Bilkent University

Department of Computer Engineering

CS 319 Project

Project short-name: Settlers of Anatolia

Analysis Report

Group members: Enes Merdane, Irmak Demir, Mehmet Alper Genç, İrem Kırmacı, Göksu Turan

Instructor: Uğur Doğrusöz
Teaching Assistant(s): Hasan Balcı

Analysis Report

Dec 1, 2019

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Object Oriented Software Engineering course CS319/1.

Contents

Analysis Report

Project short-name: Settlers of Anatolia

1. Introduction

The game we are implementing is called "Settlers of Anatolia," which is inspired by Klaus Teuber's famous board game Settlers of Catan, in which a number of players aim to settle piece of land by collecting resources, building roads and settlements with these resources and trading with each other. There is also a robber in the land who impedes the progress of the players that have a settlement near him, either by stealing from their resources or blocking resource collection. The game tracks the progress of the players using a victory point system, and the person who first reaches 10 victory points in the Classic Mode, or 20 victory points in the Crazy Mode, wins. Our group aims on implementing this game as a desktop platform game played with mouse and keyboard. The game will be played with bots. The programming language will be Java, and the JavaFX framework will be used to implement the graphical interface.

This report will document our progress in the analysis part of the project. The report will start by giving a brief explanation of how the game will be played. Then, the report will document the functional and non-functional requirements of the project, and afterwards will display the UML diagrams related to the analysis. Finally, the report will contain some early mockups of the game's graphical interface.

2. Overview

2.1. Buildings

There are three types of buildings, two that can be built on vertices of the map, and one that can be built on the edges of the map.

First type of building is the settlement, which requires 1 brick, 1 lumber, 1 wool and 1 grain to build, and must be connected to at least one of the player's roads. Another requirement for building a settlement is not to have another settlement in the neighbor vertices. In other words, a settlement cannot have neighbour settlements. Building a settlement gives one victory point.

The second type of building is the city, which requires 3 ore and 2 grain to build. A city is built on where the player already has a settlement, or in other words, a settlement is upgraded to a city. Upgrading a settlement to a city gives one extra point, or two in total.

The last type of building is the road, which is built on the edges of the map. The road requires 1 brick and 1 lumber to build, and must be connected to at least one of the player's settlements, cities or roads. Building a road by itself awards no points, but making a chain of roads with length equal to 5 grants gives the "Longest Road" card which is worth 2 victory points. If another player builds a longer road than the

player currently holding the achievement for the longest road, then the card and the 2 victory points pass to the new holder.

2.2. Tiles

In our map, tiles are shaped like hexagons and there are six different types of tiles. The main types are forest, hills, fields, mountains and pasture which generate lumber, brick, grain, stone and wool respectively. There is also a desert tile, which does not produce any resources. At the start of the game, all tiles except the desert tile are given a number from 2 to 12 as such: Numbers 2 and 12 are only given to one tile each, number 7 is not given to any tile, and the rest of the numbers in that range are given to two tiles each. When the result of a dice roll is the number of a tile, if the robber is not on that tile, the tile will produce a resource to players that have buildings in its vertices. For each settlement, the tile produces one of its resource to the player that owns it. For each city, the tile produces two of its resource. When 7 is rolled, each player with 7 or more cards must discard the floor of half their number of cards. Afterwards, the player that rolled 7 chooses another tile for the robber to go to, and steals one random resource from a player that has a building on the vertices of the new tile.

2.3. Trading

The players can trade among themselves, or trade with the bank.

To trade with other players, a player must send out a trade request during their turn. This trade request is shown to every other player to accept or decline. The trade is made with the player that first accepts the trade.

To trade with the bank, one must provide 4 of a type of resource in exchange for one of another resource.

2.4. Cards

There are 25 development cards placed randomly in a deck, which can be bought by players during their turns using resources, and can be used in their turns one turn after buying. These cards must stay hidden until their time of usage. The different types of cards are as follows:

Knight Cards: These cards allow the player to move the robber and steal one random resource from one player that has a building in the vertices of the new hexagon, similar to the case of rolling a 7. After using knight cards, they are revealed and placed in front of the player. Also, any player that reaches 3 knight cards gets a card named “Largest Army” which is worth 2 victory points. This card is passed to other players once they reach a greater number of knights than the current holder.

Victory Point Cards: These cards give the player one victory point each. They can only be revealed and used once the player has the potential to reach 10 points with them, or in other words, once the player can win.

Progress Cards: There are different types of progress cards that have different effects. They are as follows:

Road Building: These cards allow those who use them to construct 2 roads free of charge.

Year of Plenty: You can use these cards to draw 2 resources. These resources can be used to build in the turn that this card is used.

Monopoly: After using this card, you select one type of resource. All players must give all they have of this type of resource to you.

2.5. Turns

In the first turn, each player rolls the dice and the one who gets the highest result becomes the starting player and begins. The starting player places their settlement on a vertex and places a road adjacent to the settlement. Then, in the clockwise direction other players also set their settlements and roads.

In the second turn, the last person who placed his/her settlement places his/her settlement and road. Then other players place their settlements and roads in the counterclockwise direction. In the second turn, players do not have to place their roads or settlements connected to their other roads or settlements. In the second turn, players, right after placing their settlements, they gain their first resources for each hexagon terrain adjacent to the second settlement they have placed.

After these turns, starting player starts turn by rolling dice. After that, according to the total of dice values, players who has a settlement adjacent to the hexagon with the same number, gains one resource from that hexagon, and the player who has city gains two resources. Then, current player can trade with players or bank, buy an upgrade card, build settlements, upgrade settlements to cities, play upgrade cards or declare victory.

2.6. Crazy Mode

Our game will also include a Crazy Mode which includes new cards and mechanics that make the game more interesting.

One main feature of this new mode is the Events, which are Disasters and the Cybele Month. There are 3 Disasters: Flood, Earthquake and Wolf Attacks. At the start of every round (4 turns), there is a random chance of an event occurring. When the Flood event occurs, the first hexagon to produce Grain will not produce anything. When the Earthquake event occurs, all the cities are destroyed. When a Wolf Attack event occurs, the first hexagon to produce wool will not produce anything instead. When a Cybele Month event occurs, all hexagons produce double their resources.

Another addition of this new mode is new development cards. There are 2 new cards: Insurance and Anatolian Shepherd Dog**.** If a player plays the Insurance card, they will get back the resources they spent to build their cities which were previously destroyed by an Earthquake event. This resource is the sum of settlement building cost and the city building cost for each city destroyed. If a player plays the Anatolian Shepherd Dog, they will not suffer the effects of the Wolf Attack event.

2.7. Ending

The first player to achieve 10 victory points in the classic mode, or 20 points in the crazy mode during their turn is declared the winner and the game ends. If it is another player's turn, then they have to wait to win the game.

3. Functional Requirements

3.1. Current Requirements

3.1.1. Play Classic Game

Players will be able to play Classic Catan Game which is the desktop game version of the original Catan Board. This game includes only the base features of the board game which are building settlements, cities, roads, trading with other players, and buying and using upgrade cards.

3.1.2. Learn How to Play

Players will be able to learn how to play the game in the provided screen.

3.1.3. Set Preferences for the Game

Players will be able to turn on/off game music and sounds. And also they will be able to change the background image of the game.

3.2. Additional Requirements

3.2.1. Play Crazy Mode

This is a new play mode for the game in which players will be able to play the new version of Catan Board Game. In that mode, there will be "Events" which will occur at the beginning of some turns and will affect the players' game experience positively. This new feature includes events such as "Flood" which will disable grain collection from Hexagons for one turn. Another event Earthquake will destroy all cities on the map. Wolf Attack is the last event with negative effects, which will disable wool collection from Hexagons for one turn. The last event is "Cybele" event which will double the resources collected in this turn.

