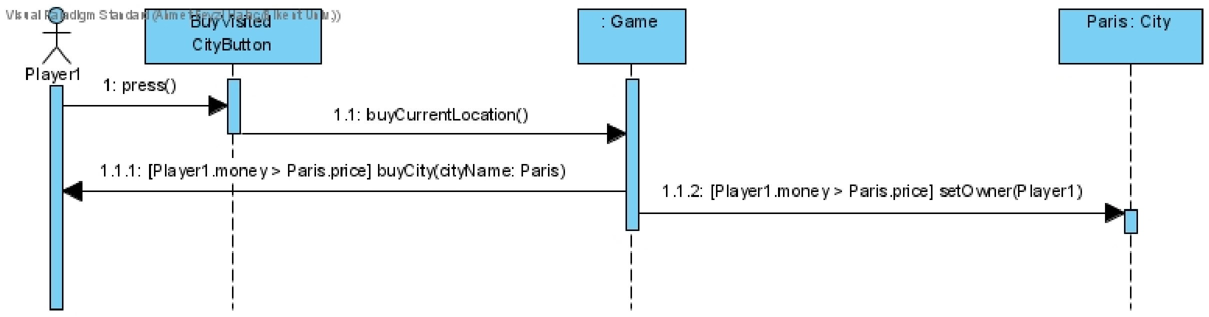
Start Game Sequence Diagram



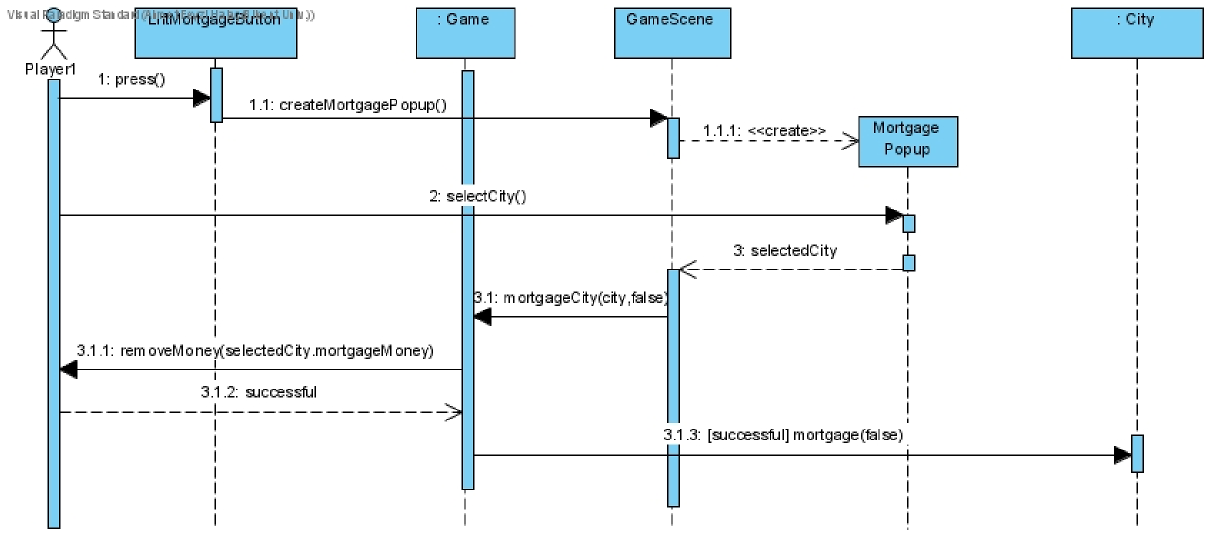
Add Player Sequence Diagram


Player1
BuyBuildingButton
: Game
GameScene
: City
1: press()
1.1: createBuyBuildingPopup()
1.1.1: <<create>>
BuyBuildingPopup
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