3.2.2. Play Additional Upgrade Cards

Additional upgrades cards will be included in the "Crazy Mode" of the game. These additional cards will provide players with the ability to protect themselves from the negative effects of events which are explained in 3.2.1. For

instance, Anatolian Shepherd card can be used to protect wool resources from Wolf Attacks. Another additional upgrade card is the Insurance card which will provide the players, who have lost their cities in Earthquake, with the resources they spent for building the city.

4. Non-Functional Requirements

4.1. Usability

Having a simple UI design is a very important issue for the project. Any player who knows the board game version already should not have problems related to the UI design while they are opening the game, changing settings, starting a game, and playing the game. To make this happen, the learning process should be easy for users at the first execution of the game. The usage should be easy to remember on the other executions as well. They should be able to find the necessary operations on the screen without any help from someone or any written documentation. To measure this criteria we will establish a usability test to two group of 10 people who both know and do not know the board game. We are aiming to measure less than 10 minutes for test users to discover most of the game's features.

4.2. Time Constraints

The game should not run at a slow speed so that the players have a smooth gameplay experience and not lose interest while waiting for the code to finish executing. Furthermore, the reaction time of the GUI to clicks and button presses must not exceed 0.5 seconds. To realize this, we increased the memory usage and the disk space of the application so that the code can reach the necessary information quicker.

4.3. Platform

The game will be playable in both Windows and Mac systems. We will test our system's functionality on each platform.

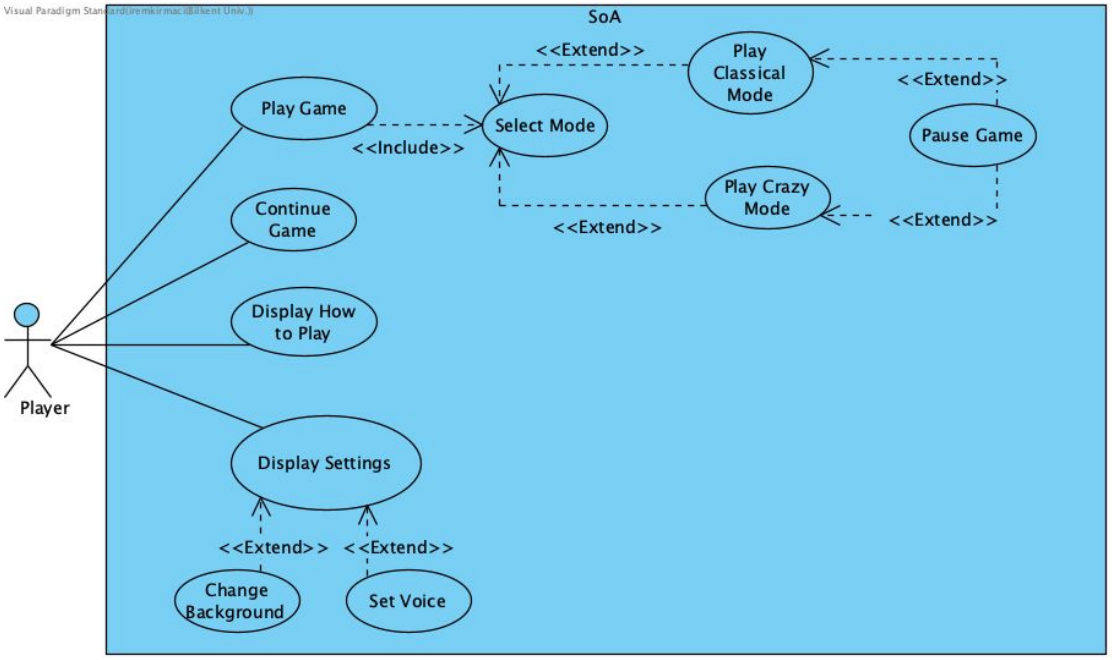
4.4. System Modifications

Extensibility and reusability are the important parts of the software. Therefore, this game will be implemented in a way that it can conform with new updates. It can be updated and extended according to the required changes which could be received by the user feedback or detected by a developer for a better service.

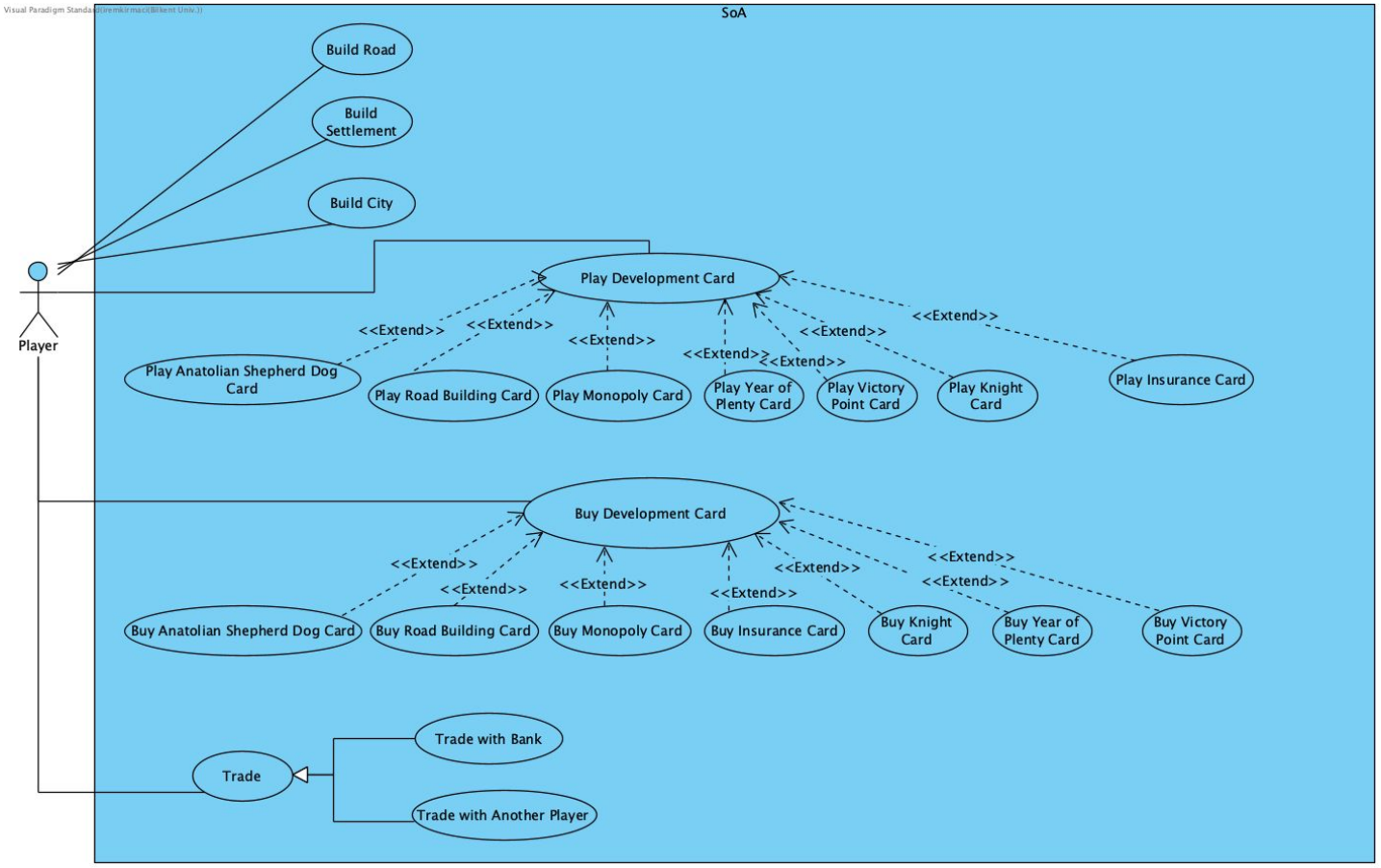
5. System Models

5.1. Use Case Models

Settlers of Anatolia Main Menu Use Case Model



Settlers of Anatolia Play Crazy Mode Use Case Model



5.1.1. Use Cases

Use Case #1

Use Case name: Play Classic Mode

Participating actors: Player

Entry Condition: Player opens the game; firstly player clicks Play Game and then Classic Mode button.

Exit Condition:

- One of the players reaches 10 points.
- Players want to exit game, one of the players clicks “Back to Main Menu” button and then players return to main menu.

Main Flow:

1. Player opens the game.
2. Player clicks “Play Game” button.
3. System displays the select modes screen.
4. Player clicks “Classic Game” button.
5. System prepares the game model and allows users to build their first road on the game tile.
6. System displays the game screen.
7. Player or one of bots reaches 10 points.
8. System displays “You Win” message or “You Lost” message and asks the player if they want to play again.
9. Player clicks “No” button and returns the main menu.

Alternative Flow #1:

1. Player clicks the “Yes” button.
2. System prepares the game model for the new game from scratch.

Alternative Flow #2:

1. Player clicks “Pause Game” button.
2. System saves the current state of the game.

Use Case #2

Use Case name: Play Crazy Mode

Participating actors: Player

Entry Condition: Player opens the game; firstly player clicks “Play Game” and then “Crazy Mode” button.

Exit Condition:

- One of the players reaches 20 points.
- Players want to exit game, one of the players clicks “Back to Main Menu” button and then players return to main menu.

Main Flow:

1. Player opens the game.
2. Player clicks “Play Game” button.
3. System displays the select modes screen.
4. Player clicks “Crazy Game” button.

5. System prepares the game model and allows players to build their first roads on the game tile.
6. System displays the game screen.
7. Player or one of bots reaches 20 points.
8. System displays “You Win” message or “You Lost” message and asks the player if they want to play again.
9. Player clicks “No” button and returns the main menu.

Alternative Flow #1:

1. Player clicks the “Yes” button.
2. System prepares the game model for the new game from scratch.

Alternative Flow #2:

1. Player clicks “Pause Game” button.
2. System saves the current state of the game.

Use Case #3

Use Case name: Learn How to Play
Participating actors: Player
Entry Condition: Player opens the game; player clicks “How to Play” button.
Exit Condition: The player clicks the back to main menu and then “Exit” button .
Main Flow:

1. Player opens the game.
2. Player clicks “How to Play” button.
3. System displays “How to Play” screen.
4. Player clicks back to main menu button.
5. Player clicks “Exit” button.

Use Case #4

Use Case name: Set Player Preferences about Game
Participating actors: Player
Entry Condition: The player opens the game and then clicks “Settings” button.
Exit Conditions:

- If the player didn’t make any changes, they exit by clicking the “Main Menu” button.
- If the player made settings, they exit by confirming the changes by first clicking Confirm and then clicking “Yes” on the frame that pops up.

Main Flow:

1. Player opens the game.
2. Player clicks the “Settings” button.
3. System display settings screen.
4. Player clicks Change Background button to change the background.
5. Player chooses from different background options.
6. Player clicks back to main menu button.

Alternative Flow #1:

1. Player clicks “Sound Off” button.

Use Case #6

Use Case name: Exit Game

Participating actors: Player

Entry condition: The player is in the main menu.

Exit Conditions:

1-Player wants to exit to desktop, player clicks the button Exit.

Main Flow:

1-Player opens the Main Menu.

2-Player clicks the Exit button.

3-The system asks the player if s/he is sure.

4-If player wants to exit the game and return to the desktop, s/he clicks “Yes”.

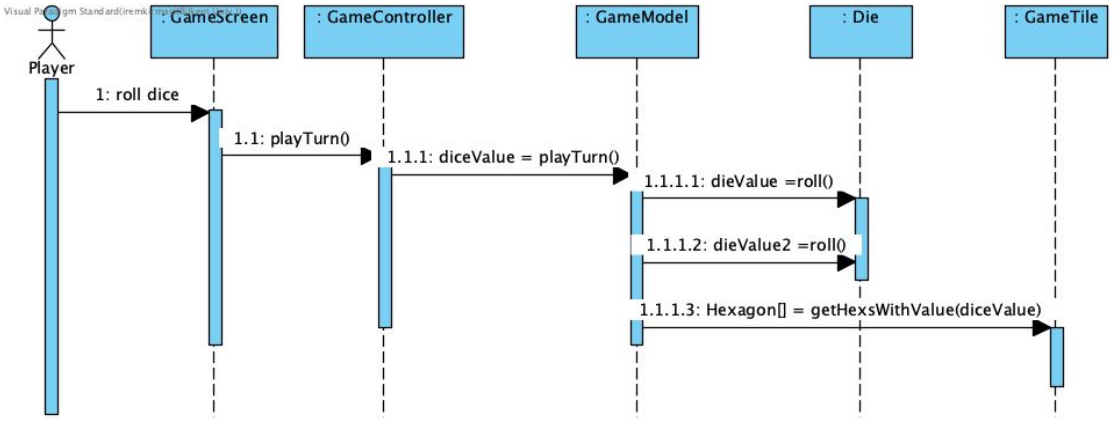
Alternative Flow:

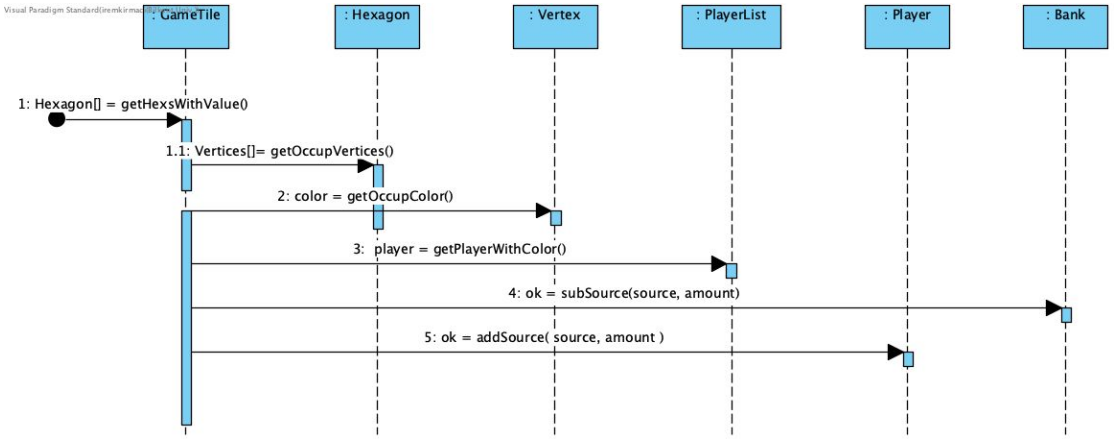
1-Player clicks “No” on the fourth step and stays on the Main Menu.