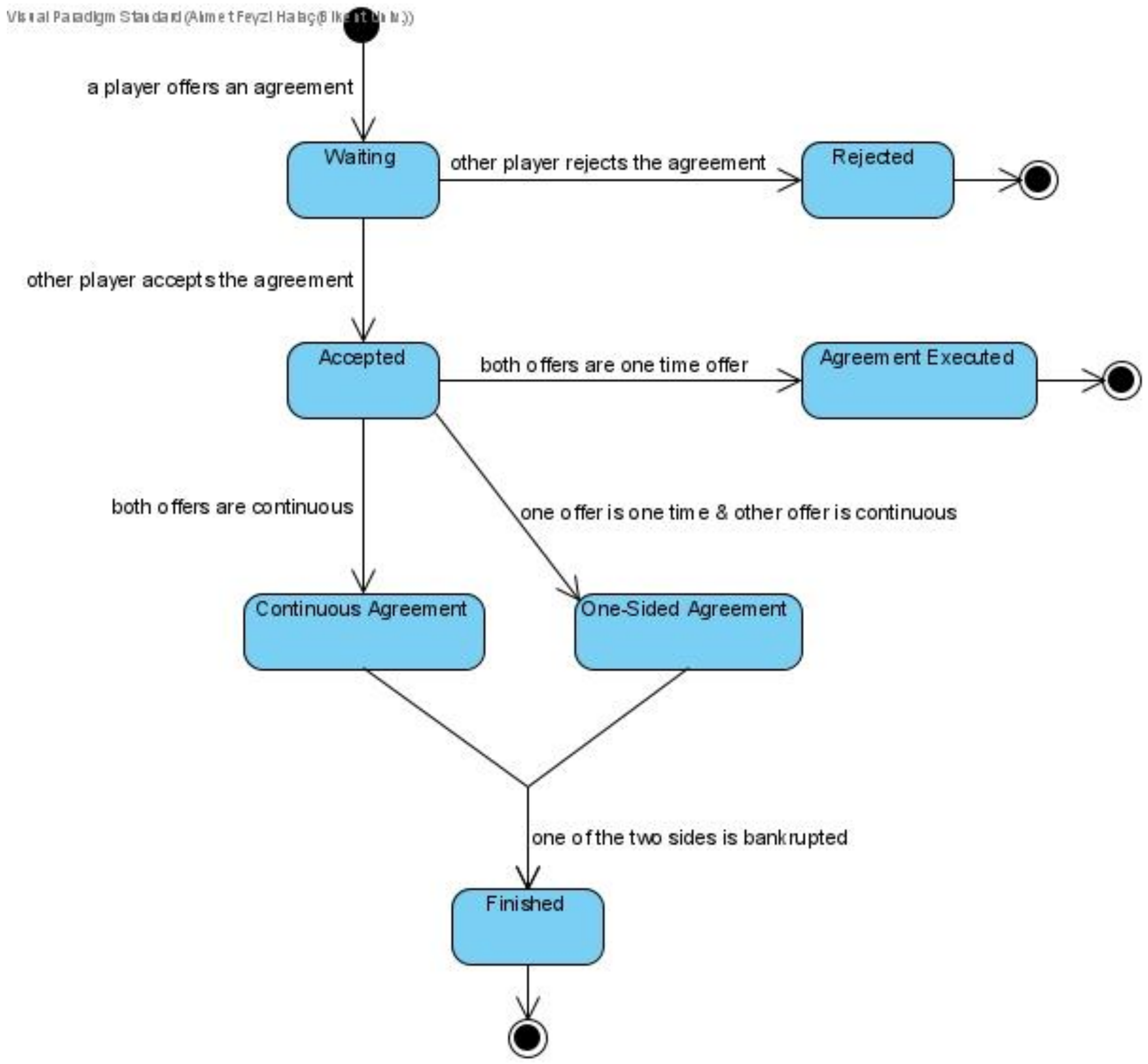

Player1
Buy Visited CityButton
: Game
Paris : City
1: press()
1.1: buyCurrentLocation()
1.1.1: [Player1.money > Paris.price] buyCity(cityName: Paris)
1.1.2: [Player1.money > Paris.price] setOwner(Player1)