5.2. Dynamic Models

5.2.1. Sequence Models

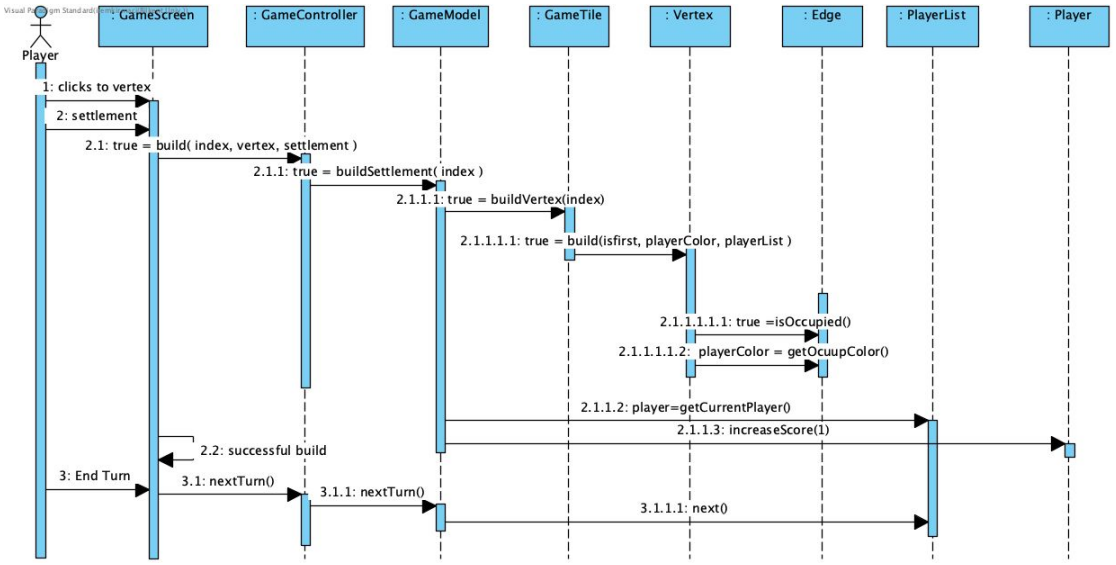
Sequence Model of Distribution of Sources





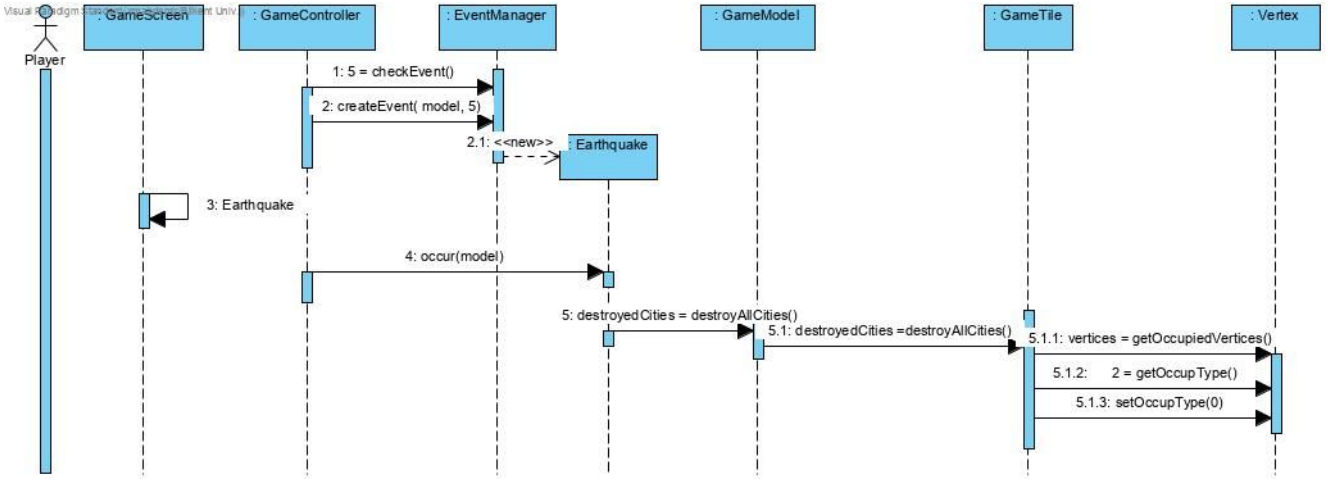
This above sequence, describes the scenario in which the players have already built some settlements and cities. The scenario starts with one player rolling the dice. Then, according to the dice value, to the owners of the buildings around the corresponding hexagons, the sources produced in the hexagons are distributed.

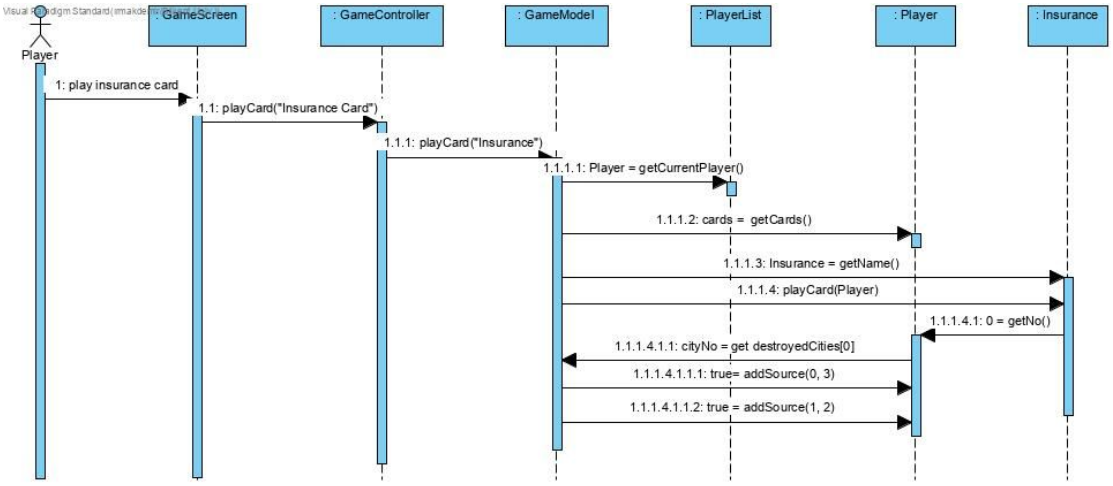
Sequence Model of Building Settlements



The scenario starts with player clicking on to the build settlement button and the vertex to build on to. The settlement is successfully
built on to that vertex and the score of the player is increased by one. The successful built message is displayed. Then the user click to end turn button and the turn passes to the next player.