Buy City Sequence Diagram

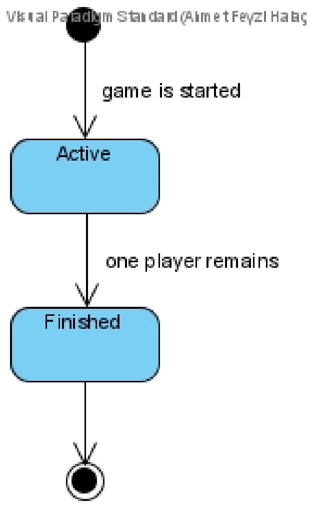

Player1
LiftMortgageButton
: Game
GameScene
: City
1: press()
1.1: createMortgagePopup()
1.1.1: <<create>>
Mortgage Popup
2: selectCity()
3: selectedCity
3.1: mortgageCity(city,false)
3.1.1: removeMoney(selectedCity.mortgageMoney)
3.1.2: successful
3.1.3: [successful] mortgage(false)

Lift Mortgage Sequence Diagram

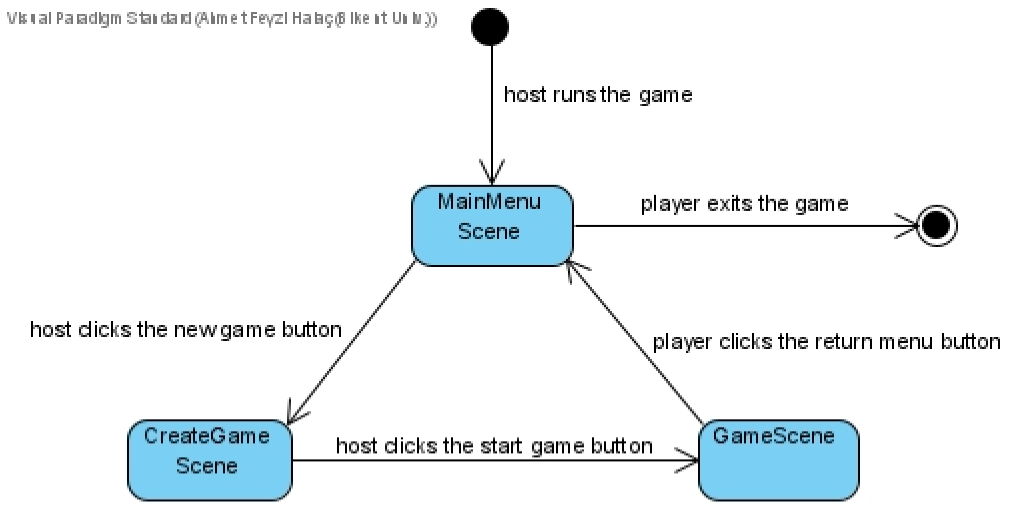
4.4 State Diagrams