Sequence Model of Earthquake Disaster

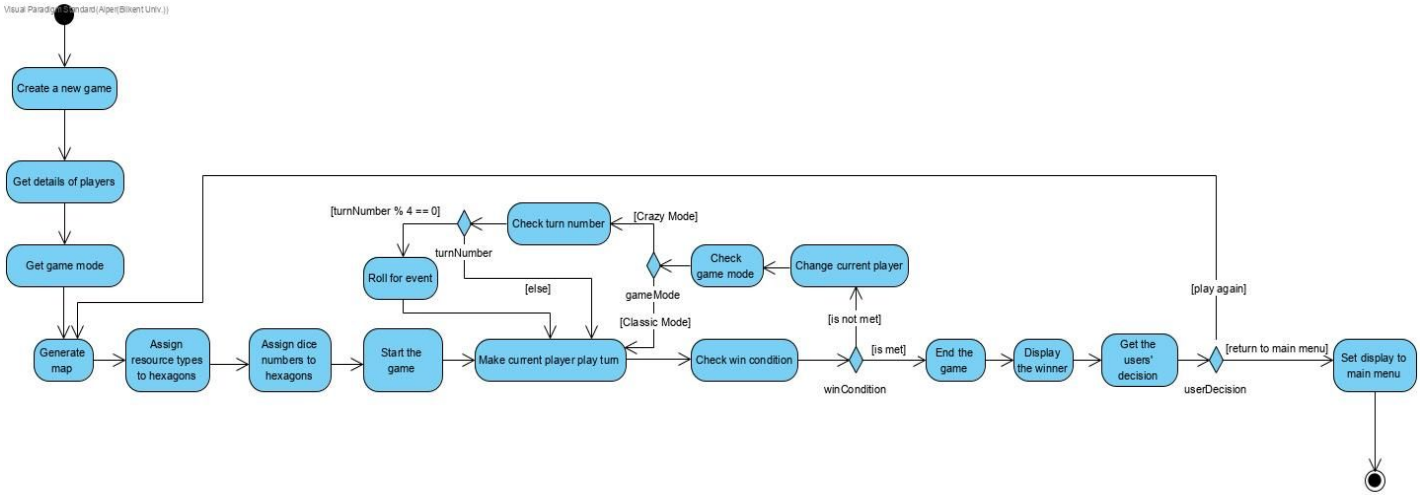




This is the sequence model of the new feature of our game which is the disaster earthquake. In this scenario, in the turn of the player, a random earthquake occurs and it destroys all the cities that has been built by the players. For each player, the number of cities of the player that has been destroyed is saved. Then the player decides to play insurance card that s/he owns, and gets 3 ore and 2 grain for each city of him/her that has been destroyed.

5.2.2. Activity Diagrams

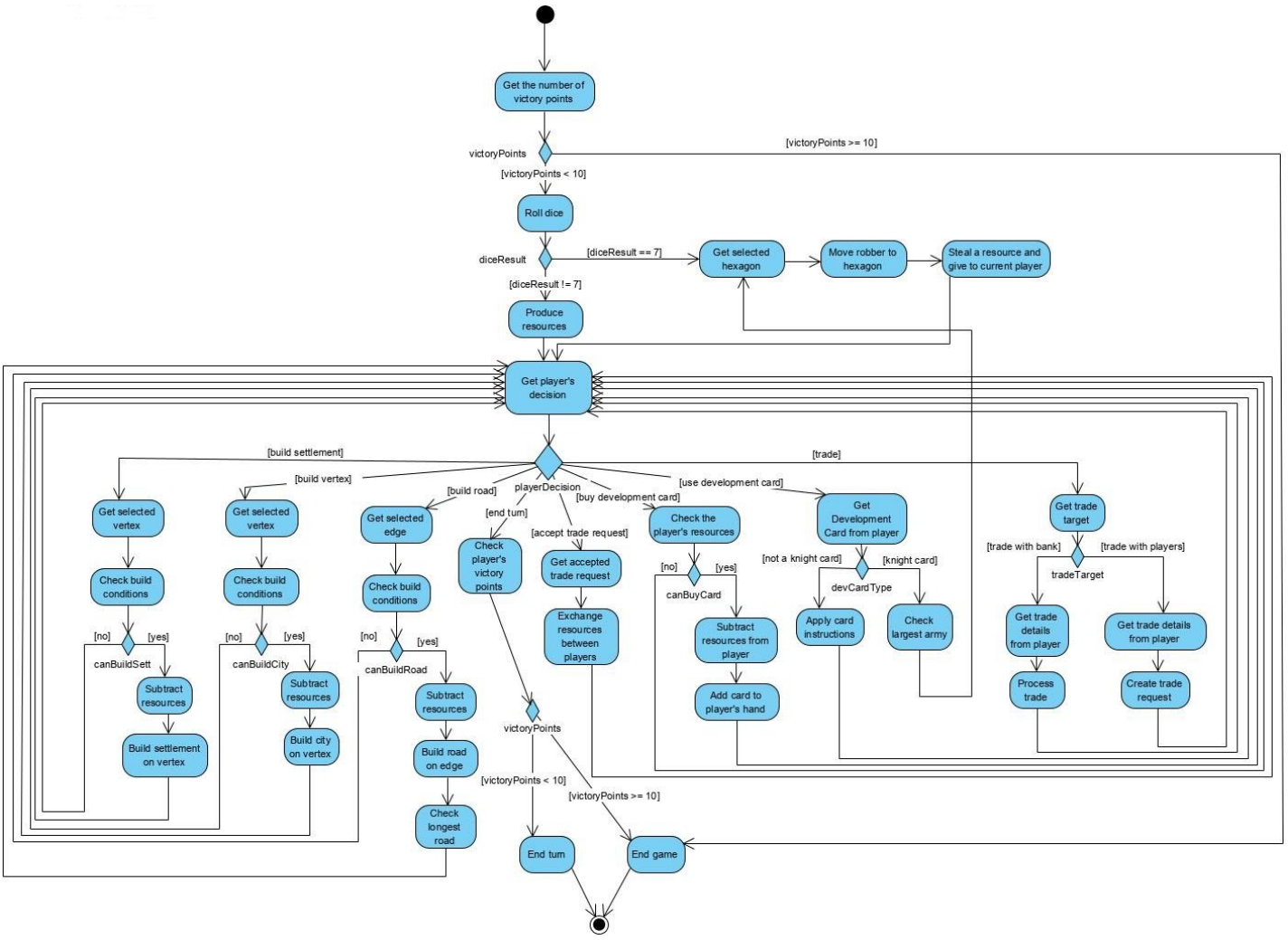
Playing the Game



This is an activity diagram that corresponds to the initialization and progression of the game. When the user clicks “New Game,” the game first creates a new game. Then, after the TTusers enter their details (number of players, the color of each player and the name of each player) and select their game mode (Classic Mode

or Crazy Mode), the game generates the map by creating the proper Vertex, Edge and Hexagon objects and connecting them appropriately. Then, the resource types and dice numbers of each hexagon are decided and assigned. Afterwards, the game starts with the first player playing their first turn. At the end of each turn, the system checks if the current player has greater than or equal to 10 victory points, or 20 for Crazy Mode. If not, then the system checks the game mode. If the game mode is Classic Mode, then the system goes to the next turn. If the game mode is Crazy Mode and the turn number is a multiple of 4, then it is the first turn of a new round, and the system must roll for a new event before starting the turn. When the win condition is met, the system ends the game and displays the winner. After that, the system asks the users if they want to play again, or return to the main menu. If they want to play again, the system generates a new game with the same player details and game mode. If they want to return to the main menu, the system sets the display to the main menu.

Playing the Turn



This is an activity diagram that shows how the system processes the activities that can be performed by a player while the turn is theirs. The only difference of the Crazy Mode for this activity diagram is that the victory point cap of the Crazy Mode is 20. Initially, the game checks whether or not the victory points of that player add up to the victory point cap. If so, then the game ends and that player is declared the winner. If not, the game continues.

At the start of each turn, the current player must roll the dice. When the user clicks “Roll Dice,” the system rolls the dice, and does one of two things depending on the result. If the result is 7, the user must click a new hexagon to move the robber there, and steal resources from them. The system first gets the selected hexagon,

moves the robber to that hexagon, and then steals a resource from one random player who has a building in that hexagon, and grants it to the current player.

Build Settlement

If the player chooses to build a settlement, then the system gets the selected vertex, and then checks the build conditions. The conditions of building a settlement are as follows:

1. The selected vertex should not contain a settlement or city.
2. The selected vertex must be adjacent to at least one of the current player's roads.
3. The player must have enough resources to build a settlement.
4. The selected vertex must not have a neighbor vertex with a settlement

If one or more of these conditions are not met, then the system disregards the action. If they are all met, the system subtracts the resources from the player and builds a settlement in the selected vertex.