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

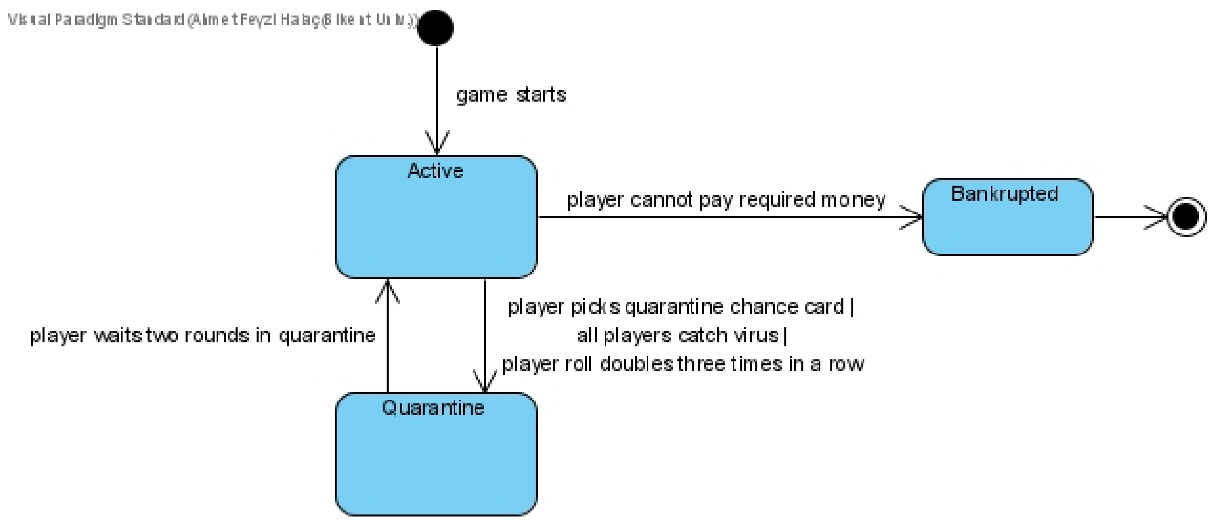
Agreement Class State Diagram



Game Class State Diagram

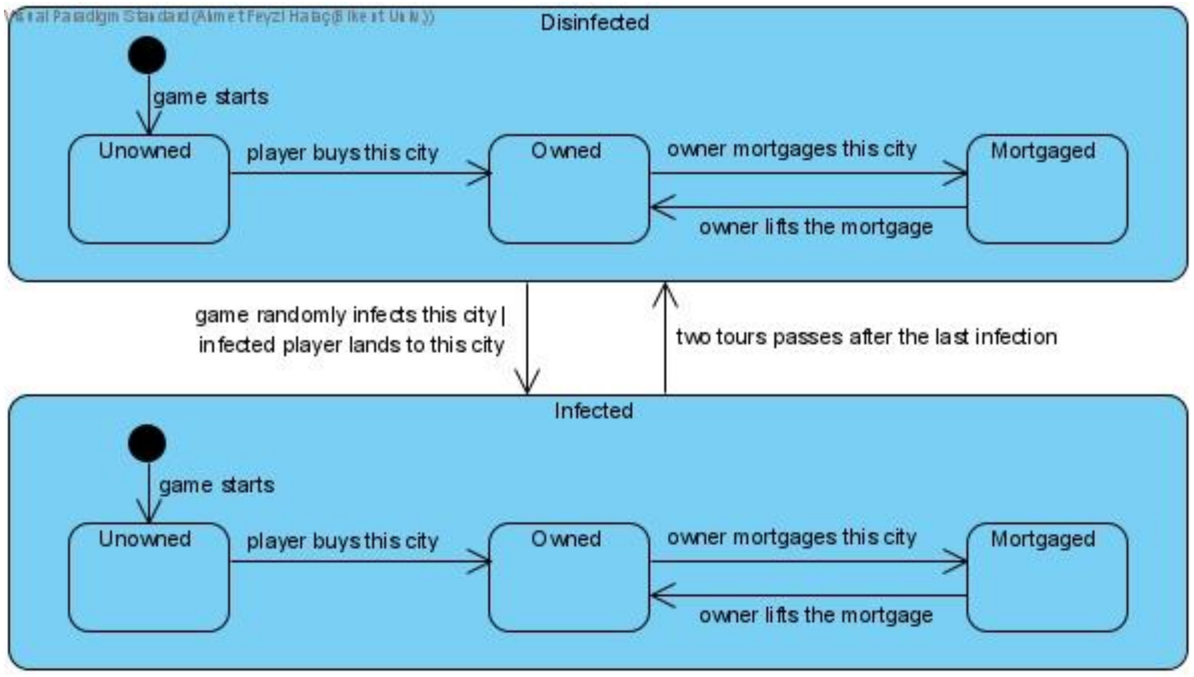

Visual Paradigm Standard (Ahmet Feyzi Halaç (Bilkent Univ.))
host runs the game
MainMenu Scene
player exits the game
host clicks the new game button
player clicks the return menu button
CreateGame Scene
host clicks the start game button
GameScene

SceneManager Class State Diagram



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

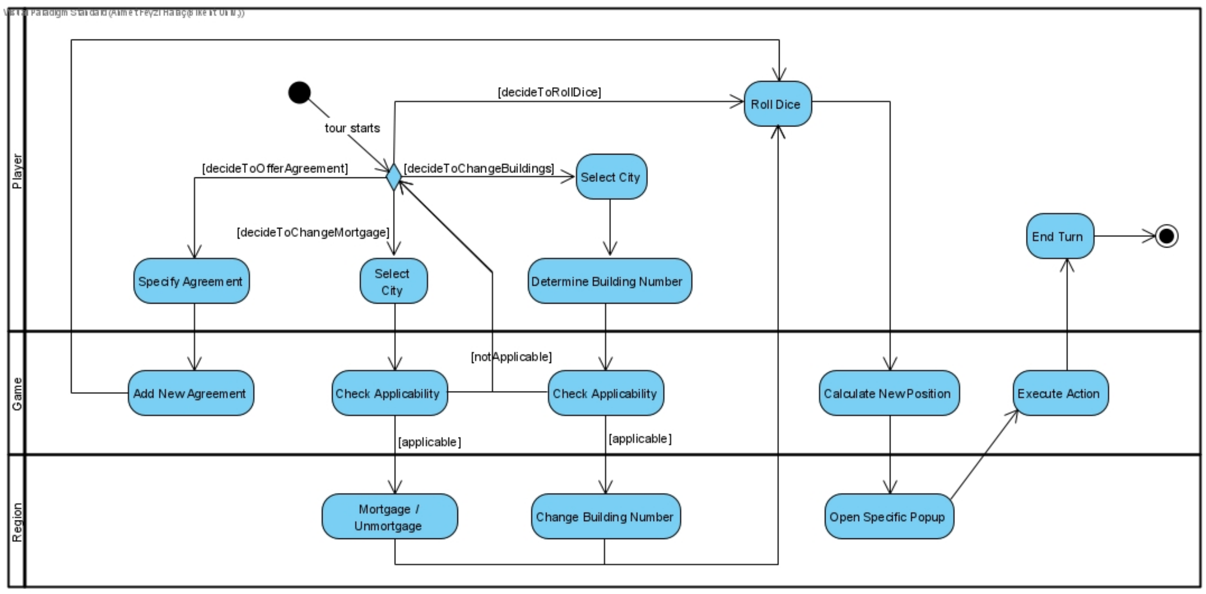
Player Class State Diagram



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