Build city

If the player chooses to build a city, then the system gets the selected vertex, and then checks the build conditions. The conditions of building a city are as follows:

1. The selected vertex must contain a settlement built by the current player.
2. The player must have enough resources to build a city.

If these conditions are met, then the system subtracts the resources from the user and upgrades the settlement in that vertex into a city.

Build road

If the player chooses to build a road, then the system first gets the selected edge. Afterwards, just like building a settlement or a city, building a road also has a number of conditions:

1. The selected edge must be connected to at least one of the current player's buildings.
2. The selected edge must not contain another road.
3. The player must have enough resources to build a road.

The system checks these conditions, and if all are met, the system subtracts the resource cost from the player and builds the road on that edge. After building, the

system checks for the longest road. If the player has the longest road after building a new one, than they are awarded the “Longest Road” achievement which is worth 2 points.

Buy card

If a player chooses to buy a development card, they must have enough resources to buy a development card. If they do, then the resource cost will be subtracted from the player and the card will be added to the player’s hand for future use.

Trade

There are two different ways of trading in this game: with the bank, and with other players.

Trading with the bank requires that the player discards 4 of a resource to get 1 of another resource. After the player specifies the trade details, the system gets the details and makes the trade.

Trading with the players involves sending a trade request containing the offer and what the player wants in return. After the trade details are specified by the user, then this trade is created and added to the trade requests list for the other players to view.

Accept Trade Request

If a player would like to accept a trade request, then the system will get the accepted trade request and exchange the resources between the players accordingly.

Use card

Players can use the development cards they have one turn after buying them.

If the card to be used is a Knight card, then the system first checks for the largest army. If the current player has the largest army after building, then they are awarded the “Largest Army” achievement worth 2 victory points. If the card to be used is a Progress card, the instructions of this card are applied and the turn continues.

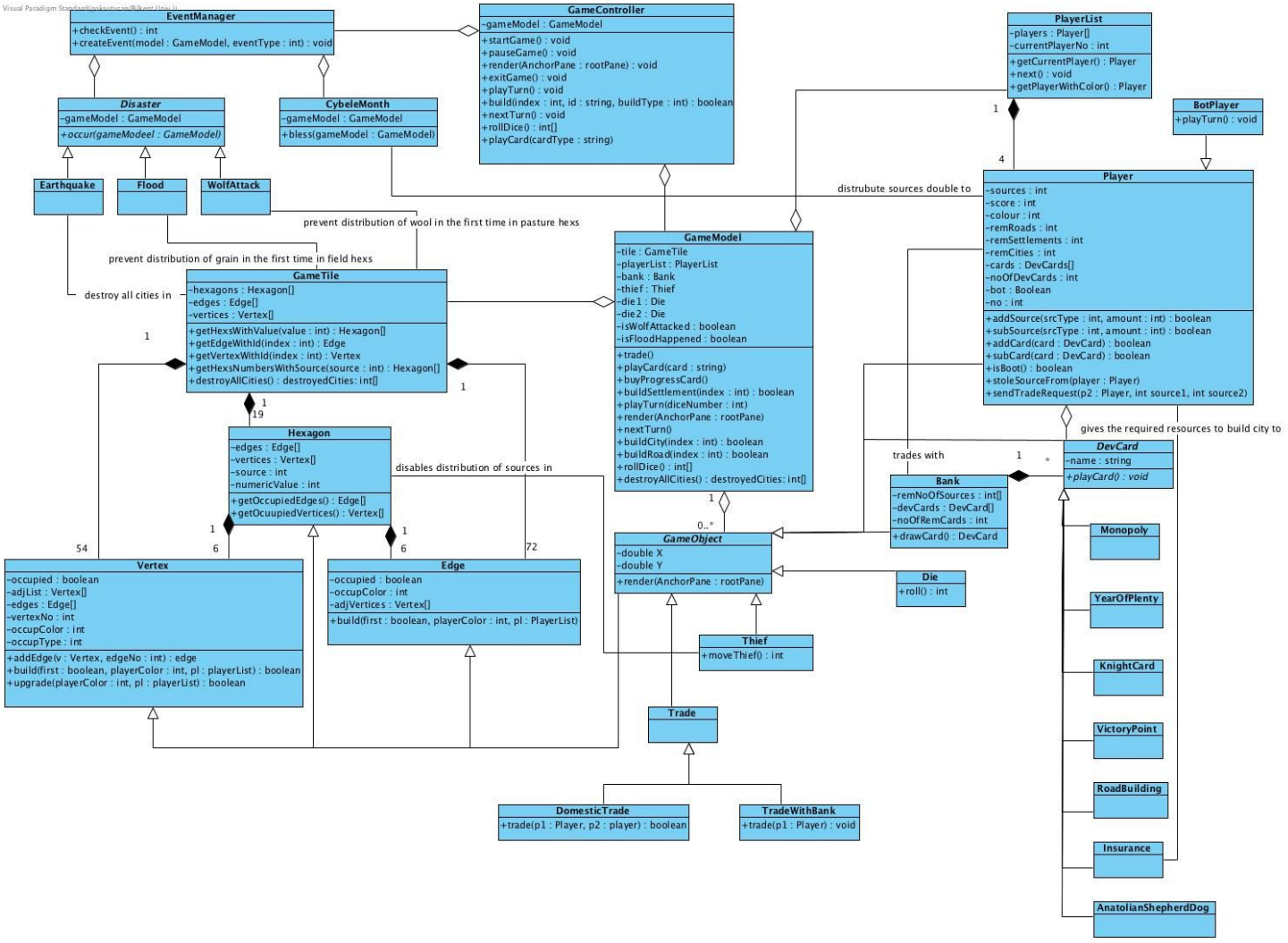
End turn

Finally, players can choose to end their turn. If so, then the system first gets the current player’s victory points. If the victory points are greater than or equal to the

victory point cap, the system ends the game and declares the current player the winner. If not, then the turn passes to the next player.

5.3. Object and Class Model

In the UML class diagram given below, the getter and setter functions for the private properties will normally be implemented but not included in the UML class diagram. The UML object class diagram was created regarding the Single Responsibility Principle of classes.



5.4. User Interface

5.4.1. Main Menu Screen

The background image is from [1].
This is the screen which the user sees after running the game.

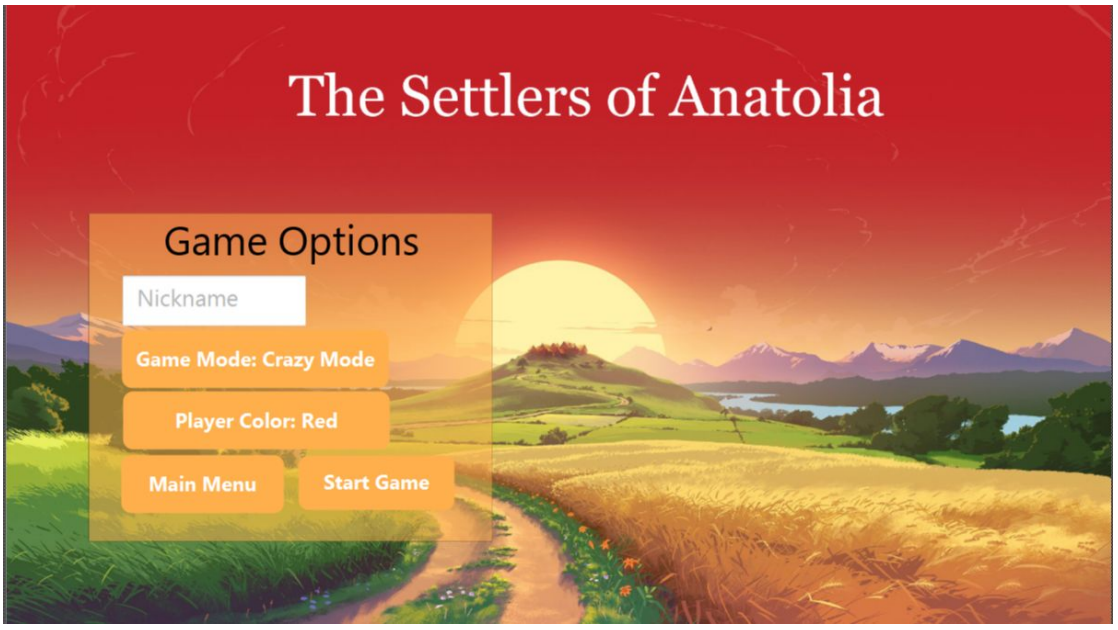
5.4.2. Settings

This menu is for users to adjust settings. They are able to turn the game music and the game sounds on or off, and change the background of the game from given collection.