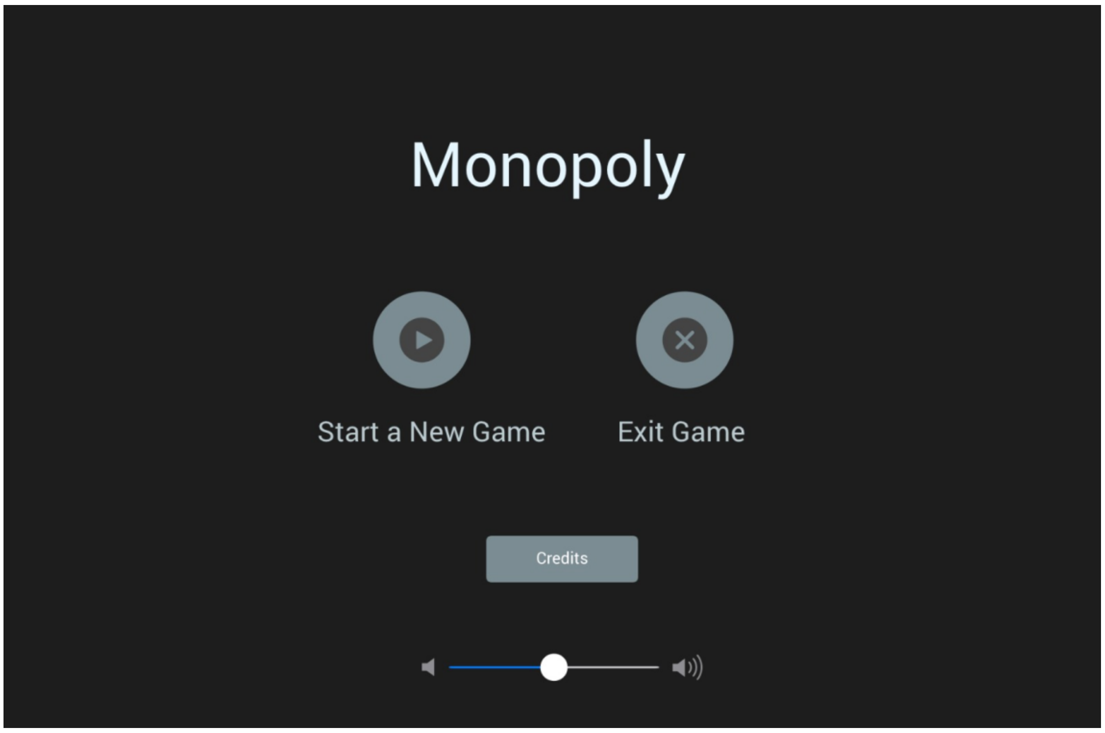
4.5 Activity Diagram



Activity Diagram of One Tour of a Player

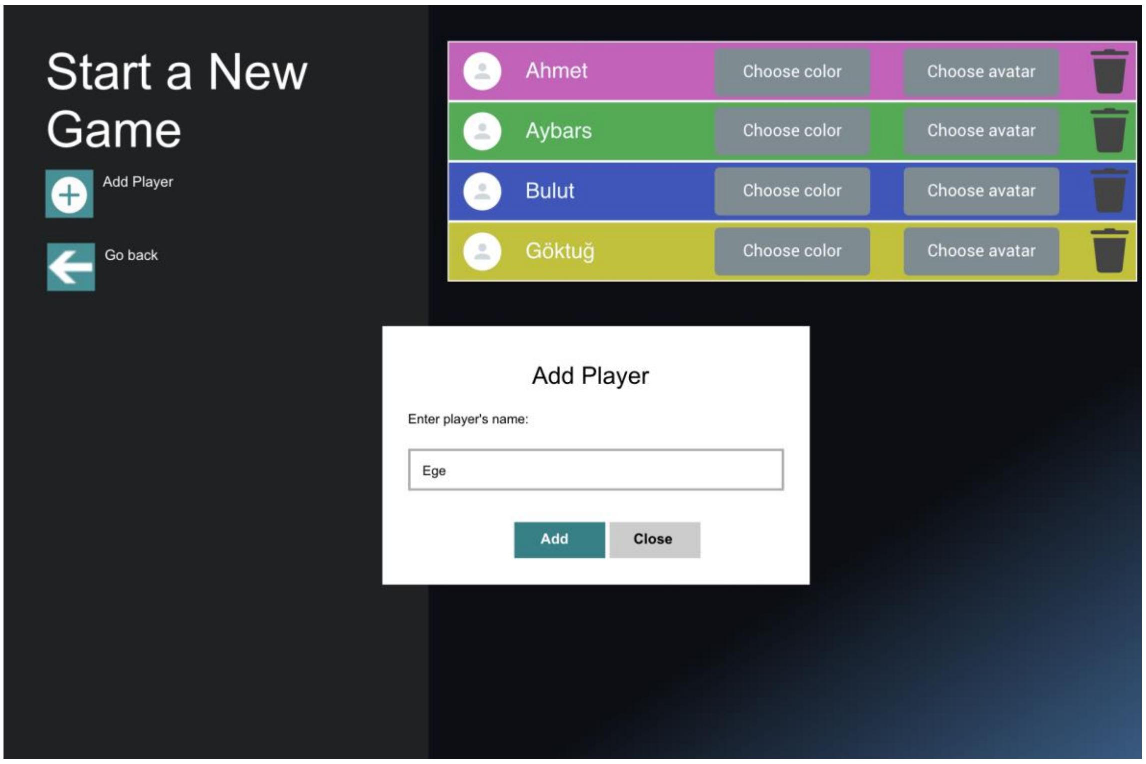
5. User Interface

Main menu:



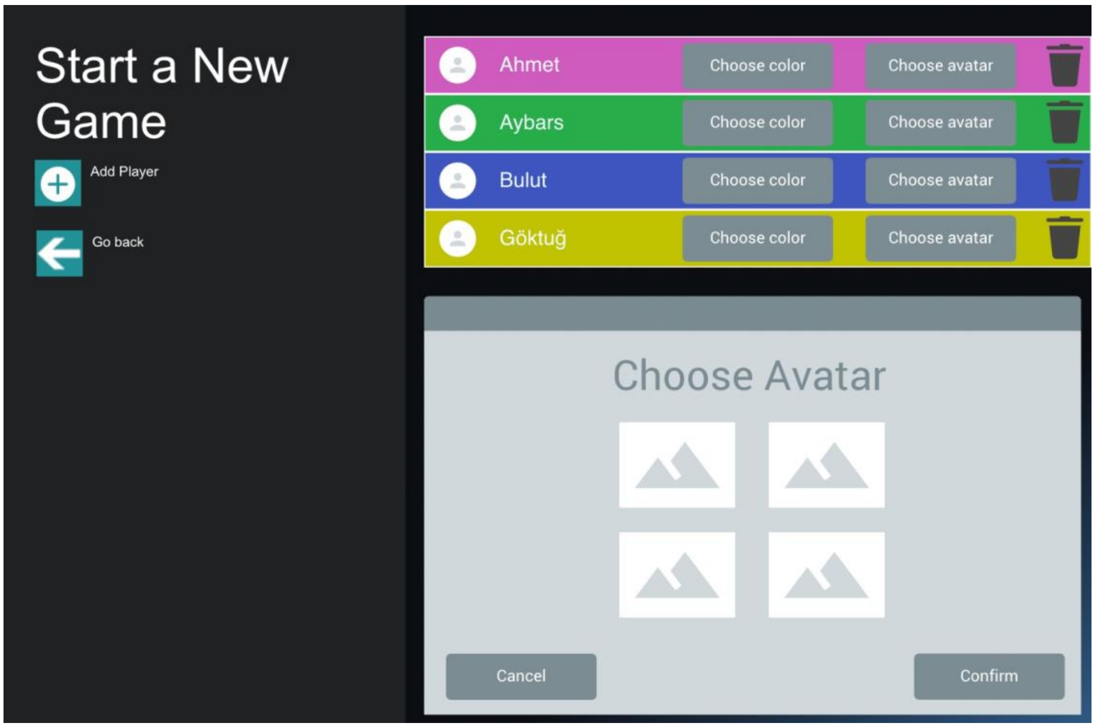
Creating a game player addition:

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



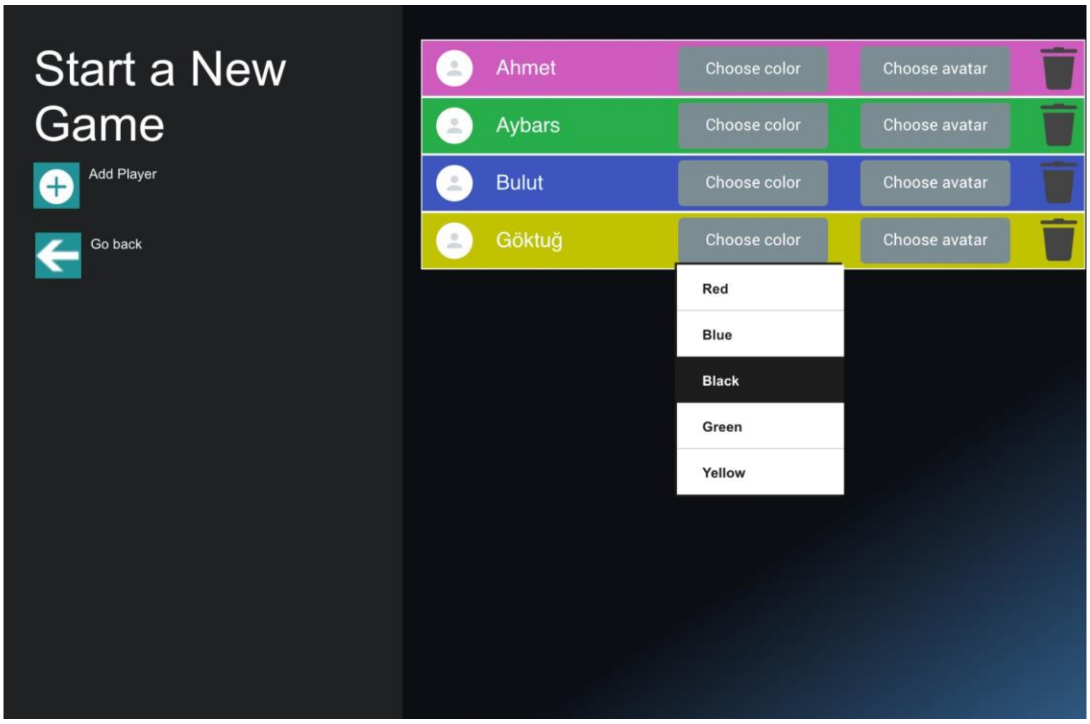
Creating a game avatar selection:

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



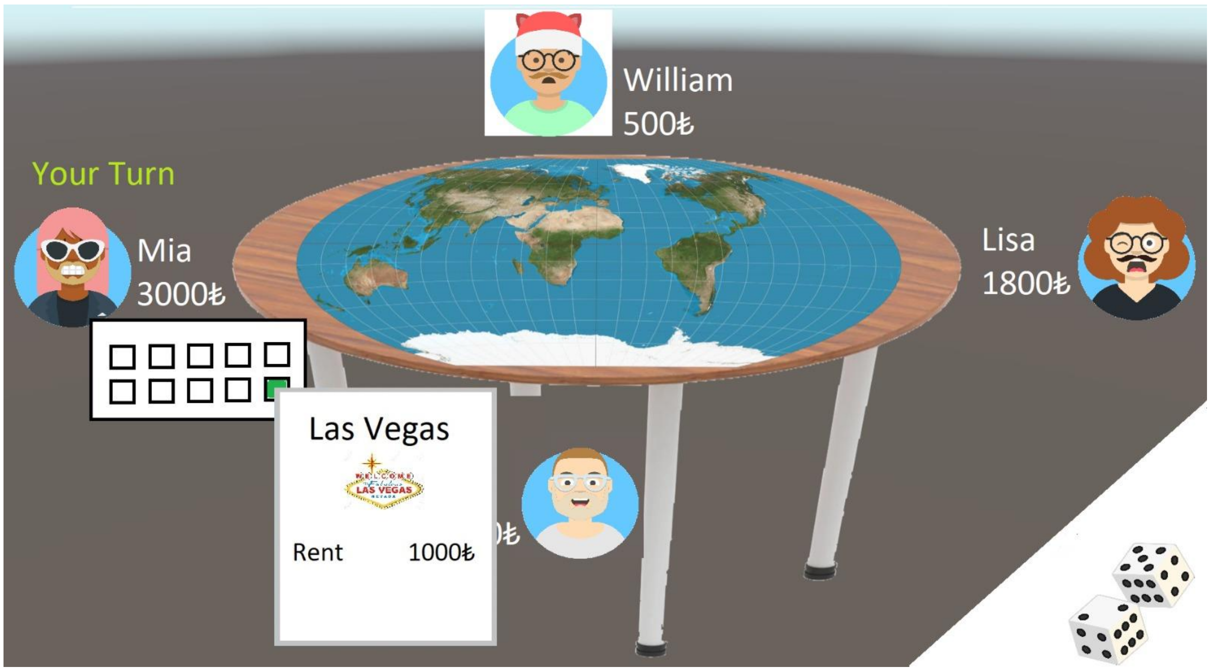
Creating a game color selection:

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
WELCOME TO Fabulous LAS VEGAS NEVADA
Rent
1000₺
)₺