The background image is from [1].

5.4.3. Game Options

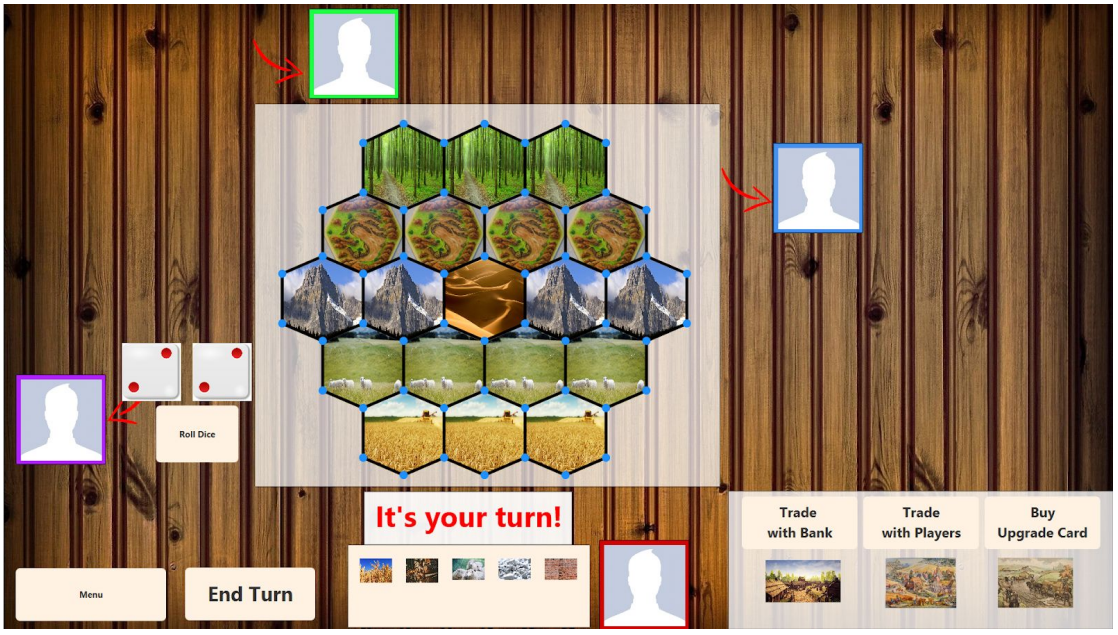
Game Options menu is the screen which opens after clicking the “Play Game” button from Main Menu. This screen is to set game mode and enter nickname of the player. And then, player can either click Start Game button and starts game or click Main Menu button and returns to the Main Menu.



The background image is from [1].

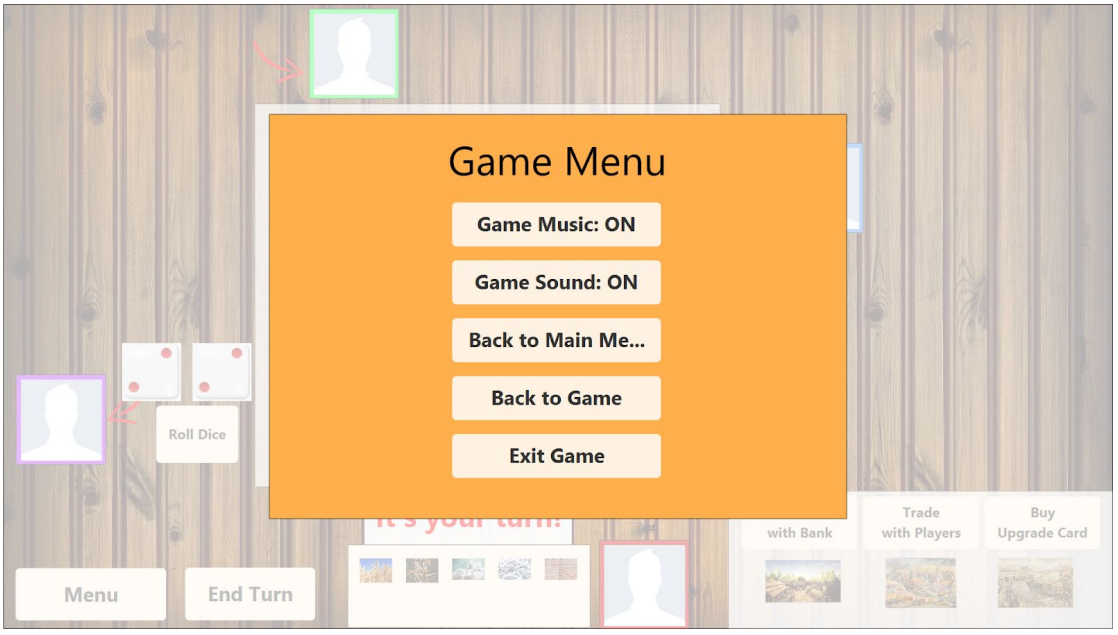
5.4.4. In Game

This screen is the game screen. It includes the game board, player avatars, the dice, resources.



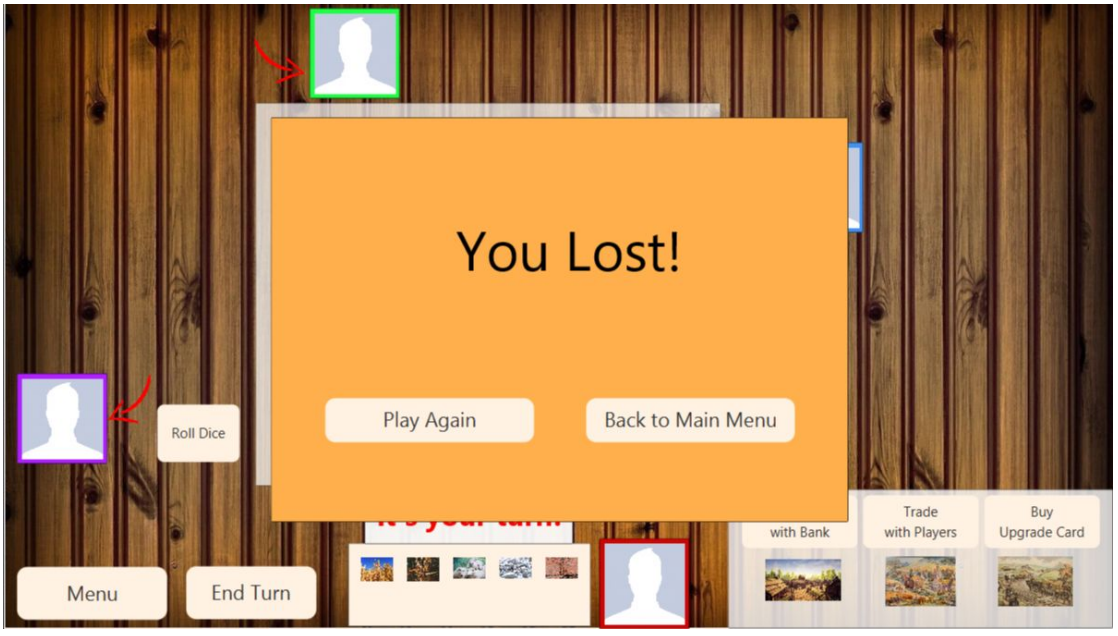
5.4.5. Game Menu

Game menu is the screen which opens when the player clicks Menu button while in game. On this screen, players will be able to set the game music and sounds on or off, go back to the Main Menu, exit game, or continue playing.



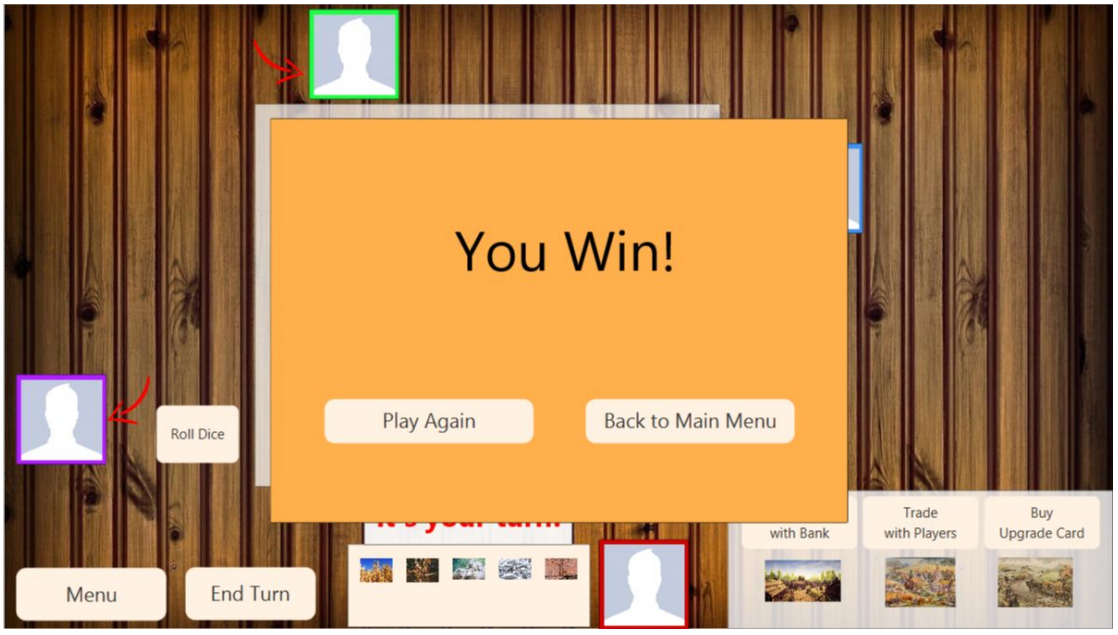
5.4.6. Game Lost

This screen appears when the player wins the game.



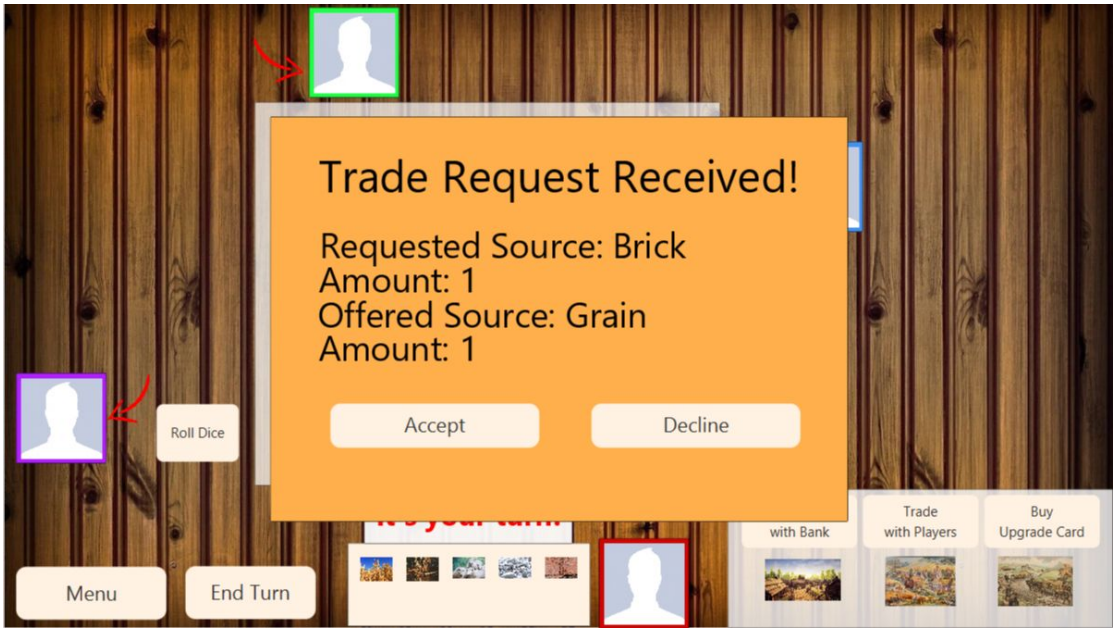
5.4.7. Game Win

This screen appears when the player wins the game.



5.4.8. Trade Request

This is the screen that appears on a player's view when another player sends a trade request.



6. Improvement Summary

Having spent time designing and implementing the project at the end of the first iteration has provided us with a clearer vision about the project, so we were able to represent the system more accurately with our diagrams and descriptions. Furthermore, we have added new features to our game.

Firstly, we have fixed the problems of our “Functional Requirements” section. We had written some parts of this section in terms of UI. We have replaced those definitions with subsections that describe the project in terms of functionality.

Then, we have tried to eliminate the vagueness in the “Usability” and “Time Constraints” subsections of our “Non-Functional Requirements” section by providing clearer descriptions. For example, in the “Time Constraints” subsection, we have specified a range for the reaction time of our clickables.

We have spent most of our time improving our models so that they best visualize our project while conforming to the UML rules. Firstly, we have improved the names of our use cases, and added another Use Case Diagram that visualizes the scenarios of playing the game. Secondly, we have improved our Activity Diagrams. The first iteration Activity Diagrams were more about the activities that the user performs, rather than the system. We have remade the activity diagrams, choosing better names for our nodes, using each item correctly, and adding the new features of our game. Thirdly, we have fixed the errors in our Sequence Diagrams, and added new Sequence Diagrams for our new features. Lastly, we have updated the “Object and Class Model” section by fixing the errors and adding related associations in the Class Diagram and adding new classes corresponding to the new features of our game such as EventManager class, Disaster classes and the CybeleMonth class. Furthermore, we added new functions and attributes to the already existing classes of our model to use our new features.

Finally, we have updated our UI mockups so that they are compatible with the UI of our first iteration.

7. Glossary & References

7.1. Glossary

Edge: The locations on the Catan game board on which the players can build a road.

Vertex: The locations on the Catan game board on which the players can build a settlement or a city.

7.2. References

[1]https://www.reddit.com/r/wallpaper/comments/bhqp3z/settlers_of_catan_1920x1080/